A STRUCTURED MULTI-BLOCK SOLUTION-ADAPTIVE MESH ALGORITHM WITH MESH QUALITY ASSESSMENT

Clint L. Ingram, Kelly R. Laflin, and D. Scott McRae
North Carolina State University
Raleigh, NC

SUMMARY

The dynamic solution adaptive grid algorithm, DSAGA3D, is extended to automatically adapt 2-D structured multi-block grids, including adaption of the block boundaries. The extension is general, requiring only input data concerning block structure, connectivity, and boundary conditions. Imbedded grid singular points are permitted, but must be prevented from moving in space. Solutions for workshop cases 1 and 2 are obtained on multi-block grids and illustrate both increased resolution of and alignment with the solution. A mesh quality assessment criteria is proposed to determine how well a given mesh resolves and aligns with the solution obtained upon it. The criteria is used to evaluate the grid quality for solutions of workshop case 6 obtained on both static and dynamically adapted grids. The results indicate that this criteria shows promise as a means of evaluating resolution.

INTRODUCTION

It is generally agreed that a geometry conformal structured mesh topology has demonstratable advantages for use in the numerical simulation of high Reynolds number viscous flows. However, the generation of geometry conformal structured meshes over complex shapes has proven to be a difficult task for two reasons: it is difficult to produce multiple block grids automatically that provide the necessary structured-mesh topology around the shape, and the grid must provide resolution of complex shear layers, shock waves, slip surfaces, etc. when the final location and extent of these features is not always known. Current research is underway to overcome the first difficulty, and is showing promise that multiblock structured meshes can be generated automatically.

It is the purpose of the present research to complement these advances by developing a means of adapting dynamically the initial structured mesh to the solution as it evolves, so that the features noted above plus any other chosen will be resolved automatically without laborious pre-clustering.

A brief review of the technique developed by Benson and McRae (refs. 1 to 4) for dynamic adaptation of 2-D and 3-D single block grids will be followed by a description of the extension of the dynamic adaptation algorithm to 2-D multi-block grids, including the adaptation of block boundaries (refs. 5 and 6) In all of the implementations of the solution adaptation algorithm presented herein, the terms which correct the final solution for cell volume variation with time have been included in the formulation. This ensures that temporal accuracy is preserved as the mesh translates to resolve moving features of the solution.

Finally, the question of cell shape (i.e., skewness, aspect ratio, etc.) effect on the solution has been a continuing issue for discussion among those who must defend the results obtained using structured meshes. An effort is presently underway to identify the errors which result from the interaction of the numerical algorithm with the mesh, including cell shape and mesh movement. Preliminary results from this research are included.

NUMERICAL PROCEDURE

Conservation Law

Conservation laws form the basis of our study of compressible fluid flow considered as a continuum. Since the volume over which conservation is being enforced can change independently, it is appropriate to examine briefly how these laws are obtained.

A conservation law results from the concept that a quantity or property may be physically conserved in both time and space. The mathematical definition of a conservation law results when the time rate of change of some quantity B summed over a given volume is shown to be equal to a quantity Γ, or

$$\frac{d}{dt}\int_V B dV = \Gamma. \tag{1}$$

In this application, B is either the density, linear momentum, or total energy. Applying the theorem of Leibnitz to the equation, two integrals result, one to account for the time rate of change of B summed over the volume and one for the motion of the boundary, *i.e.*

$$\frac{d}{dt}\int_V B dV = \int_V \frac{\partial B}{\partial t} dV + \oint_S B\vec{\omega}\cdot d\vec{s} = \Gamma. \tag{2}$$

It is common practice to assume the region or volume is not changing with time but only translating in space at the local fluid velocity. This is equivalent to assuming that the total mass is constant or the volume is "material." With this assumption, $\vec{\omega}$ becomes the local fluid velocity and thus,

$$\frac{d}{dt}\int_V B dV = \int_V \frac{\partial B}{\partial t} dV + \oint_S B(u\,\hat{i} + v\,\hat{j} + w\,\hat{k})\cdot d\vec{s} = \Gamma. \tag{3}$$

Equation 3 is easily reduced to the familiar form of the conservation laws, which are correct only if the volume of interest is fixed in magnitude or material. However, if the volume is allowed to expand or contract independently in time, then the volume boundaries no longer move at the local fluid velocity and a new relationship for $\vec{\omega}$ must be determined (ref. 7).

Consider a one-dimensional flow where the boundary of a region is moving with speed $\dot{x}$ and the fluid is moving at a velocity u. In this case, the fluid-surface interface velocity is $u - \dot{x}$, which is the velocity that should be used in the surface integral resulting from Leibnitz's Rule. Substituting the corresponding definition of $\vec{\omega}$ for a three-dimensional volume into Equation 3 results in

$$\int_V \frac{\partial B}{\partial t} dV - \oint_S B\left(\dot{x}\,\hat{i} + \dot{y}\,\hat{j} + \dot{z}\,\hat{k}\right)\cdot d\vec{s} + \oint_S B\left(u\,\hat{i} + v\,\hat{j} + w\,\hat{k}\right)\cdot d\vec{s} = \Gamma, \tag{4}$$

which is the correct statement of conservation for an arbitrary volume allowed to expand or contract in time.

If Γ, defined to ensure closure of the conservation laws, is substituted into Equation 4 and B is redefined to be the vector of properties that are conserved in fluid flow, $U = \left[\rho, \rho\vec{V}, E_t\right]^T$, a more familiar form of the conservation equations can be written as:

$$\int_V \frac{\partial U}{\partial t} dV - \oint_S U\vec{x}\cdot d\vec{s} + \oint_S \vec{A}\cdot d\vec{s} = 0. \tag{5}$$

$\vec{A}$ is a vector containing the flux components in the cartesian frame, $E\,\hat{i} + F\,\hat{j} + G\,\hat{k}$ and $\vec{\dot{x}}$ is the speed at which the surface is expanding or contracting, $\dot{x}\,\hat{i} + \dot{y}\,\hat{j} + \dot{z}\,\hat{k}$.

The second term accounts for the amount of U entrained or lost as a result of the change in volume over time, where $\vec{s}$ is the product of the surface unit normal vector and the surface area, and is the area of the projection of the given surface into the three spatial coordinate axes, *i.e.* $\vec{s} = s_x\,\hat{i} + s_y\,\hat{j} + s_z\,\hat{k}$.

Discretizing Equation 5 for a generic hexahedron changing in time from the n^{th} to the n^{th}+1 time level,

$$(UV)^{n+1} - (UV)^n \mp \left[(U\Delta V)_{i\pm\frac{1}{2},j,k} + (U\Delta V)_{i,j\pm\frac{1}{2},k} + (U\Delta V)_{i,j,k\pm\frac{1}{2}}\right]^n \pm\Delta t\left[\hat{E}_{i\pm\frac{1}{2},j,k} + \hat{F}_{i,j,k\pm\frac{1}{2}} + \hat{G}_{i,j,k\pm\frac{1}{2}}\right]^n = 0, \tag{6}$$

with $\hat{E}\Big|_{i+\frac{1}{2}} = \left(\vec{A}\cdot\vec{s}\right)\Big|_{i+\frac{1}{2}}$. $\hat{F}$ and $\hat{G}$ are similarly defined. ΔV represents the volume swept by the surface as it changes with time.

Modified Runge-Kutta Algorithm

The Navier-Stokes equations, plus the continuity and energy equations, are integrated in time using an explicit, multi-stage Runge-Kutta algorithm (ref. 8). The inviscid fluxes are described using the Advective Upwind Split Method of Liou and Steffen (ref. 9) and extended to higher-order spatial accuracy using MUSCL differencing and the van Albada and MINMOD limiters (refs. 10 and 11). For this work, a two-stage Runge-Kutta scheme will be utilized. The coefficients for each of the stages are $\alpha^1 = \frac{1}{2}$ and $\alpha^2 = 1$, which results in a second-order accurate scheme in time.

Although the changes in cell volume are independent of the time-advancement scheme, the manner in which this term is implemented is not independent and may decrease temporal accuracy. When the explicit Runge-Kutta algorithm is integrated as a single step, including the cell volume variation term, the mesh movement is at the $n-1$ level ($\Delta^{n-1}V$) and is based on mesh movement due to the prior time step. In an attempt to more closely couple mesh movement and solution, the Runge-Kutta algorithm is split into two steps. The first step integrates the terms in Equation 6 above which do not involve the ΔV term and can thereby be considered "steady" terms. The terms that involve ΔV, and thereby necessary to preserve temporal accuracy, are integrated in the second step of the procedure. Splitting the algorithm permits adaptation of the mesh based on weight functions determined from the steady portion of the integration. The second step of the integration algorithm is then applied to correct the solution for mesh movement to the $n+1$ time level. In the present work, the correction for mesh variation is first, thereby being equivalent to a first order interpolation of the steady solution to the new grid location. This algorithm is detailed below.

The modified, two-stage, time-advancement scheme that results from splitting the equations into a "steady" and an "unsteady" portion is as follows:

Stage i:

$$U^{(i)} = U^{(i-1)} - \alpha^{(i)}\frac{\Delta t}{V^n}\left\{\Delta_\xi \hat{E}^{(i-1)} + \Delta_\eta \hat{F}^{(i-1)} + \Delta_\zeta \hat{G}^{(i-1)}\right\} \tag{7}$$

Equation 7, the first step of the modified scheme, represents the advancement of the simulation in time for a static or non-moving mesh. In the definitions of the fluxes, $\vec{s}$ is evaluated at the n^{th} time level. The next step is to redistribute the nodes in the mesh using the method detailed below with

weight functions based on the solution at time level (2) obtained with the above equation. Finally, the time varying volume portion of the equations is used to advance the solution vector to the new mesh level by taking into account the amount of U that is entrained/lost by the expansion/contraction of the volume. This step is equivalent to correcting the convective velocity for a moving mesh. The equation used has the same form as the Runge-Kutta algorithm.

$$(UV)^{n+1} = V^n U^{(2)} + \left\{\Delta_\xi \left(U^{(2)}\Delta^n V\right) + \Delta_\eta \left(U^{(2)}\Delta^n V\right) + \Delta_\zeta \left(U^{(2)}\Delta^n V\right)\right\} \tag{8}$$

In Equation 8, the terms $\Delta^n V$ represent the change in volume between the n^{th} and $n^{th}+1$ time level.

ADAPTIVE ALGORITHM

The adaption process, to ensure resolution of solution features as they translate and evolve, takes place between the split steps of the integration scheme. Weight functions are based on the first stage of the integration scheme. A center of mass scheme is then used to relocate the points in a transformed parametric space. A straightforward truncation and interpolation scheme is used to determine the new location in physical space corresponding to the new parametric coordinates. An overview of these steps follows.

Weighting Function

The primary goals of an adaptive procedure must be to improve solution accuracy. A by-product of increased accuracy should be the ability to locate, identify and determine the extent of all features of the solution with increased certainty. A conventional examination of the truncation error of discrete integration techniques results in an infinite series of terms consisting of derivatives of the dependent variables multiplied by functions of the discrete spacing.

In general, these truncation error terms are large near discontinuities and rapidly changing regions in the flow. Therefore, it was decided to use the difference of the dependent variables as the basis for a suitable clustering function. Higher differences (usually second) may also be used if sufficient nodes are available to resolve them. The weight function is formed using a linear superposition of the differences where σ_k is a biasing coefficient, ϕ_k is the magnitude of the gradient, and k indicates a given flow variable. The maximum value of the difference for each flow variable varies, so a second coefficient will be introduced to scale the respective gradients thereby insuring the desired weighting (usually equal peak values) before biasing. The complete weighting function is then determined by:

$$\omega = \sum_k \sigma_k \alpha_k \left|\phi_k\right|. \tag{9}$$

In order to provide automated control of the degree of adaption, an average weight function is first determined along each coordinate line. This weight function is then limited according to predetermined functions of this average:

$$\omega_{min} \leq \omega \leq \omega_{max}. \tag{10}$$

After limiting, the weight function is then smoothed to control cell skewness. For the baseline weighting function, biasing coefficients, percentage values, and ratios are input at the beginning of the program. The remainder of the process occurs without user interaction.

Transformation To Parametric Plane

Using a similar idea to that used for obtaining a computational space for a single block grid, physical space can be mapped into a separate parametric space. The mapping is general and the standard requirements for one-to-one correspondence of grid-points and smooth mesh lines are maintained, thereby guaranteeing an inverse transformation. This initial trasnformation between physical and parametric space remains fixed during the computation and is used as a reference for remapping the changed grid locations into physical space after adaption. "Fixed" in this instance means that the original transformation does not change with time. The reason for performing the adaption in a parallelepiped in parametric space is twofold: intricate boundaries in the physical space will map into a plane in the parametric space, and a simple equation results for remapping the adapted parametric space into corresponding new locations in physical space. Adaption in parametric space will determine new values of ξ, η, and ζ for each node. The next step is to determine which metrics of the original transform must be used to map the new node locations (ξ, η, and ζ) to physical space such that the original transform is preserved. This maintains the ordering of the grid-points in both spaces.

Since the original mapping defines a parallelepiped in the parametric space and $\Delta\xi$=$\Delta\eta$=$\Delta\zeta$=1 by definition, the original nodes or grid-points are located at integer values which correspond directly to the i, j, k which are used to reference the arrays.

$$int(\xi^o, \eta^o, \zeta^o) = i, j, k \tag{11}$$

Adaption takes place in the parametric space, after which a mapping to determine the new x, y, and z locations from the new ξ, η, and ζ positions must be obtained. This process begins with a discrete approximation to the differential dx:

$$\Delta x = x_\xi \Delta\xi + x_\eta \Delta\eta + x_\zeta \Delta\zeta \tag{12}$$

Since the approximations to the differentials in the above equation are just differences, they are chosen to be the new mesh node location, referenced with i, j, k , minus a nearest original position, denoted with the superscript ^{o}. The metric derivatives are also identified with the superscript ^{o}, since the transform is only determined initially.

$$x_{i,j,k} - x^o = x^o_\xi(\xi_{i,j,k} - \xi^o) + x^o_\eta(\eta_{i,j,k} - \eta^o) + x^o_\zeta(\zeta_{i,j,k} - \zeta^o) \tag{13}$$

If the point at i, j, k is moved to a new position in the parametric space, the corresponding new position in the physical space must be determined. The vertex of the cube of the original parallelepiped that contains the new ξ, η, ζ position and is nearest the origin of the parametric mesh can be found by rounding these values down to integers.

$$\begin{aligned} l &= int(\xi_{i,j,k}) \\ m &= int(\eta_{i,j,k}) \\ n &= int(\zeta_{i,j,k}) \end{aligned} \tag{14}$$

The vertex of the cube of the original parallelepiped that is closest to the new ξ, η, ζ position is given by the nearest integer function:

$$ln = nint(\xi_{i,j,k}) = l + 1$$

$$mn = nint(\eta_{i,j,k}) = m + 1$$

$$nn = nint(\zeta_{i,j,k}) = n \tag{15}$$

Recall that ξ, η, and ζ were defined to be integers that corresponded directly to the reference coordinates i, j, k and therefore, the values defined in Equations 14 and 15 correspond directly to the array positions for x, y, and z of the original grid-point at those respective vertices.

The metrics, x_ξ, x_η, and x_ζ, are appoximated such that they represent the distance between adjacent nodes in the ξ-, η-, and ζ-directions. The metrics are stored in arrays as forward differences and therefore, they are based at the point l, mn, nn for the ξ-direction, ln, m, nn for the η-direction, and ln, mn, n for the ζ-direction. By using the integer value of ln in place of ξ^o in Equation 13, this will subtract the distance $x^o_\xi(\xi - \xi^o)$ if $\xi_{i,j,k}$ is closer to the ξ-axis than the nearest original point and add the distance if $\xi_{i,j,k}$ is greater than the nearest original point. The result is similar for η^o and ζ^o. Therefore, a final expression for the new value of x in the physical space is:

$$x^n_{i,j,k} = x^o_{ln,mn,nn} + x^o_{\xi_{l,mn,nn}}(\xi_{i,j,k} - ln) + x^o_{\eta_{ln,m,nn}}(\eta_{i,j,k} - mn) + x^o_{\zeta_{ln,mn,n}}(\zeta_{i,j,k} - nn), \tag{16}$$

which is a Taylor series expansion in three dimensions utilizing the initial grid as a reference grid. Similar equations can be derived for y and z, by substituting for x.

The above can be shown to preserve the original boundary shape. Choosing the boundary where η=const=1., note that a term drops out of Equation 16 leaving

$$x^n_{i,j,k} = x^o_{ln,mn,nn} + x^o_{\xi_{l,mn,nn}}(\xi_{i,j,k} - ln) + x^o_{\zeta_{ln,mn,n}}(\zeta_{i,j,k} - nn). \tag{17}$$

This maps into the surface in the physical space that corresponds to the original boundary on which $\eta_{i,j,k}$=constant.

Adaption Procedure

With exceptions noted below for multi-block grids, performing the adaption in parametric spaces allows the restrictions on movement in the method of mean-value relaxation (ref. 12) and the method of minimal moments (refs. 13 and 14) to be removed.

Considering the grid-points to be point masses with the weighting function analagous to the mass, the location of the center of mass can be determined for the computational cell in three dimensions as:

$$\xi_{cm_{i,j,k}} = \frac{\sum_{k-1}^{k+1}\sum_{j-1}^{j+1}\sum_{i-1}^{i+1}\omega_{i,j,k}\xi_{i,j,k}}{\sum_{k-1}^{k+1}\sum_{j-1}^{j+1}\sum_{i-1}^{i+1}\omega_{i,j,k}} \tag{18}$$

This determines the movement of the point at i, j, k to a localized center of mass. This calculation is repeated for every point in the parametric domain with the boundaries being calculated using a reduced stencil. The grid points are locally redistributed until the entire grid of weights is globally at rest or has moved the desired amount.

For single-block grids, adaption in parametric space effectively eliminates crossover of mesh lines if the initial grid is generated by an elliptical grid generator. Crossover will result only when the center of mass of the stencil is outside of its original domain, which is most likely to occur at concave boundaries. These boundaries are not present in parametric space for single block grids.

Implementation Of The Algorithm

The adaptation algorithm is utilized in the following way: the transform and inverse transform for mapping the physical space into the parametric space is computed initially; the first part of the integration step on the governing equations is completed; the grid is adapted based on the solution from that step; the second step of the integration is performed; the transformation for the governing equations is recomputed; and the process starting with the integration step is repeated until the grid reaches a steady-state or until a total time in the solution evolution is reached. When the grid reaches a steady-state, this implies that only small changes are occurring in the flow variables that cause small changes in the weight function distribution. The grid motion is then ended and the simulation continues on the static adapted grid until the desired flow convergence is obtained. Although the algorithm was developed to be a dynamic solution-adaptive method, it can also be used as part of a grid refinement study. The first step is to run a simulation to convergence on an initial grid. Using the adaptive algorithm, the grid-points of the original grid are relocated based on the solution, which is interpolated to the new grid as part of the algorithm. The simulation is rerun using the clustered mesh as the initial grid. This can be repeated until the desired resolution is attained.

MULTI-BLOCK GRIDS

The multi-block solver/adaptor scheme is designed to accomodate general multi-block geometries and permit the application of the solution-adaptive mesh algorithm to structured, multi-block initial grids by specifying only the block connections (refs. 5 and 6). Therefore, the coupling of the multi-block solver with the multi-block solution-adaptive mesh algorithm allows dynamic adaption for flows over complex structures for which single-block meshes are inadequate.

In this case, effective adaption without *a priori* knowledge of the flow requires that the block boundaries be adapted in addition to the interiors. This implies that new mesh locations from one block can move on to the fixed reference locations of an adjacent block. Logic must, therefore, be installed to maintain overall connectivity and continuity of the adapted grids as this occurs.

The block connectivity is governed by a set of specified block splicings. The block splicings can be defined by any of ten possible combinations. That is, any block side can connect to any other block side. The only limitation is that both blocks of a splicing must contain the same number of mesh cells in the direction along the splicing. Each block may have its own orientation (ξ, η direction). Therefore, there may not be a global ξ, η direction. This feature not only allows the solver/adaptor to handle more complex grid structures in which a global ξ, η direction is not possible, but also allows imbedded singular points to reside in the grid domain. These splicings are defined in an input file along with the dimensions and boundary conditions of each block. The grid-blocks are read in as separate files, and can be generated using GRIDGEN or any other appropriate grid generator.

MULTI-BLOCK SOLVER

The extension of the solver to a multi-block grid is straightforward. Each block of a grid is treated as a separate domain and boundary conditions enforce propagation of data from adjacent

blocks or boundaries. Conditions along splice block boundaries are determined by continuity with adjacent blocks. Reference 6 describes the procedure and illustrates how the process provides a known boundary condition along the splices of the blocks in both directions. This process is dubbed a *splice boundary condition* and is performed at the same time that all other boundary conditions (viscous wall, free stream, etc.) are enforced.

MULTI-BLOCK ADAPTOR

The adaptive algorithm is extended to include adaption in complex multi-block domains with dynamically adapted block boundaries. The extension of the adaptor to multi-block grids is not as straightforward as the overlay approach above.

For multi-block configurations, the conditions for grid point movement lead to three primary types of outer block boundary and splice block boundary intersections. Reference 6 describes these intersections in detail. Points are constrained from movement around outer ξ, η corners in the parametric space. However, the splice boundary/outer boundary intersection of two adjacent blocks can move along the outer boundary, which allows adaptive movement of the splice boundary. One exception to movement of a splice boundary/outer boundary intersection occurs when more than one splice boundary intersects an outer boundary at the same point. This results in a concave corner in the parametric space and can cause crossover of grid lines if movement of such a point is allowed without special provision. Adaption of the block boundaries while preserving the original structure of the mesh leads to these criteria:

1. Points can move from one original transformed block to another. Therefore, the new mesh point locations must be determined, no matter which of the original transformed blocks they fall in.
2. Physical boundary conditions must be specified based on current mesh point location.
3. Crossover of grid lines near concave corners in the parametric space must be prevented.

Modifications to the adaptor are made to meet these criteria. Logic is added to determine the locations of points that move into other blocks and to insure that the boundary points acquire or retain the correct boundary condition representation. These processes are also detailed in Reference 6.

As noted previously, a center of mass scheme may result in crossover in the vicinity of concave boundaries. This does not occur in the parametric space for single block grids since no concave boundaries exist. When the method is extended to multi-block grids, concavities may be unavoidable in parametric space. The movement of the corner point, itself, can be easily limited by fixing its location at the concave corner. By locally restricting grid point movement near the concavity, crossover can be prevented. This is done by requiring mesh lines in the vicinity of the concavity to follow a von Neumann condition. The slopes of the lines at the concavity are set to be normal and change linearly to the slope defined by the adaption criteria at a set number of nodes away.

MULTI-BLOCK BOUNDARY CONDITIONS

Boundary conditions are coded generally as a set of options for defining any block boundary. For the transonic 0012 airfoils, four options are required. The block boundaries representing the far field inflow are assigned freestream density and velocity values and obtain the pressure values from the integration of the appropriate compatabity relation. The block boundaries representing the far field outflow are assigned freestream pressure values and obtain the density and velocity values from the integration of the appropriate compatibity relations. The block boundaries representing the surface are assigned entropy corrected inviscid wall conditions as described in Reference 15. The final boundary condition is the block splice boundary condition described above.

MESH QUALITY ASSESSMENT

The quality of a computational mesh is generally considered to be important to both the convergence and accuracy of a computational solution. Distorted mesh cells, e.g., those stretched and/or skewed, may cause significant errors in the developing solution. In addition, mesh cells whose sides are not aligned with local flow velocities or discontinuities can also cause inaccuracies. When adaptive mesh redistribution is used, the mesh is deformed automatically as the solution progresses. Therefore, distorted cells may occur near strong features in the course of the adaptive process. The following solution dependent mesh quality measure (SDMQM) has been developed to help assess the impact of mesh redistribution adaptation on computational solutions. It will also be useful for evaluating the results of standard mesh generation codes. The SDMQM couples the mesh cell geometry with the local solution field via the governing equations.

Consider the integral form of the conservation equations,

$$\frac{d}{dt}\int_V U dV + \oint_S \left(\vec{F}\cdot\vec{n}\right) ds = 0. \tag{19}$$

The second term of the above equation couples the solution field with the cell geometry. In the usual formulation of finite volume algorithms, this term is expressed as,

$$\oint_S \left(\vec{F}\cdot\vec{n}\right) ds = \sum_S \vec{\overline{F}}\cdot\vec{s}, \tag{20}$$

where $\vec{s}$ is the product of the surface unit normal vector and the surface area, and the overbar indicates average values over a cell surface, S. This equation is exact for planar sided cells and gives the average value of $(\nabla\cdot\vec{F})$ within the mesh cell. This is shown by using the divergence and mean-value theorems

$$\sum_S \vec{\overline{F}}\cdot\vec{s} = \int_V (\nabla\cdot\vec{F}) dV = V\left(\overline{\overline{\nabla\cdot\vec{F}}}\right)$$

where the double overbar indicates the average value over a cell volume, V.

Assume that the terms $\vec{\overline{F}}\cdot\vec{s}$ are evaluated at the center of their respective cell sides. Then, by using Taylor's series expansion we expand these terms about the cell center (cc) to obtain,

$$V\left(\overline{\overline{\nabla\cdot\vec{F}}}\right) = \sum_S \vec{\overline{F}}\cdot\vec{s} = V\left(\nabla\cdot\vec{F}\right)_{cc} + RE.$$

The resolution error, RE, is dependent on the cell geometry and the solution field and is a measure of the product of the cell volume and the difference between the average value of $(\nabla\cdot\vec{F})$ over the cell volume and the point value of $(\nabla\cdot\vec{F})$ at the cell center. The first few terms of the RE are expressed in Figure 19 for a 2-D quadrilateral cell.

We now have,

$$\mu = \frac{|RE|}{V} = \left|\left(\overline{\overline{\nabla\cdot\vec{F}}}\right) - \left(\nabla\cdot\vec{F}\right)\Big|_{cc}\right|,$$

which is a measure of the mean deviation of $(\nabla\cdot\vec{F})$ at the mesh cell center. Finally, μ is a vector quantity so we calculate $\|\mu\|_2$ which we dub the solution dependent mesh quality measure.

In the case of structured meshes, μ can be efficiently estimated by applying the transformation $(x, y, z) = (x(\xi, \eta, \zeta), y(\xi, \eta, \zeta), z(\xi, \eta, \zeta))$ to $V(\nabla \cdot \vec{F})_{cc}$ which yields

$$\mu = \frac{1}{V} \left| \sum_S \left(\vec{\overline{F}} \cdot \vec{s} \right) (E_\xi + F_\eta + G_\zeta)|_{cc} \right|. \tag{21}$$

The first term is known from the finite volume computation and the derivatives of the second term are estimated by using high-order difference approximations in computational space. In the current computations fourth-order approximations are used. The μ is not an estimate of the local solution error, but, rather, it is a measure of the local flow resolution which directly effects the solution error. If $\|\mu\|_2$ is large, then the flow within the cell is poorly resolved. When this is the case, information about the flow is lost which in turn diminishes an algorithm's ability to accurately approximate new cell interface flux values for the next integration step. On the other hand, if $\|\mu\|_2$ is small, then it is assumed that it is small at all locations within the cell indicating good flow resolution. Well resolved flow within all mesh cells leads to more accurate cell interface flux approximations and higher accuracy.

There are combinations of the cell geometry and solution field for which $\|\mu\|_2$ is small at the center of the cell but would be large if it were evaluated at some other point within the cell. However, in these circumstances the cell geometry appears to be optimal for the corresponding solution field. For example, A local linear distribution of the solution field will cause $\|\mu\|_2$ to be predicted as zero, but only if the cell being examined and those involved in the approximation of the derivatives appearing in Equation 21 are orthogonal, of equal size, and aligned with the gradient of the solution field.

In theory $\|\mu\|_2$ is a measure of the local flow resolution within a cell, but the current centered stencil used in approximating the cell center value of $(\nabla \cdot \vec{F})$ is essentially a high-order finite difference evaluation. Therefore, $\|\mu\|_2$ may be thought of as a measure of the difference between a finite volume and a finite difference formulation connected with a particular cell. When $\|\mu\|_2$ becomes zero the finite volume scheme reduces to a finite difference scheme of the order of accuracy with which $(\nabla \cdot \vec{F})$ is approximated.

A third way to view $\|\mu\|_2$ is to think of the finite volume formulation as being equivalent to second-order accurate finite difference evaluations of $(\nabla \cdot \vec{F})$ in computational space. Then $\|\mu\|_2$ is a measure of the difference between second-order and fourth-order derivative approximations and predicts a second-order truncation error associated with calculating $(\nabla \cdot \vec{F})$ within a given cell. By interpreting $\|\mu\|_2$ in this manner, conditions which cause $\|\mu\|_2$ to be small can be easily assessed. Also, it is clear that flow discontinuities (shocks, contact surfaces, slip lines) will be predicted as causing a large $\|\mu\|_2$, despite a reduction in mesh spacing, since the higher derivative terms of the aforementioned truncation error will always persist. With this in mind, we can use $\|\mu\|_2$ to examine the effect of adaptive mesh redistribution on a computed solution.

MULTI-BLOCK BOUNDARY RESULTS

Case 1: AGARD 01

AGARD 01 is a transonic 0012 airfoil. The flow conditions for this test case are $M_\infty = 0.8$ and $\alpha = 1.25°$. The important aspects of the solution are the location of the upper and lower surface shocks as well as the shape of the sonic lines. The initial grid consists of four gridblocks of 69 by 69 points which make up a C-grid as shown in Figure 1. The far field boundaries are set at 30 chord

lengths away in the x-direction and 40 chord lengths away in the y-direction. Figure 2 shows a closer view of the initial grid. Notice that the cell center locations as well as the block boundaries are shown in these figures. Figure 3 shows the final mesh after dynamic grid adaption. Figures 4 and 5 show the numerical solutions of the pressure and mach number. Notice in Figure 3 that the grid is clustered at the shock and trailing gradient and is also aligned to the gradients in presure and mach number. The sonic line is emphasized on Figure 5. The interpolated shock locations on the upper and lower surfaces of the airfoil are at 48.1% and 35.0% chord, respectively. The sonic line extends .88 chords from the center line. The pressure coefficient along the surface of the airfoil is shown in Figure 6 and is compared to computational results obtained from Reference 16. Notice that the adaptor allows for finer resolution of the shock on the upper surface. The lift coefficient is .341.

Case 2: AGARD 03

AGARD 03 is also based on the 0012 airfoil. The flow conditions are $M_\infty = 0.95$ and $\alpha = 0.°$. This case has a "fish-tail" like shock structure in which the location of the normal shock depends upon the accuracy obtained. The initial grid distribution consists of four gridblocks of 69 by 69 points which make up a C-grid similar to the grid of Case 1. The outer boundaries have a radius of 150 chords with the upstream boundary located 200 chords upstream of the airfoil. Figures 7 and 9 show views of the initial mesh and the final mesh after dynamic grid adaption. Figures 8 and 10 show closer views of the initial and adapted mesh. Figures 11 and 12 show the numerical solutions of the pressure and mach number. Notice in Figure 10 not only the grid clustering at the shocks but also the overall grid alignment to the gradients in presure and mach number. The sonic line is emphasized in Figure 13. The sonic line extends out to nearly 21 chord lengths away from the airfoil.

The distance from the trailing edge to the first subsonic Mach number value downstream is plotted as a function of nondimensional time in Figure 14. Since interpolation is not used to gain a more accurate location of the normal shock, there are non-physical jumps in the shock location in Figure 14. Notice that the normal shock location begins to converge to a location near 3.5 chords from the trailing edge and then abruptly changes speed. This change in speed is attributed to the influence of a wave which results from the weak reflection from the outer boundary of the outgoing solution startup transient. In a nonadaptive scheme, these waves are observed infrequently due to the dissipation caused by the increased mesh spacing toward the outer boundary. However, in the adaptive solution, the startup waves are sufficiently resolved such that they are still discernable as they reach the outer boundary. Although a characteristic-like outer boundary condition is used, cancellation (*i.e.* pass-through) of the wave is not perfect and a small-amplitude expansion wave is reflected toward the interior. This wave results in a small increase in effective freestream Mach number as seen by the airfoil. This increased Mach number causes the normal shock to move further aft and the upper and lower sonic regions to increase. The solution does not converge to the correct value before there is any influence from the reflected waves. This phenomenon was most prevalent when local time stepping was used. A compromise maximum time step of five times the minimum allowable was found to reduce the effect. However, as the limit is reduced (toward having a global time step), the CFL limit due to the explicit time dependent scheme results in increased compution time.

Case 6: Shock Reflection from a Double Wedge (Time Dependent)

The time dependent workshop test case No. 6 consists of an inviscid planar normal shock wave which translates over a double angled ramp. The geometry is defined by a 20° section of the ramp beginning at the Cartesian coordinate $(2, 0)$ and a 55° section passing through the coordinate $(7, 4)$.

The translating ($M = 2.16$) shock is initially perpendicular to the y-axis and positioned at $x = 1$, which was taken as the reference length ($L = 1$) for non-dimensionalization. The non-dimensional quiescent pressure and density are both set to a value of 1. The initial mesh was the same for both the static and adaptive mesh computations. This mesh consisted of 468 cells in the streamwise direction and 162 cells in the normal direction and was constructed such that all of the cells where rhomboids. The mesh was redistributed every time step for the adaptive computations. The memory requirements were small enough for this problem that both the static and adaptive computations could be run interactively on the CRAY Y-MP at NASA LaRC.

Figures 15 and 16 compare static mesh and adaptive mesh solutions of density and pressure contours, respectively. The solutions are shown for $t/t_c = 3.5$. Here t is the physical time and $t_c = U/L$, where U is the velocity of the compression region behind the normal shock. The adaptive mesh solution shows improved resolution of shock waves. Also, flow within the double compression regions behind the reflected shocks are smoother and details are more resolved in the adapted mesh computations. In addition, a region of large amplitude high frequency oscillations behind the normal shock near the triple point which occurred in the static mesh computations has been greatly reduced in the adapted mesh solution.

Figure 17 shows a series of adapted meshes for subsequent times, for $2.0 \leq t/t_c \leq 4.5$ in increments of $t/t_c = .5$. The meshes have been coarsened for clarity of display by removing every other mesh line. Also, the upper portion of the mesh has been deleted to save space. This sequence illustrates how the adaptive algorithm continuously adapts the mesh to various flow features by clustering the mesh in regions of high flow gradients and increasing alignment of the mesh with discontinuities.

Figure 18 compares the relative errors of the computed normal shock position for the static and adaptive mesh calculations. The relative position error is defined as $|X - X_E|/|X_E|)$, where X_E is the theoretical normal shock position. The position of the shock, X, was chosen to be the interpolated x/L location in the pressure transition corresponding to the mid-point of the theoretical pre- and post-shock conditions. A linear behavior of the relative position error indicates that the computed shock is moving at a constant velocity, and the magnitude of the slope indicates how close the computed shock velocity agrees with the theoretical shock velocity. When the slope is zero the two velocities are equal.

The static computation predicts a more consistent relative error throughout the computational time history because of the uniformity of the static mesh. Abrupt changes in the wave form of the relative error for the static case are due to changes in grid spacing in the direction of the shock movement. Because the initial grid was constructed of rhomboids, the spacing between the vertical grid lines is proportional to the cosine of the wedge angle. As a result, a small but sudden change in the spacing of the vertical grid lines occurs at each angle change. The reductions in mesh spacing and the required reductions in time step by the CFL condition result in a more accurate shock velocity. This fact is shown by the successive reductions in the magnitude of the slope of the static case curve.

The relative position error of the shock computed by the adaptive mesh case is larger then the static mesh case. However, by comparing the slopes of the curves, it is seen that the trends in velocity are quite similar. Therefore, the difference between the two curves is primarily due to the difference in the magnitude of the initial jump in position error. This jump is believed to be caused by the emergence of a spike in the dependent flow variables on the high-pressure side of the shock just after start-up from initial conditions. This spike apparently alters the conditions behind the shock to the point of affecting its initial propagation velocity. This spike occurs in both the static and adaptive computations. The static mesh case has a smaller initial error in velocity, and thus position, because the spike is poorly resolved and damps quickly. However, the adaptive mesh, with its ability to resolve high gradient features, resolves the spike which more severely alters the initial propagation velocity. Eventually, the spike is damped, and the shock velocity of the adapted case becomes more

in agreement with the theoretical velocity. The jumps in position error occurring in the static mesh case, when the vertical mesh line spacing is suddenly reduced, can also be explained by the above hypothesis. The static and adaptive computations show similar trends in the shock velocity history, and both yield shock positions that are accurate to within one and a half percent of the theoretical value indicating that time accuracy is maintained on the adapted mesh.

Figures 20 and 21 show exponential contour plots of $\|\mu\|_2$ for solutions of the double Mach reflection workshop test case at $t/t_c = 3.5$ obtained on a static mesh and on an adapted mesh, respectively. The mesh was redistributed every time step during the adapted mesh computation. Light contour lines correspond to well resolved flow and dark lines to poorly resolved flow as indicated by $\|\mu\|_2$. By comparing Figures 20 and 21 we can see that the deformed mesh of the adaptive computations has not introduced any significant increase in $\|\mu\|_2$. In fact, deforming the mesh to the solution field has reduced $\|\mu\|_2$ in many regions of the flow, which of course is what we try to achieve by mesh redistribution adaptation. The adaptive mesh solution has reduced wave like structures which appear in the double compression regions of the static mesh computations. Because of this, the adaptive mesh case is able to more cleanly capture flow details in the double compression region over the 55° wedge section (refer also to Figures 15 and 16). Also, the regions of high $\|\mu\|_2$ about the shocks has been reduced in extent in the adaptive mesh case. In both cases the largest values of $\|\mu\|_2$ are associated with the fastest moving discontinuities. As the static mesh solution progressed to the present time, a region of large amplitude high frequency oscillations began to emerge behind the moving normal shock near the triple point. The presence of these oscillations is clearly visible by the large concentration of high values of $\|\mu\|_2$ in this region. The adaptive mesh computation did not produce this anomaly. The only noticeable increase in $\|\mu\|_2$ of the adaptive mesh case over the static mesh case is in the region behind the normal shock and above the reflected shock. However, this increase is on the order of 1E-6. The ability of $\|\mu\|_2$ to detect discontinuites as well as low amplitude waves has encouraged ongoing research to investigate its use as an adaptive criteria.

CONCLUSIONS

The dynamic solution adaptive mesh algorithm, DSAGA3D, is successfully extended to 2-D multi-block structured grids, including adaptation of block boundaries. The multi-block solver and grid adaptor has been developed to be general, requiring only description of the grid blocks, simple splicing and boundary condition information, and criteria for adaption. All of this can be done through straightforward input without code editing.

Solutions obtained for workshop cases 1 and 2 demonstrated that the multi-block adaptive algorithm resolves the important features of the flows very well, including alignment of the mesh such that shock definition is enhanced beyond that expected from mesh spacing alone. Solutions obtained for workshop case 6 indicated that the mesh adaption procedure maintains temporal accuracy.

A mesh quality assessment criteria is proposed that provides a measure of how well the mesh resolves and aligns with the solution. This criteria has been applied to even distributed and dynamically adapted grid solutions of workshop case 6. Analysis of the results reveals the criteria to show promise for determining the resolution quality of solution features.

REFERENCES

1. Benson, R.A.,"A Dynamic Solution-Adaptive Grid Algorithm in Two and Three Dimensions," Masters' Thesis, North Carloina State University, December 1989.

2. Benson, R.A. and M^{c}Rae, D.S., "A Three-Dimensional Dynamic Solution-Adaptive Mesh Algorithm," AIAA Paper 90-1566, Seattle WA, June 1990.

3. Benson, R.A. and McRae, D.S., "A Solution-Adaptive Mesh Algorithm for Dynamic/Static Refinement of Two and Three Dimensional Grids," *Proceedings of the Third International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Barcelona Spain, June 1991.

4. Benson, R.A. and McRae, D.S., "Time-Accurate Simulations of Unsteady Flows with a Dynamic Solution-Adaptive Algorithm ," *Proceedings of the Fourth International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, Swansea UK, April 1994.

5. Ingram, C.L. and McRae, D.S., "Time-Accurate Simulation of a Self-Excited Oscillatory Supersonic External Flow with a Multi-Block Solution-Adaptive Mesh Algorithm ," *Proceedings of the 11th AIAA Computational Fluid Dynamics Conference* , Orlando Florida, July 1993.

6. Ingram, C.L.,"Extension Of A Dynamic Solution-Adaptive Mesh Algorithm And Solver To General Structured Multi-Block 2-D Configurations," Masters' Thesis, North Carloina State University, August 1995.

7. Tamura, Y. and Fujii, K., "Conservation Law for Moving and Transformed Grids," AIAA Conference Paper 93-3365-CP, Orlando Fla, July 1993.

8. Hirsch, C., Numerical Computation of Internal and External Flows, Volume 1, Wiley and Sons, 1990, pp. 445-449.

9. Liou, M.-S. and Steffen, Jr., C.J., "A New Flux Splitting Scheme," NASA TM-104404, May 1991.

10. van Albada, G.D., van Leer, B., and Roberts JR., W.W., "A Comparative Study of Computational Methods in Cosmic Gas Dynamics," Astronomy and Astrophysics, **108**, pp 76-84, 1982.

11. Scott, James N. and Niu, Yang-Yao (1993) "Comparison of Limiters in Flux-Split Algorithms for Euler Equations", AIAA paper 93-0068.

12. Eisman, P.R., Adaptive Grid Generation by Mean Value Relaxation, in: K.N. Ghia and U. Ghia, eds., Advances in Grid Generation, ASME-FED-5 1983 (ASME, New York, 1983) 29-34; also: ASME J. Fluids Engrg. 107 (1985) 477-483.

13. Connett, W.C., Agarwal, R.K., and Schwartz, A. L., "An Adaptive Grid-Generation Scheme for Flowfield Calculations," AIAA Paper 87-0199, 1987.

14. Connett, W.C., Agarwal, R.K., and Schwartz, A. L., "An Adaptive Grid-Algorithm for the Euler/Navier-Stokes Equations," AIAA Paper 88-0519, 1988.

15. Dadone, A. and Grossman, B., "Surface Boundary Conditions for the Numerical Solution of the Euler Equations," *Proceedings of the 11th AIAA Computational Fluid Dynamics Conference* , Orlando Florida, July 1993.

16. Warren, Gary Patrick, "Adaptive Grid Embedding for the Two- Dimensional Flux-Split Euler Equations," Masters' Thesis, Mississippi State University, May 1990.

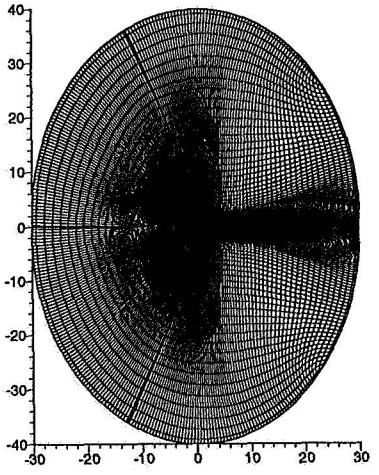


Figure 1: The Initial C-Grid: Entire Grid

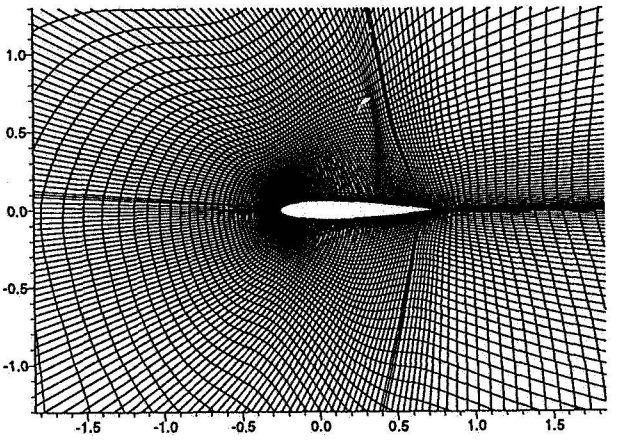


Figure 3: The Adapted C-Grid

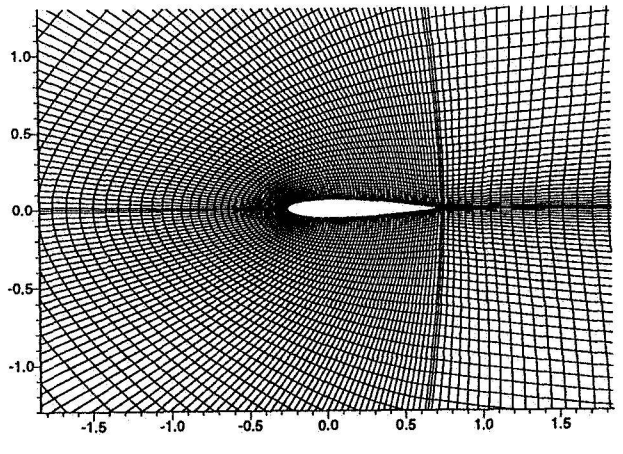


Figure 2: The Initial C-Grid: A Close View

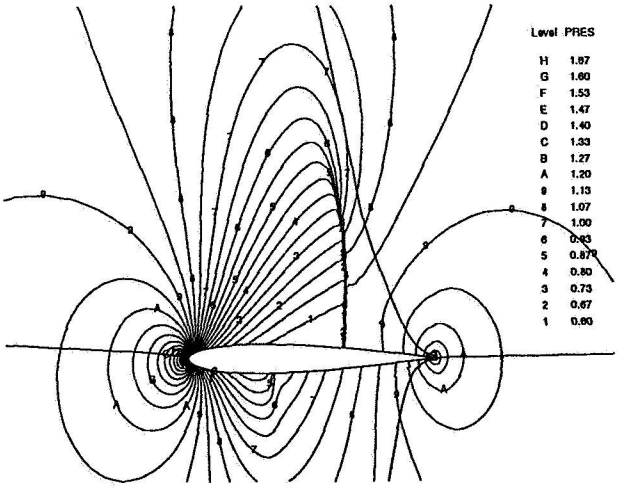


Figure 4: Pressure Contours

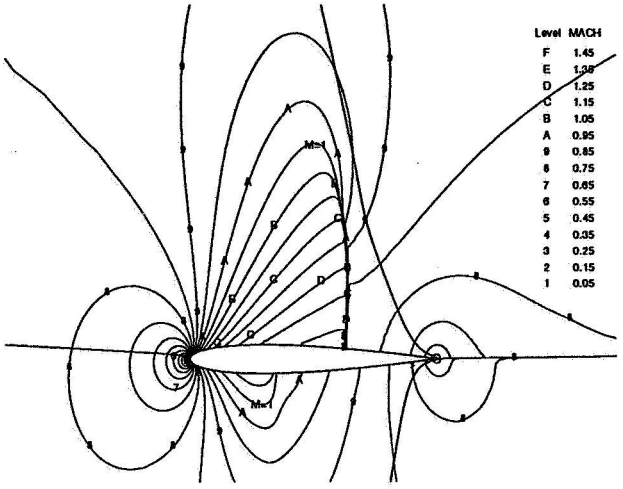


Figure 5: Mach Number Contours

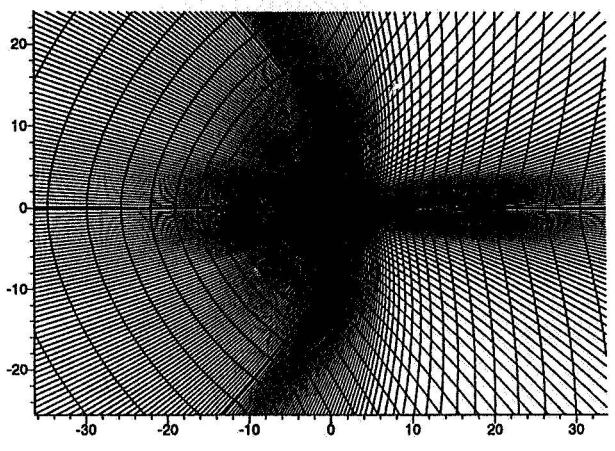


Figure 7: The Initial C-Grid

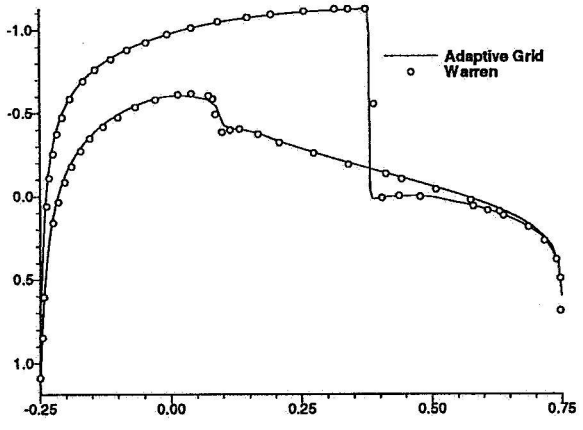


Figure 6: Pressure Coefficient Comparison

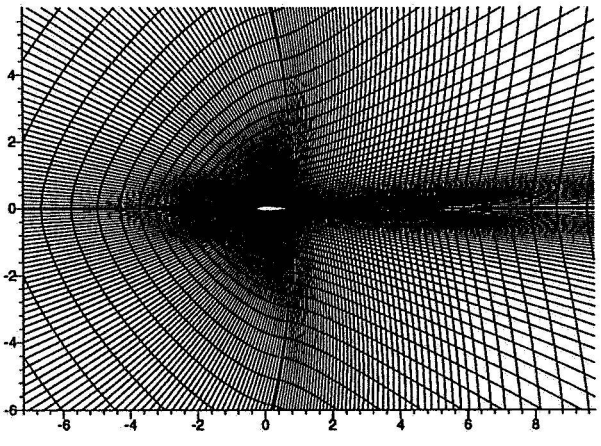


Figure 8: The Initial C-Grid: A Close View

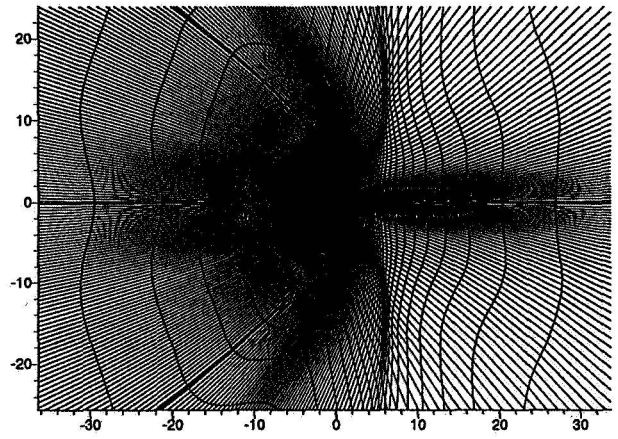


Figure 9: The Adapted C-Grid

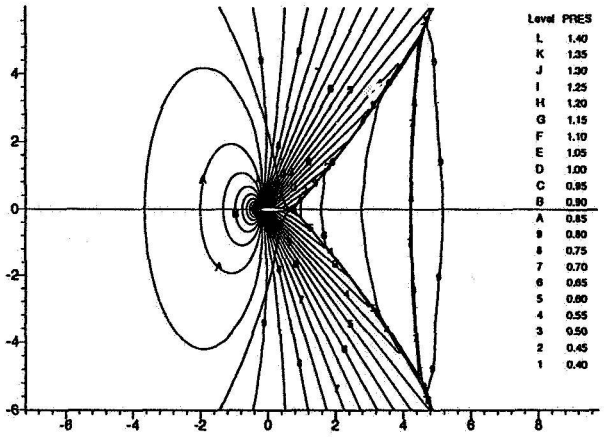


Figure 11: Pressure Contours

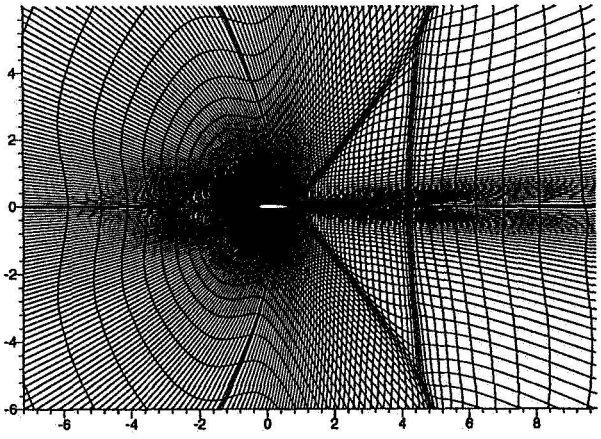


Figure 10: The Adapted C-Grid: A Close View

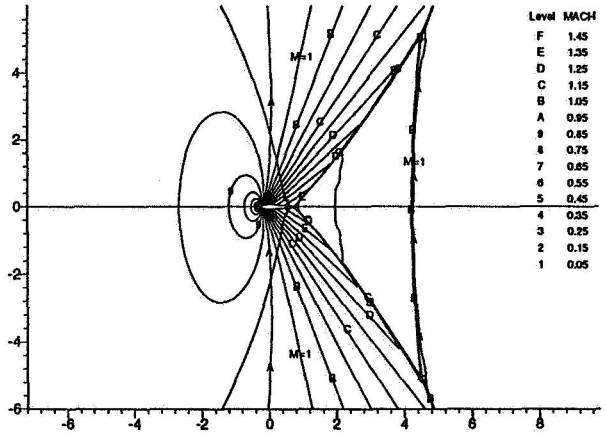


Figure 12: Mach Number Contours

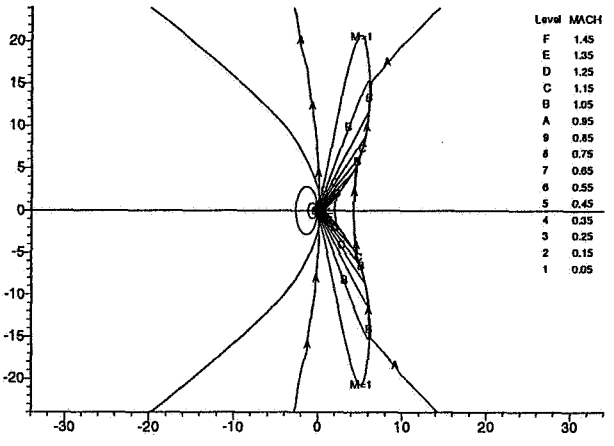


Figure 13: Mach Number Contours Featuring The Sonic Line

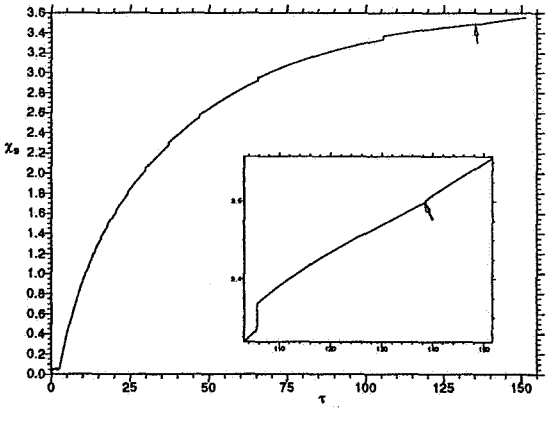


Figure 14: Normal Shock Distance From Trailing Edge

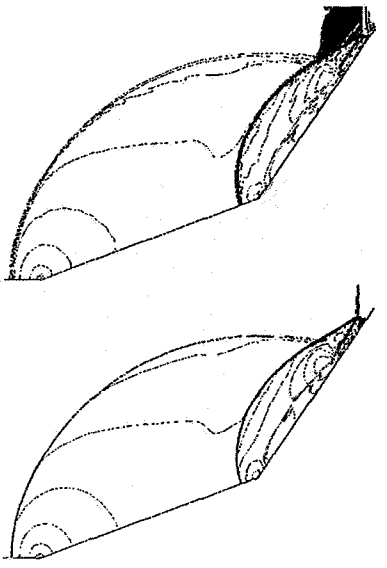


Figure 15: Density Contour Line ($\Delta\rho = 0.2$) For A Static Mesh Computation (Top) And An Adapted Mesh Computation (Bottom) For Case 6 At $t/t_c = 3.5$

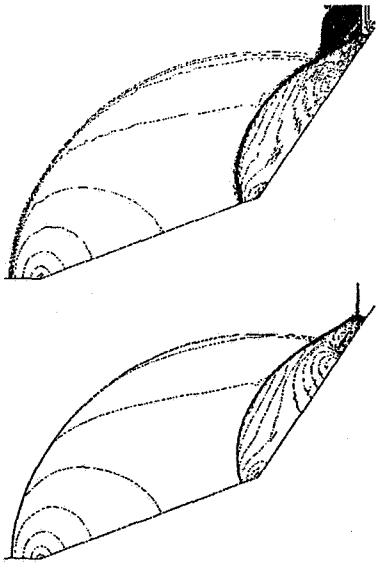


Figure 16: Pressure Contour Line ($\Delta P = 0.5$) For A Static Mesh Computation (Top) And An Adapted Mesh Computation (Bottom) For Case 6 At $t/t_c = 3.5$

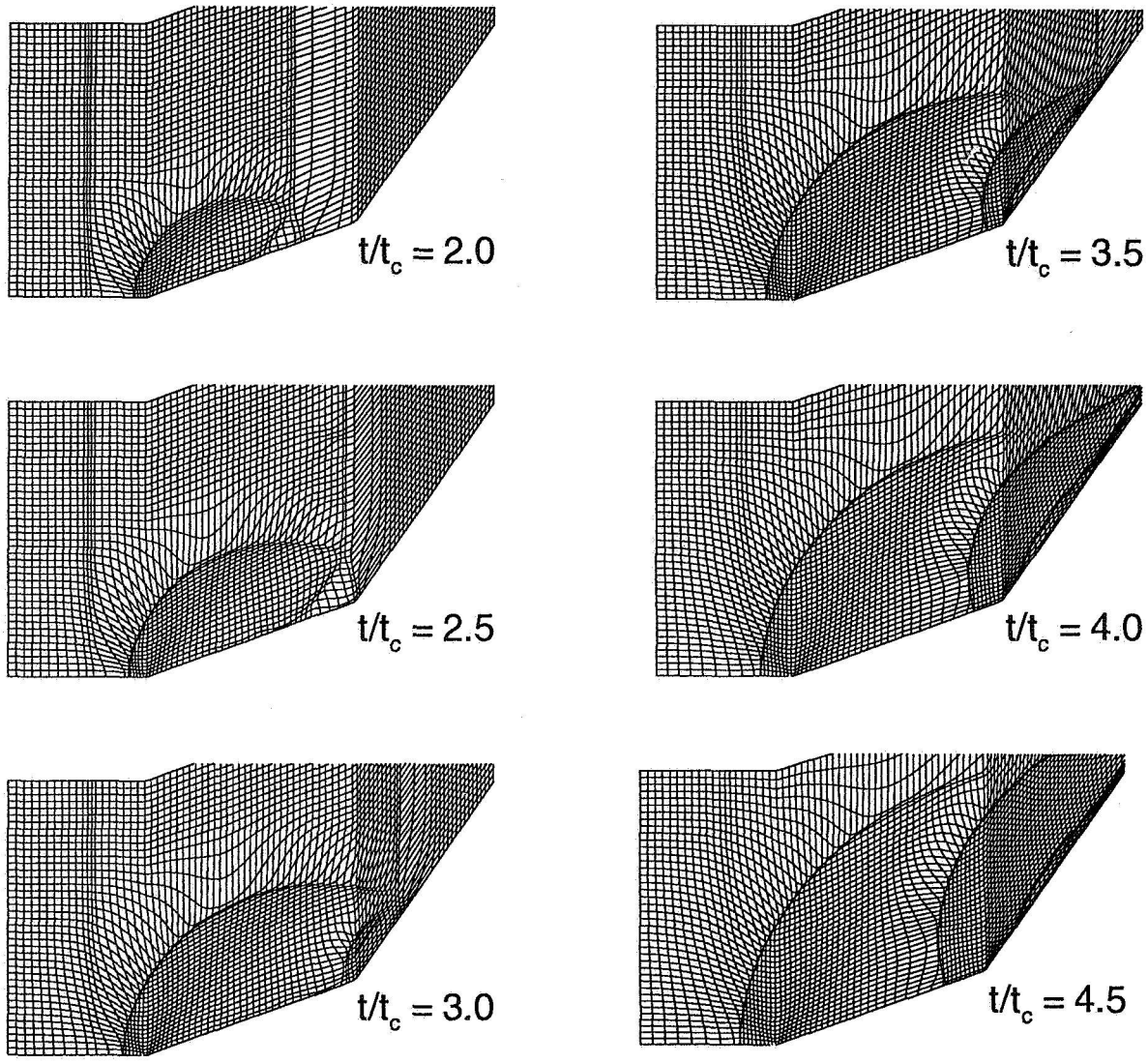


Figure 17: Adapted Mesh Series For Case 6

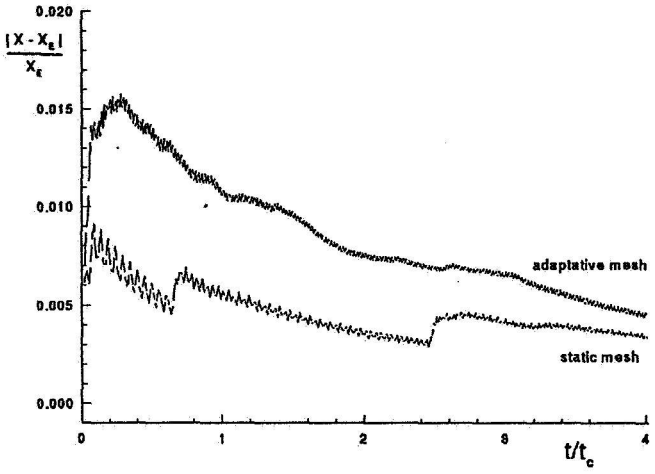


Figure 18: Shock Relative Position Error vs. Nondimensional Time For Case 6

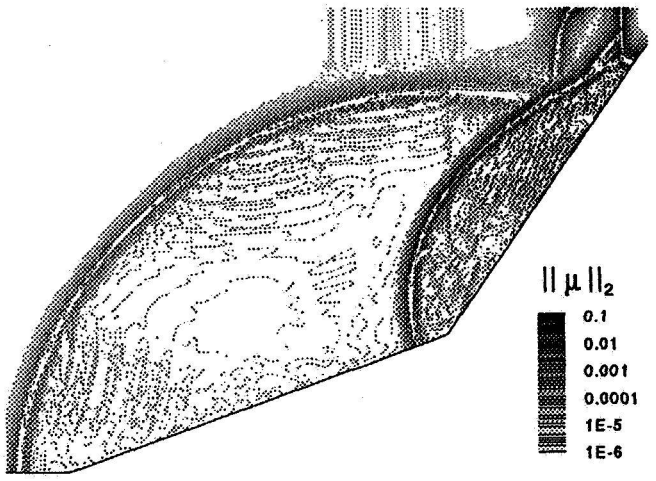
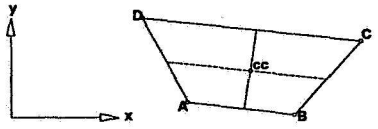


Figure 20: Solution Dependent Mesh Quality Measure (SDMQM) For The Static Mesh Computation of Case 6 at $t/t_c = 3.5$



$$\int_\Omega \nabla \cdot F dV = (f_x + g_y) V$$
$$+\frac{Y_1}{32}\left[\left(X_3^2 - X_2^2\right) f_{xx} \left(Y_3^2 - Y_2^2\right) f_{yy}\right]$$
$$+\frac{X_1}{32}\left[\left(X_3^2 - X_2^2\right) g_{xx} \left(Y_3^2 - Y_2^2\right) g_{yy}\right]$$
$$+\frac{1}{16}\left[\left(X_3 Y_3 - X_2 Y_2\right)\left(Y_1 f_{xy} - X_1 g_{xy}\right)\right]$$
$$+\frac{f_{xxx}}{384}\left(Y_2 X_3^3 - Y_3 X_2^3\right) + \frac{f_{yyy}}{384}\left(Y_2 Y_3^3 - Y_3 Y_2^3\right)$$
$$+\frac{g_{xxx}}{384}\left(X_3 X_2^3 - X_2 X_3^3\right) + \frac{g_{yyy}}{384}\left(X_3 Y_2^3 - X_2 Y_3^3\right)$$
$$+\frac{f_{xyy}}{128}\left(Y_2 X_3 Y_3^2 - Y_3 X_2 Y_2^2\right) + \frac{g_{xxy}}{128}\left(Y_2 X_3 X_2^2 - Y_3 X_2 X_3^2\right)$$
$$+\frac{f_{xxy}}{128}\left[Y_2 Y_3 \left(X_3^2 - Y_3^2\right)\right] + \frac{g_{xyy}}{128}\left[X_2 X_3 \left(Y_2^2 - X_2^2\right)\right] + HOT$$

$$X_1 = x_A - x_B + x_C - x_D \qquad Y_1 = y_A - y_B + y_C - y_D$$
$$X_2 = x_A + x_B - x_C - x_D \qquad Y_2 = y_A + y_B - y_C - y_D$$
$$X_3 = x_A - x_B - x_C + x_D \qquad Y_3 = y_A - y_B - y_C + y_D$$

Figure 19: Resolution Error For 2-D Quadrilateral Cell

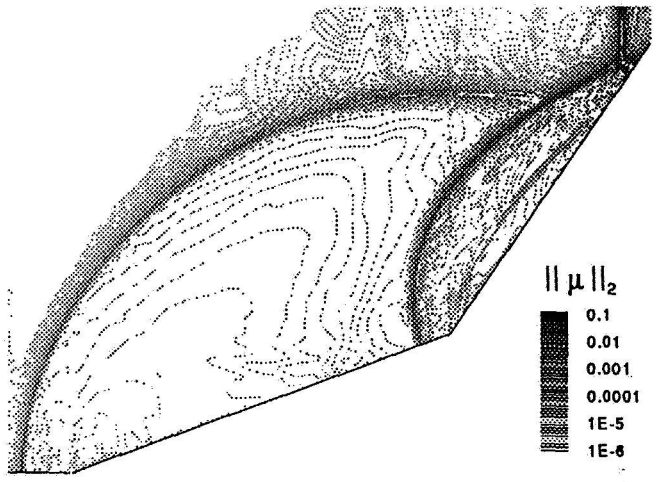


Figure 21: Solution Dependent Mesh Quality Measure (SDMQM) For The Adapted Mesh Computation of Case 6 at $t/t_c = 3.5$