NASA Technical Memorandum 106704
ICOMP–94–17; AIAA–94–2330

A.2.15

Computations of Unsteady Viscous Compressible Flows Using Adaptive Mesh Refinement in Curvilinear Body-Fitted Grid Systems

E. Steinthorsson and D. Modiano
Institute for Computational Mechanics in Propulsion
Lewis Research Center
Cleveland, Ohio

P. Colella
University of California
Berkeley, California

Prepared for the
25th Fluid Dynamics Conference
sponsored by the American Institute of Aeronautics and Astronautics
Colorado Springs, Colorado, June 20–23, 1994

National Aeronautics and
Space Administration

COMPUTATIONS OF UNSTEADY VISCOUS COMPRESSIBLE FLOWS USING ADAPTIVE MESH REFINEMENT IN CURVILINEAR BODY-FITTED GRID SYSTEMS

E. Steinthorsson* and David Modiano*
Institute for Computational Mechanics in Propulsion,
NASA Lewis Research Center, Cleveland, Ohio

and

Phillip Colella†
Department of Mechanical Engineering
University of California, Berkeley, California

Abstract

A methodology for accurate and efficient simulation of unsteady, compressible flows is presented. The cornerstones of the methodology are a special discretization of the Navier-Stokes equations on structured body-fitted grid systems and an efficient solution-adaptive mesh refinement technique for structured grids. The discretization employs an explicit multidimensional upwind scheme for the inviscid fluxes and an implicit treatment of the viscous terms. The mesh refinement technique is based on the AMR algorithm of Berger and Colella (*J. Comp. Phys., Vol. 82, pp. 64–84, 1989*). In this approach, cells on each level of refinement are organized into a small number of topologically rectangular blocks, each containing several thousand cells. The small number of blocks leads to small overhead in managing data, while their size and regular topology means that a high degree of optimization can be achieved on computers with vector processors.

Introduction

Many flows of interest to scientists and engineers are fundamentally unsteady in character. Yet, when computational methods are used to analyze such flows, the effects of unsteadiness are often neglected and the flows are simply modeled as steady. The use of Computational Fluid Dynamics (CFD) in the design of gas turbine engines is a prime example of this practice. The flow of gases through multi-stage compressors and turbines is typically modeled, stage by stage, as steady. Any unsteadiness due to passage of wakes from upstream blade rows is ignored. At best, the effects of unsteadiness is accounted for through empirical correlations (see e.g., Ref. 1). The reason behind this "pragmatic" approach is that with current computer resources and CFD algorithms, *accurate* simulations of unsteady viscous flows in turbomachinery are forbiddingly expensive. CFD algorithms that minimize the cost of such simulations while guaranteeing sufficient accuracy are needed.

For simulating steady and unsteady inviscid flows, techniques that combine high resolution shock capturing schemes with unstructured grid systems and solution adaptive mesh refinement have proven quite successful in recent years (see, *e.g.*, Ref. 2-5). Most of these techniques tessellate the flow domain using either fully unstructured grid systems (triangles and tetrahedra) or a patchwork of non-boundary-conforming Cartesian grids. The attractive feature of such grid systems is that they can be generated quickly and autonomously for geometries of arbitrary complexity, significantly reducing the man-hours needed for problem setup.

For viscous flows, these same techniques have not yet brought the same success. Several factors contribute to this lack of success. First, adequately resolving boundary layers in high Reynolds number flows has proven difficult.[6,7] Second, the non-trivial data structures required for unstructured grids do not lend themselves to high level of optimization on computers with vector processors. This can be a particularly serious drawback in simulations of viscous flows which require far greater resolution of the flow field than do inviscid flows and, hence, greater computational efficiency. The difficulty is often exacerbated by a significantly greater memory usage of computer codes employing unstructured grids compared to codes employing structured, body-fitted grid systems. Finally, performance of implicit discretizations on unstructured grids has not been satisfactory.[6] While this is not a serious limitation for

*Senior Research Associate, Member AIAA
†Professor, Member AIAA.

inviscid flows, it can be a major limitation in simulations of viscous flows due to cell-Reynolds number constraint on the allowable time step size of explicit schemes.

Compared to unstructured and non-boundary-conforming Cartesian grids, structured body-fitted grid systems offer many advantages related to accuracy and efficiency in simulations of viscous flows. Specifically, boundary layers can be resolved very efficiently with structured body-fitted grids. Also, the regularity of the grid system admits simple array data structures, facilitating optimization and vectorization of the computer codes. The regularity of the grids likewise facilitates application of implicit discretizations. Finally, generation of structured grids is becoming easier than ever before. Recent developments in techniques for generating high quality, structured, body-fitted, multiblock grid systems[8-12] indicate that same level of autonomy can be achieved in generation of structured grids for complicated geometries as in generation of unstructured grids.

To minimize cost of simulations that use structured grid systems and to enhance acuracy of the computed results, techniques need to be developed for performing local solution-adaptive refinement in structured grids. Such techniques must avoid the complicated data structures on the cell level that are used in unstructured grids, or else many of the important advantages articulated above will be lost. One grid refinement technique, generally applied to Cartesian grids, appears exceptionally suitable for structured grids. This technique is the AMR (Adaptive Mesh Refinement) algorithm of Berger and Colella.[13] The AMR algorithm takes advantage of the fact that, on a given grid system, cells that require refinement come in clusters—a whole area needs refinement rather than single cells. The AMR algorithm organizes these clusters of cells into a small number of topologically rectangular blocks, each containing a few thousand cells. Thus, simple and efficient array data structures can be used for storage of data in each grid block and a block-structured grid can be maintained on every level of refinement. The relatively small number of blocks leads to small overhead in manipulation of data during computations.

The objective of this research is to develop an efficient methodology for performing accurate simulations of unsteady viscous flows that capitalizes on the advantages of structured grid systems. Cornerstones of the new methodology include a special discretization of the compressible Navier-Stokes equations that is designed especially for accuracy and efficiency, and a solution-adaptive mesh refinement technique for structured, curvilinear grid systems that is based on the AMR algorithm of Berger and Colella.[13] The discretization employed is second order accurate in both space and time. It combines the explicit multi-dimensional upwind scheme of Colella[14] for the inviscid fluxes with an implicit scheme for the diffusion terms. The multi-dimensional upwind scheme has proven highly accurate in computations of inviscid flows,[14,5] while the implicit treatment of the diffusion terms significantly improves the robustness and computational efficiency of the scheme by eliminating the cell Reynolds number constraint on the time step size that results from explicit treatment of the viscous terms. The use of solution adaptive mesh refinement enhances both accuracy and efficiency of the scheme by providing high mesh resolution in regions where it is required, but only where it is required.

This paper focuses on only a few main aspects of this ongoing research. These aspects are the discretization of the governing equations, the extension of the AMR algorithm to structured, body fitted grid systems, and implementation of the AMR algorithm using a mixed language programming (C++ and FORTRAN). Some sample results are shown at the end of the paper.

Governing Equations and Discretization

The governing equations employed are the compressible Navier-Stokes equations (see *e.g.*, Ref. 15). The fluid is taken to be a calorically and thermally perfect gas. Transformed to general curvilinear coordinates (ξ, η) and cast in conservation law form, the governing equations can be written as

$$\frac{\partial(JU)}{\partial t} + \frac{\partial F^{\xi}}{\partial \xi} + \frac{\partial F^{\eta}}{\partial \eta} - \frac{\partial F_v^{\xi}}{\partial \xi} - \frac{\partial F_v^{\eta}}{\partial \eta} = 0 \qquad (1)$$

where $U = (\rho\ \rho u\ \rho v\ \rho e)^T$ is the vector of conserved variables: density, momentum per unit volume in the $x-$ and $y-$coordinate directions, and total energy per unit volume, respectively, and J is the transformation Jacobian. F^{ξ} and F^{η} represent the inviscid fluxes in the ξ and η directions, respectively, whereas F_v^{ξ} and F_v^{η} represent the viscous fluxes. Temperature is related to the remaining variables through

$$T = (\gamma - 1)\left(e - \frac{1}{2}(u^2 + v^2)\right) \qquad (2)$$

For details of the fluxes see *e.g.*, Ref. 16.

The governing equations are discretized using a hybrid explicit-implicit, cell-centered, finite volume discretization on curvilinear, structured grids. The discretized equations for cell (i, j) are written as follows:

$$\sigma_{ij}\frac{U_{ij}^{n+1}-U_{ij}^{n}}{\Delta t}+L_{ij}^{c}(U^{n})$$

$$-\frac{1}{2}\left(L_{ij}^{v}(U^{n})+L_{ij}^{v}(U^{n+1})\right)=0 \qquad (3)$$

where σ_{ij} is the area of cell (i,j), L_{ij}^{c} represents the discretization of the convection terms in Eq. (1) and L_{ij}^{v} represents the discretization of the viscous terms. The operator L_{ij}^{c} is derived using a straightforward extension of the multi-dimensional upwind method of Colella[14] to viscous flows. In this method, L_{ij}^{c} is written as follows:

$$L_{ij}^{c}(U^{n})=\Delta a_{i+1/2,j}F_{i+1/2,j}^{n+1/2}-\Delta a_{i-1/2,j}F_{i-1/2,j}^{n+1/2}$$

$$+\Delta a_{i,j+1/2}F_{i,j+1/2}^{n+1/2}-\Delta a_{i,j-1/2}F_{i,j-1/2}^{n+1/2} \qquad (4)$$

where $\Delta a_{i+1/2,j}$ is the area of cell face $(i+1/2,j)$ (between cells (i,j) and $(i+1,j)$), $\Delta a_{i,j+1/2}$, is the area of cell face $(i,j+1/2)$ (between cells (i,j) and $(i,j+1)$), and $F_{i+1/2,j}^{n+1/2}$ and $F_{i,j+1/2}^{n+1/2}$ are the numerical fluxes normal to those faces. To illustrate how the fluxes are evaluated, consider cell face $(i+1/2,j)$. The flux $F_{i+1/2,j}^{n+1/2}$ is evaluated using an approximate Riemann solver and can be written as:

$$F_{i+1/2,j}^{n+1/2}=F(U_{i+1/2,j,L}^{n+1/2},U_{i+1/2,j,R}^{n+1/2},\mathbf{n}_{i+1/2,j}) \qquad (5)$$

where subscripts "L" and "R" indicate "left" and "right" state for the Riemann solver, $\mathbf{n}_{i\pm1/2,j}$ is the unit normal to the cell face at $(i\pm1/2,j)$ and

$$U_{i\pm1/2,j,S}^{n+1/2}=U_{ij}^{n}\pm\frac{\Delta\xi}{2}\frac{\partial U}{\partial\xi}+\frac{\Delta t}{2}\frac{\partial U}{\partial t}$$

Using Eq. (1), we can write

$$U_{i\pm1/2,j,S}^{n+1/2}=U_{ij}^{n}\pm\frac{\Delta\xi}{2}\frac{\partial U}{\partial\xi} \qquad (6)$$

$$-\frac{\Delta t}{2J_{ij}}\left(\frac{\partial F^{\xi}}{\partial\xi}+\frac{\partial F^{\eta}}{\partial\eta}-\frac{\partial F_{v}^{\xi}}{\partial\xi}-\frac{\partial F_{v}^{\eta}}{\partial\eta}\right)$$

In the above, the subscript S stands for "L" when state at face $(i-1/2,j)$ is evaluated and "R" when state at face $(i+1/2,j)$ is evaluated. The complete formula for $U_{i\pm1/2,j,S}^{n+1/2}$ is then obtained by appropriately discretizing the derivative terms in Eq. (6). For the viscous terms, $L_{ij}^{v}(U^{n})$ (described below) is used unchanged, whereas the convection terms are treated exactly as shown in Ref. 14.

The operator L_{ij}^{v} can be written as

$$L_{ij}^{v}(U)=\begin{pmatrix}0\\ L_{ij}^{m}(\mathbf{V})\\ L_{ij}^{k}(T)+L_{ij}^{d}(\mathbf{V})\end{pmatrix} \qquad (7)$$

where $L_{ij}^{m}(\mathbf{V})$, $L_{ij}^{k}(T)$ and $L_{ij}^{d}(\mathbf{V})$ represent the viscous terms of the momentum equations, the conduction terms of the total energy equation and the viscous dissipation term of the total energy equation, respectively. $\mathbf{V}=(u,v)$ is the velocity vector. L_{ij}^{m}, L_{ij}^{k} and L_{ij}^{d} are approximated in conservative fashion using central differencing on a standard nine point stencil. In the interest of brevity, these are not detailed here. Note, for constant viscosity and conductivity, the operators L_{ij}^{m}, and L_{ij}^{k} are linear.

To facilitate advancing the solution in time, Eq. (3) is expressed as a two step scheme as follows:

$$U_{ij}^{*}=U_{ij}^{n}-\frac{\Delta t}{\sigma_{ij}}L_{ij}^{c}(U^{n})+\frac{\Delta t}{2\sigma_{ij}}L_{ij}^{v}(U^{n}) \qquad (8a)$$

$$U_{ij}^{n+1}-\frac{\Delta t}{2\sigma_{ij}}L_{ij}^{v}(U^{n+1})=U_{ij}^{*} \qquad (8b)$$

The first step, namely the evaluation of U_{ij}^{*}, is a purely explicit computation. The second step, involving the implicit part of the viscous operator, requires resolution of a system of equations. This step can be broken up into smaller equation sets as follows: Since the viscous terms do not contribute to the continuity equation, Eq. (8b) can immediately be cast as

$$\rho_{ij}^{n+1}=\rho_{ij}^{*} \qquad (9a)$$

$$\mathbf{V}_{ij}^{n+1}-\frac{\Delta t}{2\rho_{ij}^{n+1}\sigma_{ij}}L_{ij}^{m}(\mathbf{V}^{n+1})=\frac{(\rho\mathbf{V})_{ij}^{*}}{\rho_{ij}^{n+1}}=\mathbf{V}_{ij}^{*} \qquad (9b)$$

$$(\rho e)_{ij}^{n+1}-\frac{\Delta t}{2\sigma_{ij}}\left(L_{ij}^{k}(T^{n+1})+L_{ij}^{d}(\mathbf{V}^{n+1})\right)=(\rho e)_{ij}^{*} \qquad (9c)$$

If viscosity is constant, or if it can be lagged in time behind the conserved variables, Eq. (9b) effectively decouples from Eq. (9c). It can thus be inverted first. Once Eq. (9b) has been inverted and $\mathbf{V}_{ij}^{n+1}$ is known, Eq. (2) can be used to recast Eq. (9c) as

$$T_{ij}^{n+1}-(\gamma-1)\frac{\Delta t}{2\rho_{ij}^{n+1}\sigma_{ij}}L_{ij}^{k}(T^{n+1})=T_{ij}^{**} \qquad (10)$$

where

$$T_{ij}^{**} = (\gamma - 1)\left(\frac{(\rho e)_{ij}^{*}}{\rho_{ij}^{n+1}} - \frac{1}{2}\mathbf{V}_{ij}^{n+1}\cdot\mathbf{V}_{ij}^{n+1} + L_{ij}^{d}(\mathbf{V}^{n+1})\right) \tag{11}$$

Both Eq. (9b) and Eq. (10) are linear systems of equations. They can be inverted using a variety of methods. Here, a Gauss-Seidel relaxation scheme is used with red-black ordering of the unknowns. Multigrid scheme can be used very effectively to accelerate convergence of this relaxation scheme.

AMR Algorithm for Structured Curvilinear Grid Systems

One of the cornerstones of the present methodology is the use of solution-adaptive mesh refinement in structured, body-fitted grid systems. In this approach, a relatively uniform body-fitted grid system is used to provide the initial tessellation of the flow domain. Since solution-adaptive refinement is to be used, there is little need during the grid generation to cluster grid points near walls or in other regions where high resolution of the flow field is anticipated to be needed. Consequently, the structured grid system can be optimized with respect to smoothness and orthogonality alone. In general, high quality grids can be obtained this way since grid spacing is not a primary concern during the grid generation.

The AMR Algorithm

The grid refinement algorithm that best complements the use of structured grid systems is the AMR algorithm of Berger and Colella.[13] In the AMR algorithm, the computed solution exists on a sequence of mesh levels with finer and finer grids (see Fig. 1). The coarsest mesh level covers the entire physical domain of interest. Each finer mesh level is created by refining cells on the next coarser level in regions where greater resolution of the flow field is judged to be needed. A fine mesh levels typically covers only a small part of the domain and is contained in its entirety within the next coarser level. Furthermore, the boundary of the fine mesh level must lie a certain distance (measured in number of cells) from the boundary of the next coarser level, except where the boundaries of both levels coincide with the boundary of the physical domain. This is called "proper nesting" of the mesh levels. Proper nesting is required so that sufficiently accurate boundary conditions can be supplied to the fine grid at coarse–fine grid interfaces (see *e.g.,* Ref 13,17). Each mesh level is a union of topolgically rectangular blocks. Therein lies the congruence between the use of structured grid systems and the AMR algorithm. This allows the block to be the unit of operation in a computer code. Some flexible data structures are needed in the codes to allow blocks to be created and deleted. These data structures require only small overhead in the codes since the number of blocks is relatively small. Within each block, however, regular arrays can be used very effectively, contributing to the overall efficiency of the method.

In the AMR algorithm, the generation and maintenance of the hierarchy of mesh level takes place essentially as follows: Assume there exist a hierarchy of mesh levels and solutions on those levels (in the beginning, the hierarchy is simply the starting coarsest grid and the solution is the initial condition). At regular intervals, error in each cell on a given level is estimated using some suitable measure. Cells where the error exceeds a specified limit are tagged for refinement. Errors on all finer levels are also checked and cells tagged for refinement. Next, tags on the finer levels are propagated "down" to the subsequently coarser and coarser levels, until the level where the error estimation started is reached. During this process, additional cells are tagged on the coarser levels to ensure that any finer levels will be properly nested. Once the tagging of cells is completed, a special algorithm (see Ref. 13) fits a limited number of topolgically rectangular blocks around the tagged cells on each level. The mesh within these blocks is then refined by an integer ratio to create *new* fine mesh levels. The solution in the *old* fine mesh levels is then copied onto the *new* mesh levels where the two overlap. In regions where new fine grid cells have been created, the fine solution is obtained by interpolation on the next coarser grid. Note, by this procedure, fine grid cells can be removed as well as created. Note also that the mesh level where the error estimation started does not change. In this work, error is estimated using a procedure based on Richardson extrapolation (see Ref. 13).

Refinement of Curvilinear Grids

In the current methodology, refined curvilinear grid systems must be generated from an existing coarser curvilinear grid. This grid refinement must be done carefully to ensure sufficiently smooth grids on all levels of refinement. It is not sufficient to simply connect grid points on the coarse grid by straight lines and divide the lines into equal segments. This would lead to nonsmooth grids on the finer levels and to loss in accuracy. Therefore a more elaborate grid refinement scheme is needed to ensure sufficient smoothness of the fine grids.

In this study, the grid refinement is done by combining parametric cubic spline interpolation and simplified Hermite interpolation. The cubic splines, here natural cubic splines, are used to "reconstruct" the grid lines from the discrete grid points. The interpolation is done grid line by grid line and produces cubic polynomials that bridge between any two neighboring grid points on a grid line. The parameters used in constructing the splines are simply the coordinates of the computational domain. This choice automatically captures any nonuniformity of the grid spacing (*e.g.*, grid stretching) in the original grid. Thus, when new grid points are needed on the grid lines of the original grid, uniform spacing in the parameter produces a smooth grid. When new grid points are needed that lie in the interior of a coarse grid cell, the simplified Hermite interpolation is used to bridge between the four cubic polynomials that define the edges of the coarse grid cell. Full Hermite interpolation allows the enforcement of both the shape of the edges and also direction of grid lines transverse to the edges (transverse derivatives). In this usage, however, it is not necessary to enforce transverse derivatives. Hence the simplified version is used. As in the construction of the cubic splines, the parameters used in the Hermite interpolation are the computational coordinates of the coarse grid system. Uniform spacing in these parameters produces a smooth refined grid in the interior of the cell. Note, since the polynomials describing the shape of the edges of the cell were constructed using cubic splines, the overall refined grid system, obtained by refining the coarse grid cell by cell, will be smooth and at least C^1 continuous. Mathematical formulation of the grid refinement scheme is as follows:

Assume we want to refine cell (i,j). The corners of the cell are the nodes (i,j), $(i+1,j)$, $(i,j+1)$, and $(i+1,j+1)$. The physical coordinates of node (i,j) is denoted by $\mathbf{r}_{ij}$. The tangential derivative in the direction of the i-coordinate is denoted by $\mathbf{s}_{ij}$, whereas the tangential derivative in the direction of the j-coordinate is denoted by $\mathbf{t}_{ij}$. The $\mathbf{s}_{ij}$ and $\mathbf{t}_{ij}$ are obtained through the cubic spline interpolation. Let ξ be the parameter that varies along the i-grid lines and η be the parameter that varies along the j grid lines. Without loss of generality, we can assume that $(\xi,\eta) = (0,0)$ at node (i,j) and $(\xi,\eta) = (1,1)$ at node $(i+1,j+1)$. Now, the parametric formulas for the edges of cell (i,j) can be written as

$$\begin{aligned}\mathbf{r}_{i,j-1/2}(\xi) &= \mathbf{r}_{ij}h_1(\xi) + \mathbf{r}_{i+1,j}h_2(\xi) \\ &\quad + \mathbf{s}_{ij}h_3(\xi) + \mathbf{s}_{i+1,j}h_4(\xi) \qquad (12a)\end{aligned}$$

$$\begin{aligned}\mathbf{r}_{i,j+1/2}(\xi) &= \mathbf{r}_{i,j+1}h_1(\xi) + \mathbf{r}_{i+1,j+1}h_2(\xi) \\ &\quad + \mathbf{s}_{i,j+1}h_3(\xi) + \mathbf{s}_{i+1,j+1}h_4(\xi) \qquad (12b)\end{aligned}$$

$$\begin{aligned}\mathbf{r}_{i-1/2,j}(\eta) &= \mathbf{r}_{ij}h_1(\eta) + \mathbf{r}_{i,j+1}h_2(\eta) \\ &\quad + \mathbf{t}_{ij}h_3(\eta) + \mathbf{t}_{i,j+1}h_4(\eta) \qquad (12c)\end{aligned}$$

$$\begin{aligned}\mathbf{r}_{i+1/2,j}(\eta) &= \mathbf{r}_{i+1,j}h_1(\eta) + \mathbf{r}_{i+1,j+1}h_2(\eta) \\ &\quad + \mathbf{t}_{i+1,j}h_3(\eta) + \mathbf{t}_{i+1,j+1}h_4(\eta) \qquad (12d)\end{aligned}$$

where

$$h_1(u) = 1 - 3u^2 + 2u^3 \qquad (13a)$$

$$h_2(u) = 3u^2 - 2u^3 \qquad (13b)$$

$$h_3(u) = u - 2u^2 + u^3 \qquad (13c)$$

$$h_4(u) = -u^2 + u^3 \qquad (13d)$$

are the Hermite interpolants. Note,

$$h_1(0) = 1, \quad h_1(1) = h_1'(0) = h_1'(1) = 0;$$

$$h_2(1) = 1, \quad h_2(0) = h_2'(0) = h_2'(1) = 0;$$

$$h_3'(0) = 1, \quad h_3(0) = h_3(1) = h_3'(1) = 0;$$

$$h_4'(1) = 1, \quad h_4(0) = h_4(1) = h_4'(0) = 0.$$

Now simplified Hermite interpolation can be written as

$$\begin{aligned}\mathbf{r}(\xi,\eta) &= \mathbf{r}_{i,j-1/2}(\xi)h_1(\eta) + \mathbf{r}_{i,j+1/2}(\xi)h_2(\eta) \\ &\quad + \mathbf{r}_{i-1/2,j}(\eta)h_1(\xi) + \mathbf{r}_{i+1/2,j}(\eta)h_2(\xi) \qquad (14) \\ &\quad - \mathbf{r}_{ij}h_1(\xi)h_1(\eta) \quad - \mathbf{r}_{i+1,j}h_2(\xi)h_1(\eta) \\ &\quad - \mathbf{r}_{i,j+1}h_1(\xi)h_2(\eta) - \mathbf{r}_{i+1,j+1}h_2(\xi)h_2(\eta)\end{aligned}$$

The function $\mathbf{r}(\xi,\eta)$ effectively reconstructs a transformation from the computational domain to the physical domain. Using this transformation, the grid points for the fine grid are placed at uniform intervals in the parameters ξ and η.

The method described above for refining the grid system produces smooth grids as it was designed to. However, a small complication arises if the flow solver is designed to work with cells whose edges are taken to be straight lines. In this case, the fine grid cells "seen" by the flow solver do not in general match with the coarse grid cells. This is illustrated in Fig. 2. This complicates any inter-grid transfer operators used in interpolation of coarse grid data to the fine grid and in coarsening of the fine grid data, particularly if conservation mass, momentum and total energy in the operation is to be ensured. In Ref. 18, efficient transfer operators were designed for this purpose. An alternative approach used here is to let the flow solver take

the shape of the cells into account. For 2-D problems, this only requires that the area of the cells be computed using the correct shape. For the formulation used here to refine the cells (see Eq. 12-14), the area, σ, of a cell is efficiently computed as follows:

$$\sigma = \iint_C dx\,dy = \frac{1}{2}\iint_C div(\mathbf{r})\,dx\,dy$$
$$= \frac{1}{2}\oint_{\delta C} \mathbf{r}\cdot\mathbf{n}\,ds = \frac{1}{2}\sum_{l=1}^{4}\int_{\delta C_l} \mathbf{r}\cdot\mathbf{n}\,ds \qquad (15)$$

where $\mathbf{r} = (x, y)$, C denotes the cell, δC denotes the edges of the cell, $\mathbf{n}$ is the outward pointing normal to the cell. The summation is over the faces of the cell. The direction of integration around the cell is taken to be counter clockwise. This makes the normal $\mathbf{n}$ point to the right relative to the direction of integration. After a little algebra it can be shown that when the cells are defined by the cubic polynomials, the integral over a cell face is

$$\frac{1}{2}\int_{\delta C_l} \mathbf{r}\cdot\mathbf{n}\,ds = \frac{1}{2}(x_a y_b - x_b y_a)$$
$$+ f\left(\mathbf{r}_a, (d\mathbf{r}/ds)_a, \mathbf{r}_b, (d\mathbf{r}/ds)_b\right) \qquad (16)$$

where

$$\begin{aligned} & f\left(\mathbf{r}_a, (d\mathbf{r}/ds)_a, \mathbf{r}_b, (d\mathbf{r}/ds)_b\right) = \\ & -\frac{1}{60}\left(\left(\frac{dx}{ds}\right)_a\left(\frac{dy}{ds}\right)_b - \left(\frac{dx}{ds}\right)_b\left(\frac{dy}{ds}\right)_a\right) \\ & +\frac{1}{10}\left((x_b - x_a)\left(\left(\frac{dy}{ds}\right)_b - \left(\frac{dy}{ds}\right)_a\right)\right) \qquad (17) \\ & -\frac{1}{10}\left((y_b - y_a)\left(\left(\frac{dx}{ds}\right)_b - \left(\frac{dx}{ds}\right)_a\right)\right) \end{aligned}$$

In the above, subscripts "a" and "b" indicate the beginning node and end node on the edge, corresponding to the direction in which the parameter "s" in the integral varies. Note, the first term in Eq. (15) corresponds to integration along a straight line from $\mathbf{r}_a$ to $\mathbf{r}_b$. The second term, therefore, can be thought of as a correction to the integral corresponding to the deviation of the curve from a straight line curve. Changing the direction of integration in Eq. (16)(while keeping a normal pointing to the right) simply changes the sign of the results, *i.e.*,

$$\begin{aligned} & f\left(\mathbf{r}_a, (d\mathbf{r}/ds)_a, \mathbf{r}_b, (d\mathbf{r}/ds)_b\right) = \qquad (18) \\ & -f\left(\mathbf{r}_b, -(d\mathbf{r}/ds)_b, \mathbf{r}_a, -(d\mathbf{r}/ds)_a\right) \end{aligned}$$

Applying Eq. (14-16) to cell (i, j), and keeping with the notation of Eq. (12-14), we obtain

$$\begin{aligned} \sigma_{ij} &= \sigma^s_{ij} \\ &+ f\left(\mathbf{r}_{ij}, \mathbf{s}_{ij}, \mathbf{r}_{i+1,j}, \mathbf{s}_{i+1,j}\right) \\ &+ f\left(\mathbf{r}_{i+1,j}, \mathbf{t}_{i+1,j}, \mathbf{r}_{i+1,j+1}, \mathbf{t}_{i+1,j+1}\right) \qquad (19) \\ &+ f\left(\mathbf{r}_{i+1,j+1}, -\mathbf{s}_{i+1,j+1}, \mathbf{r}_{i,j+1}, -\mathbf{s}_{i,j+1}\right) \\ &+ f\left(\mathbf{r}_{i,j+1}, -\mathbf{t}_{i,j+1}, \mathbf{r}_{i,j}, -\mathbf{t}_{i,j}\right) \end{aligned}$$

where

$$\begin{aligned} \sigma^s_{ij} = & \frac{1}{2}\left(x_{i+1,j+1} - x_{i,j}\right)\left(y_{i+1,j+1} - y_{i,j}\right) \qquad (20) \\ & -\frac{1}{2}\left(x_{i+1,j} - x_{i,j+1}\right)\left(y_{i+1,j} - y_{i,j+1}\right) \end{aligned}$$

is the area of a cell whose faces are straight lines. Equation (19) can finally be written as

$$\begin{aligned} \sigma_{ij} = \sigma^s_{ij} &+ \left(f_{i+1/2,j} - f_{i-1/2,j}\right) \qquad (21) \\ &- \left(f_{i,j+1/2} - f_{i,j-1/2}\right) \end{aligned}$$

where

$$f_{i+1/2,j} = f\left(\mathbf{r}_{i+1,j}, \mathbf{t}_{i+1,j}, \mathbf{r}_{i+1,j+1}, \mathbf{t}_{i+1,j+1}\right) \qquad (22a)$$

$$f_{i,j+1/2} = f\left(\mathbf{r}_{i,j+1}, \mathbf{s}_{i,j+1}, \mathbf{r}_{i+1,j+1}, \mathbf{s}_{i+1,j+1}\right) \qquad (22b)$$

As Eq. (21) and (22) suggest, σ_{ij} for a block of cells is most efficiently computed by evaluating $f_{i+1/2,j}$ and $f_{i,j+1/2}$ for the edges of the mesh and adding the correction to σ^s_{ij}.

Note, to ensure that cell edges on the coarsest grid level always coincide with grid lines on the finer levels, the grids on all finer levels are obtained by refining the coarsest level. Consequently, the spline coefficients used to reconstruct the grid lines need only be known for the nodes on the coarsest level.

Transfer of Data Between Levels

When new fine grids are created the solution on that grid must be initialized by interpolating the data on the underlying coarse grid. Interpolation from coarse grids is also needed at interfaces between coarse and fine grids to provide boundary conditions for the fine grid. Also, when fine grids are deleted, the data from those grids must be transferred to the underlying coarse grid. In all cases the transfer of data must ensure conservation of mass, momentum and total energy in order to maintain accuracy and, for example, to maintain correct speed of moving discontinuities such as shocks. With the current definition of cells, the transfer of data from a fine grid to a coarse grid is accomplished by simply adding up the mass, momentum and total energy

in all the fine grid cells that correspond to a particular coarse grid cell. Transfer of data from coarse to fine grids is currently done using a conservative linear interpolation as shown in Ref. 18. That particular approach works well on relatively uniform grids but may need to be improved for grids with rapidly changing cell areas (Jacobians).

Implementation—Object-Oriented Mixed Language Programming

Implementation of a solution adaptive mesh refinement algorithm like the AMR algorithm described above, requires the use of programming languages that support dynamic memory allocation (and de-allocation) and user-defined data structures. The former capability is needed so that mesh levels and blocks can be created and deleted extemporaneously as the solution developes in time. The latter capability is desirable so that effective organization of the data can be done systematically and autonomously in the computer codes. Programming languages that offer both capabilities include C and C++. FORTRAN90 will also offer some of those capabilities.

While dynamic memory management and flexible data structures are needed for an effective implementation of the methodology, efficient floating point operations are also needed for fast execution of the code. The programming language that currently offers the most efficient floating point operations, particularly on vector supercomputers, is FORTRAN77 (due to highly developed compilers). This language does not, however, have the needed capabilities for memory management and data structures. Fortunately, the modularity of the AMR algorithm allows one to take advantage of the strength of the different programming languages. FORTRAN can be used very efficiently to implement all operations within a block that are related to advancing the solution in time, computing fluxes, applying physical boundary conditions, *etc.* A driver module that allocates memory for blocks, controls the time stepping, error estimation and refinement, and calls the FORTRAN routines can then be implemented in another programming language.

In this work, the AMR driver module was implemented using the C++ programming language. This language is very well suited for this purpose due to its support for object oriented programming and well defined procedure for calling FORTRAN programs. As the AMR algorithm suggests, the basic object in the implementation is the block. The corresponding *class* in the code is called "LevelBlock." A *class* in C++ is a user defined type and consists of data and a collection of functions (member functions) that operate on the data. In this case, the data in "LevelBlock" consist of arrays for the conserved variables, primitive variables and metrics, and a special data type for information about physical boundary conditions for the block. The member functions include functions that call the FORTRAN implemented flow solver. A second object in the implementation is a *class* called "MeshLevel." The data in this class includes the collection of "LevelBlocks" that make up a single mesh level, and pointers to the "MeshLevel" objects that contain the next coarser and next finer mesh levels. In this work, extensive use was made of the AMR *library* developed by Crutchfield and Welcome.[19] The AMR library is a collection of dimension independent *classes* specially designed to aid in implementation of schemes employing the AMR algorithm.

Results

When this paper is written, the methodology described in previous sections has only been tested on a number of test cases involving inviscid flow. Here, three such cases are presented. These are a subsonic flow over a NACA0012 airfoil, a transonic flow over a NACA0012 airfoil, and a supersonic flow over a blunt body. In all cases only one level of refinement is used with a refinement ratio of four.

NACA0012 airfoil at $M = 0.5$

The first test case is a subsonic flow over a NACA-0012 airfoil at zero angle of attack. The free stream Mach number (M) in this case was taken to be 0.5. To take advantage of the symmetry of the geometry, the computations were restricted to the half-plane above the airfoil. The starting (coarse) grid used in the computations contained 32 by 72 cells. The far field boundaries of the grid system were between 50 and 100 chords from the airfoil. Figure 3 shows the center region of the grid to a distance of about 4 chord lengths from the airfoil. This coarse grid has only 12 cells over the surface of the airfoil (on one side), and 48 cells after refinement. Figure 4 shows the region around the airfoil that was refined and Fig. 5 shows contours of pressure coefficient near the air foil. The refined region consisted of three blocks. A total of 9.5% of the coarse grid cells were refined. The computed solution compares well with experimental data published in Ref. 20. According to this data the minimum pressure coefficient at the surface is -0.4687, compared to computed value of -0.471, an error of about one half a percent.

NACA0012 airfoil at $M = 0.8$

The second test case is a transonic flow over a NACA0012 airfoil at zero angle of attack. The free stream Mach number was taken to be 0.8. The starting grid system was the same as described above and shown in Fig. 3. Figure 6 shows the refined grid system, and the computed pressure coefficient is shown in Fig's 7 and 8. As seen in Fig. 6, the refined grid consisted of two blocks. Fewer than 12% of the coarse grid cells were refined. corresponding to a savings by a factor of more than eight if the entire grid had been refined. The present scheme resolves the shock on the airfoil very crisply, within only two cells. As seen in Fig's 7 and 8, the location of the shock is at about 48% chord. Acchording to experimental data reported in Ref. 20, the correct shock location is 40% to 44% chord. The reason for the discrepancy is lack of resolution—experience with similar geometries indicate that with one extra level of refinement located right over the shock, the correct shock location and strength will be predicted with the current scheme.

Supersonic flow over a blunt body at $M = 5$

The last test case is a supersonic flow over a 26.56° wedge with a round leading edge. The leading edge is a cylindrical surface whose radius is 0.125 in the current scale. The free stream flow is at $M = 5$ and a detached bow shock is formed.

The grid system at the end of the computations is shown in Fig. 9. Figures 10 and 11 show contours of density and entropy, respectively, around the wedge, whereas, Fig. 12 shows plot of density on the stagnation streamline versus axial location. As Fig. 9 shows, the refined grid consisted of several topologically rectangular boxes. The refinement has mainly taken place around the shock and the stagnation region. Approximately 47.6% of the coarse grid cells were refined. As Fig. 11 to 12 show, the shock is very well captured. Also, the entropy generated at the shock is properly convected along the stream lines. In Fig. 12 it can be seen that no overshoots or undershoots in density are formed at the shock, and the density increases monotonically behind the shock. As Fig. 12 shows, the shock stands at axial distance between 0.060 and 0.063 from the leading edge. This corresponds to ratio of stand-off distance over two times the radius of curvature at the leading edge of 0.241 to 0.252. In comparison, Ref. 21 shows an experimental value of about 0.24 for flow at $M = 5$ over a cylinder.

Conclusions

In this paper, a methodology is proposed for simulating unsteady viscous and inviscid flows. The cornerstones of the methodology are the use of structured (multiblock) grid systems, solution adaptive mesh refinement based on the AMR algorithm of Berger and Colella[13], and a hybrid explicit–implicit discretization of the Navier-Stokes equations. The methodology is implemented in a computer code which is written using mixed language programming—C++ for a driver module and FORTRAN for a flow solver module. At this point only limited tests of the methodology have been performed. Overall, the results of those tests were very good. However, it was found that the simple interpolation scheme used to transfer data from coarse grids to fine grids did not work sufficiently well on meshes with rapidly changing grid spacing in two directions at the same time. Further development in this area is needed. The development and testing of the methodology is ongoing.

References

1 Sharma, O.P., Ni, R.H., Tanrikut, S., "Unsteady Flows in Turbines–Impact on Design Procedure," AGARD-LS-195, pp. 5-1 — 5-27, May to June, 1994.

2 Jameson, A., Baker, T.J., and Weatherill, N.P., "Calculation of Inviscid Transonic Flow over a Complete Aircraft," AIAA-86-0102.

3 Mavriplis, D.J., "Unstructured Mesh Algorithms for Aerodynamic Calculations," ICASE Report No. 92-35.

4 De Zeeuw, D. and Powell, K.G., "An Adaptively-Refined Cartesian Mesh Solver for the Euler Equations," AIAA-91-1542-CP, Proceedings, 10th AIAA Computational Fluid Dynamics Conference, Honolulu, 1991

5 Pember, R.B., Bell, J.B., Colella, P., Crutchfield, W.Y., and Welcome, M.L., Adaptive Cartesian Grid Methods for Representing Geometry in Inviscid Compressible Flow," AIAA-93-3385, Proceedings, 11th AIAA Computational Fluid Dynamics Conference, 1993.

6 Thompson, J.F., and Weatherill, N.P., "Aspects of Numerical Grid Generation: Current Science and Art," AIAA-93-3539-CP, Proceedings, AIAA Applied Aerodynamics Conference, 1993.

7 Frink, N.T., "Recent Progress Toward a Three-Dimensional Unstructured Navier-Stokes Flow Solver," AIAA-94-0061.

8 Allwright, S. E., "Techniques in Multiblock Domain Decomposition and Surface Grid Generation," in *Numerical Grid Generation for Computational Fluid Mechanics '88*; Sengupta, S., Häuser, J., Eiseman, P. R., and Thompson, J. F., (ed), pp. 559-568, Pineridge Press, 1988.

9 Amdahl, D. J., "Interactive Multi-Block Grid Generation," in *Numerical Grid Generation for Computational Fluid Mechanics '88*; Sengupta, S., Häuser, J., Eiseman, P. R., and Thompson, J. F., (ed), pp. 579-578, Pineridge Press, 1988.

10 Stewart, M., "A General Decomposition Algorithm Applied to Multi-Element Airfoil Grids," AIAA 90-1606, 1990.

11 "GRIDPRO,™ /AZ3000, Users Guide and Reference Manual", Program Development Corporation, White Plains, NY, 1993.

12 Schönfeld, T. and Weinerfelt, P., "Automatic Generation of Quadrilateral Multi-Block Grids by the Advancing Front Technique," Proceedings of the *Third International Conference on Numerical Grid Generation*, pp. 743-754, S.-Arcilla, A., Häuser, J., Eisemann, P.R., and Thompson, J.F. (Eds.), Elsevier Science Publishers, North Holland, June 1991.

13 Berger, M.J. and Colella, P., "Local Adaptive Mesh Refinement for Shock Hydrodynamics," *J. of Comp. Physics,* Vol. 82, pp. 64-84, 1989.

14 Colella, P. "Multidimensional Upwind Methods for Hyperbolic Conservation Laws," *J. of Comp. Physics,* Vol. 87, pp. 171-200, 1990.

15 Schlichting, H., "Boundary-Layer Theory," McGraw-Hill, New York, 1979.

16 Anderson, D.A., Tannehil, J.C., Pletcher, R.H., "Computational Fluid Mechanics and Heat Transfer," Hemisphere Pub. Co., Washington, 1984.

17 Berger, M.J., "On Conservation at Grid Interfaces," *SIAM J. Numer. Anal.,* Vol. 24, No. 5, pp. 967-984, October, 1987.

18 Bell, J., Colella, P., Trangenstein, J., Welcome, M., "Adaptive Mesh Refinement on Moving Quadrilateral Grids," AIAA-89-1979-CP, Proceedings, AIAA 9th CFD Conference, 1989.

19 Crutchfield, W.Y. and Welcome, M.L., "Object Oriented Implementation of Adaptive Mesh Refinement Algorithm,", *Scientific Programming,* Vol. 2, number 4, winter 1993.

20 "Experimental Data Base for Computer Program Assessment," AGARD-AR-138, Report of the Fluid Dynamics Panel Working Group 04, May 1979

21 Liepmann, H.W., Roshko, A., "Elements of Gasdynamics," J.Wiley and Sons, New York, 1957, p. 105.

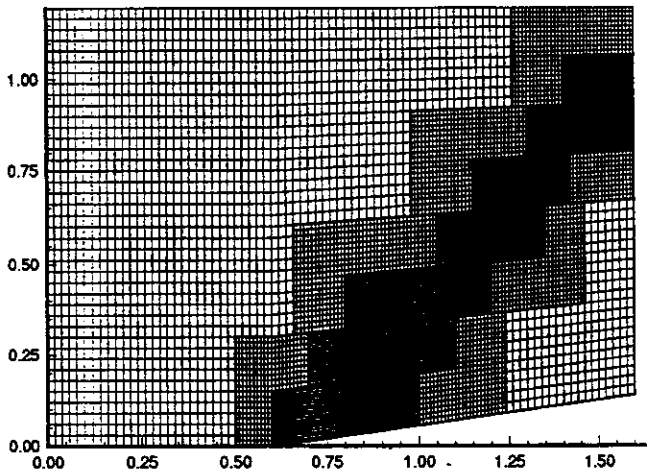

Figure 1. A grid system on three levels—a coarse grid covering the entire physical domain and two properly nested fine grid levels, each consisting of several topologically rectangular blocks.

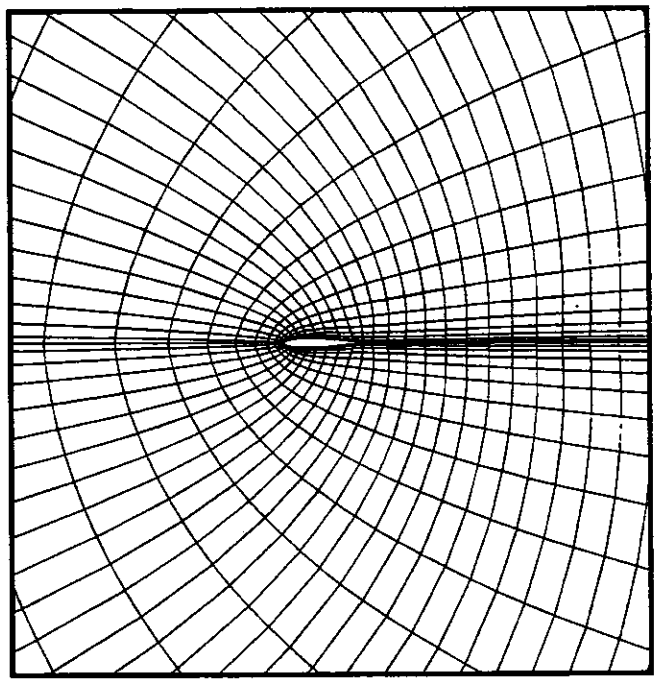

Figure 3. Initial (coarse) grid system for NACA0012 airfoil—computations were done on only the upper half of the symmetric grid.

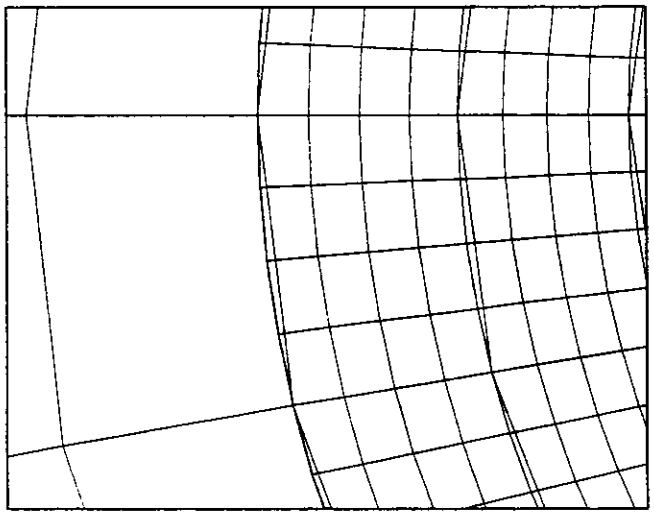

Figure 2. A curvilinear coarse grid system and a refined grid—cells are defined by straight lines between nodes, resulting in fine grid cells that do not match with the "parent" cells on the coarse grid.

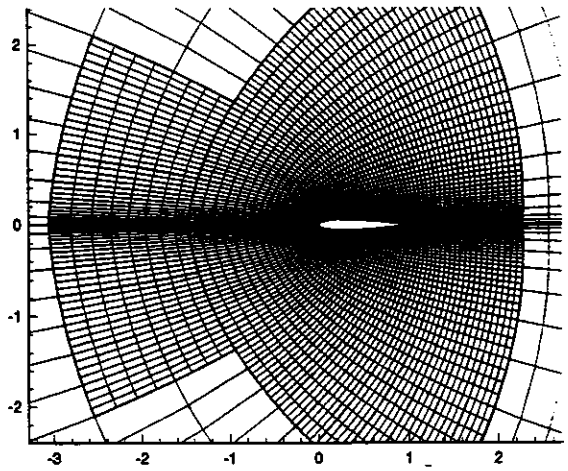

Figure 4. Refined grid around a NACA0012 airfoil—flow at $M = 0.5$ and zero angle of attack.

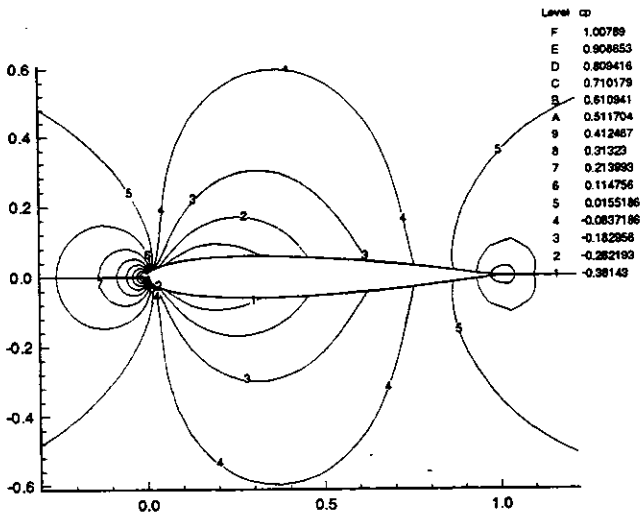

Figure 5. Flow over a NACA0012 airfoil at $M = 0.5$ and zero angle of attack—contours of pressure coefficient.

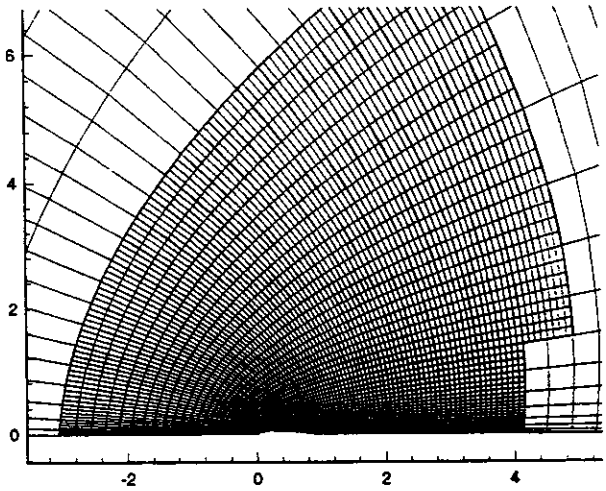

Figure 6. Refined grid around a NACA0012 airfoil—flow at $M = 0.8$ and zero angle of attack.

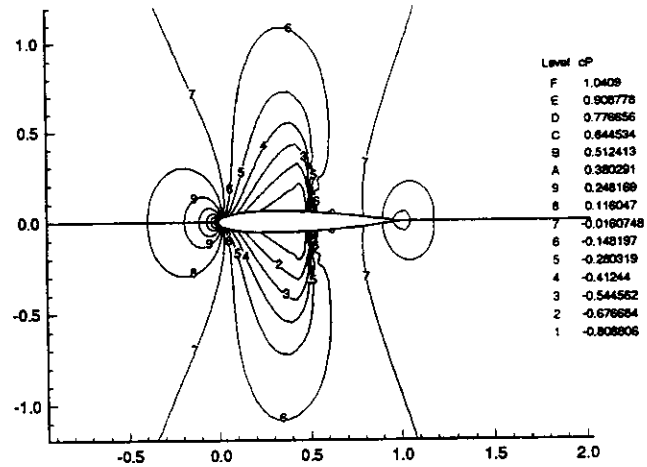

Figure 7. Flow over a NACA0012 airfoil at $M = 0.8$ and zero angle of attack—contours of pressure coefficient.

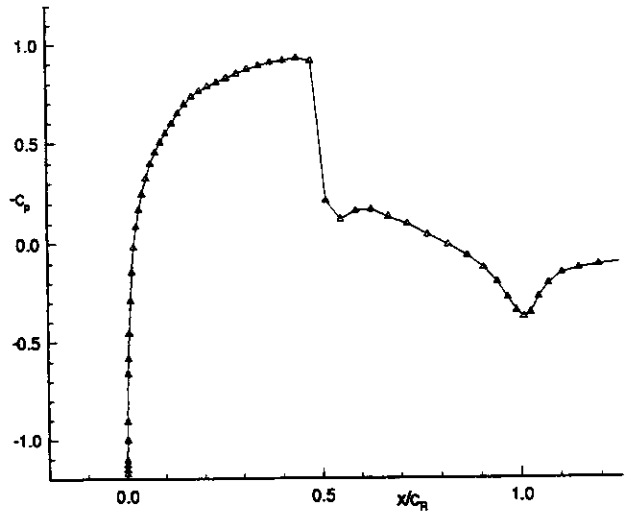

Figure 8. Flow over a NACA0012 airfoil at $M = 0.8$ and zero angle of attack—plot of pressure coefficeint versus chord.

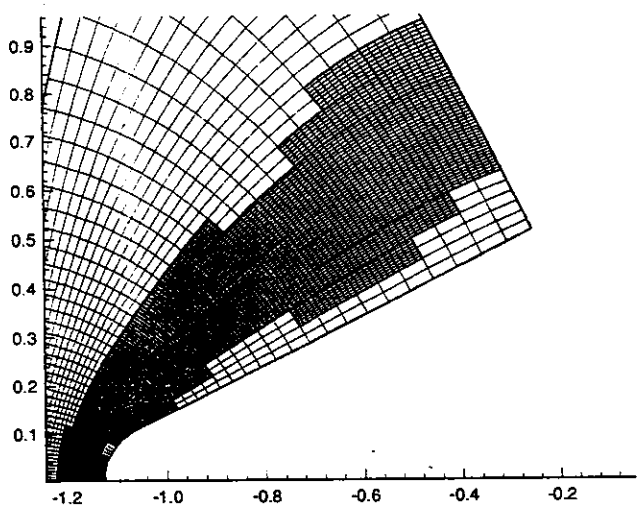

Figure 9. Grid system for a 26.56° wedge with a blunt leading edge—coarse grid is 32 × 48; 47.6% of the coarse grid cells are refined.

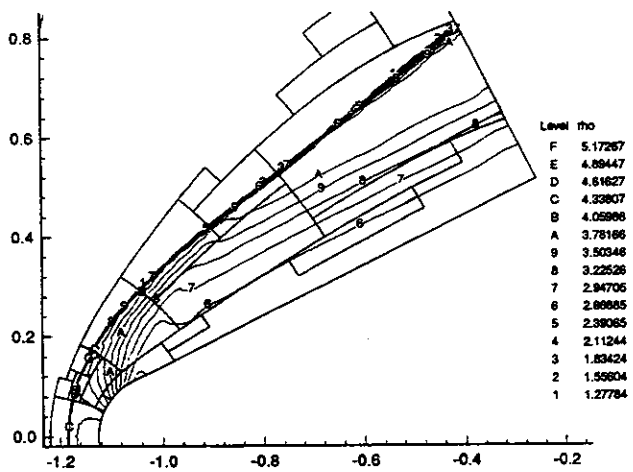

Figure 10. $M = 5$ flow over a 26.56° wedge with a blunt leading edge—contours of density.

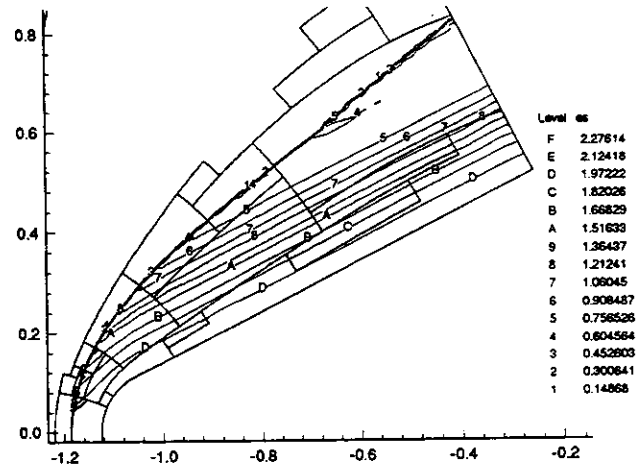

Figure 11. $M = 5$ flow over a 26.56° wedge with a blunt leading edge—contours of entropy.

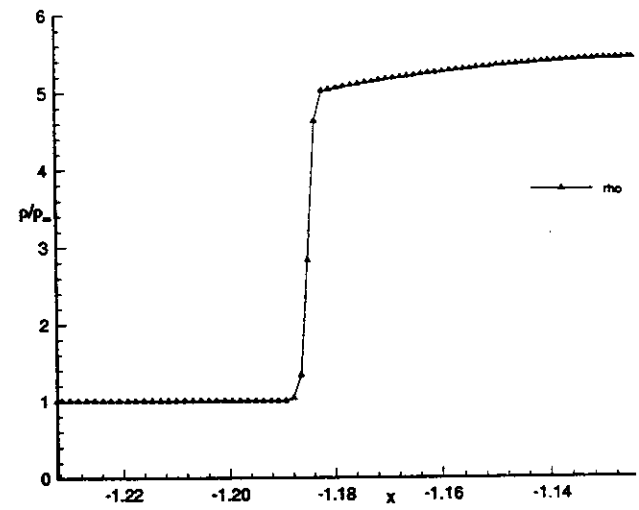

Figure 12. $M = 5$ flow over a 26.56° wedge with a blunt leading edge—plot of normalised density on the stagnation stream line versus axial location (leading edge is at x=-1.125; radius of curvature of leading edge is 0.125).