A.2.16
AIAA-95-1727-CP

AN ADAPTIVE SEMI-IMPLICIT SCHEME FOR SIMULATIONS OF UNSTEADY VISCOUS COMPRESSIBLE FLOWS

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

William Y. Crutchfield† and John B. Bell‡
Center for Computational Sciences and Engineering
Lawrence Livermore National Laboratory, Livermore, California

and

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

Abstract

A numerical scheme for simulation of unsteady, viscous, compressible flows is considered. The scheme employs an explicit discretization of the inviscid terms of the Navier-Stokes equations and an implicit discretization of the viscous terms. The discretization is second order accurate in both space and time. Under appropriate assumptions, the implicit system of equations can be decoupled into two linear systems of reduced rank. These are solved efficiently using a Gauss-Seidel method with multigrid convergence acceleration. When coupled with a solution-adaptive mesh refinement technique, the hybrid explicit-implicit scheme provides an effective methodology for accurate simulations of unsteady viscous flows. The methodology is demonstrated for both body-fitted structured grids and for rectangular (Cartesian) grids.

Introduction

For simulation of unsteady inviscid compressible flows, techniques based on explicit discretizations of the governing equations of fluid mechanics have proven quite useful. For viscous flows however, purely explicit schemes are useful only for a very limited class of low Reynolds number flows. The reason for this is well known, namely, the severe restriction on the allowable time step size due to small cell Reynolds numbers in boundary layer regions of high Reynolds number flows. For instance, for the explicit MacCormack scheme[1] the allowable time step for stability is given by the following empirical formula[2,3]

$$\frac{\Delta t}{\Delta t_{inv}} < \frac{1}{1 + 2/Re_{\Delta}}$$

where Δt_{inv} is the limiting time step for inviscid flow and Re_{Δ} is the minimum cell-Reynolds number. In boundary layers of high Reynolds number flows the Re_{Δ} can be several orders of magnitude less than one, due to small grid spacing needed to resolve the boundary layer and due to the small velocity normal to the wall, exactly where the grid spacing is smallest. This small cell Reynolds number leads to a time step orders of magnitude smaller than Δt_{inv} and a corresponding increase in cost of computations.

To overcome the difficulty associated with small cell Reynolds numbers, we have in a recent paper[4] proposed a methodology for accurate simulations of unsteady, viscous and inviscid compressible flows. The methodology employs a second order accurate discretization of the Navier-Stokes equations that is explicit for the inviscid terms but implicit for the viscous terms. The methodology also employs solution adaptive mesh refinement based on the AMR algorithm of Berger and Colella[5] to enhance both accuracy and efficiency of the scheme. The focus of our first paper was on the use of the AMR algorithm in structured grids. In this paper, which can be considered a continuation of the first, the focus will be on the implicit discretization of the viscous terms and on the use of the AMR algorithm in simulations of viscous flows. Note, the discretization that we propose can be used with or without adaptive mesh refinement, as required by the problem to be analyzed. However, with adaptive refinement, it becomes possible to analyze efficiently and accurately

*Senior Research Associate, Member AIAA
†Staff Scientist.
‡Director.
§Professor, Member AIAA.

Copyright ©1993 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for governmental purposes. All other rights are reserved by the copyright owner.

fluid flow problems that involve vastly different length scales, such as shock-boundary layer interactions. This is demonstrated in this paper.

The remainder of this paper is organized as follows: Following this introduction, the discretization used for the governing equations will be reproduced for completeness of the paper. Afterwards, the algorithm used to solve the systems of equations will be described and discussed. Then some special considerations related to the integration of the numerical scheme into a code employing AMR will be discussed. Finally, results will be shown for selected steady and unsteady viscous flows.

Governing Equations and Discretization

The governing equations employed are the compressible Navier-Stokes equations (see *e.g.*, Ref. 6). 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

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

For details of the fluxes see *e.g.*, Ref. 3.

The governing equations are discretized using a semi-implicit, cell-centered, finite volume discretization that is second order accurate in both space and time. In this discretization the convection terms are treated explicitly, whereas the viscous terms are integrated implicitly using the trapezoidal rule. The discretized equations for cell (i, j) are written as follows:

$$\sigma_{ij}\frac{U_{ij}^{n+1} - U_{ij}^{n}}{\Delta t} + L_{ij}^{c}(U^n)$$
$$-\frac{1}{2}\big(L_{ij}^{v}(U^n) + L_{ij}^{v}(U^{n+1})\big) = 0 \qquad (3)$$

where σ_{ij} is the area of cell (i, j), L_{ij}^{c} represents the discretized convection terms in Eq. (1) and L_{ij}^{v} represents the discretized viscous terms.

The operator L_{ij}^{c} is derived using a straightforward extension to viscous flows of the multi-dimensional upwind method of Colella[7]. 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 $(i \pm 1/2, j)$ 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. 8.

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^m_{ij}, L^k_{ij} and L^d_{ij} 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^m_{ij} and L^k_{ij} are linear.

Solution Algorithm for the System of Equations

When the computed solution is advanced in time using semi-implicit discretization, a system of equations needs to be solved in every time step. To facilitate advancing the solution in time and solving the system of equations, Eq. (3) is expressed as a two step scheme as follows:

$$U^*_{ij} = U^n_{ij} - \frac{\Delta t}{\sigma_{ij}} L^c_{ij}(U^n) + \frac{\Delta t}{2\sigma_{ij}} L^v_{ij}(U^n) \qquad (8a)$$

$$U^{n+1}_{ij} - \frac{\Delta t}{2\sigma_{ij}} L^v_{ij}(U^{n+1}) = U^*_{ij} \qquad (8b)$$

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

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

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

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

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 $\mathbf{V}^{n+1}_{ij}$ is known, Eq. (2) can be used to recast Eq. (9c) as

$$T^{n+1}_{ij} - (\gamma - 1)\frac{\Delta t}{2\rho^{n+1}_{ij}\sigma_{ij}} L^k_{ij}(T^{n+1}) = T^{**}_{ij} \qquad (10)$$

where

$$T^{**}_{ij} = (\gamma - 1)\left(\frac{(\rho e)^*_{ij}}{\rho^{n+1}_{ij}} - \frac{1}{2}\mathbf{V}^{n+1}_{ij} \cdot \mathbf{V}^{n+1}_{ij} + L^d_{ij}(\mathbf{V}^{n+1})\right) \qquad (11)$$

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. This has proven to be an effective relaxation scheme for high Reynolds number flows, and converges in only a few iterations. For low Reynolds number flows, with stronger coupling between unknowns, such a local relaxation scheme may not be sufficiently efficient and may require a multigrid convergence acceleration.

Note, it is not always necessary to solve Eq. (9b) and (10) exactly. All that is needed is to relax until the error is small enough to ensure the stability of the overall scheme. However, it is necessary to ensure conservation at every time step, especially for flows involving strong shocks. Thus, the algorithm used to advance the solution by one time step is as follows:

(1) Compute U^*_{ij} using Eq. (8a). This gives the final value of density, *i. e.*, $\rho^{n+1}_{ij} = \rho^*_{ij}$

(2) Compute intermediate values of velocity, $\mathbf{V}^{\overline{n+1}}_{ij}$, by relaxing Eq. (9b) until sufficient convergence is achieved.

(3) Compute U^{**}_{ij} as

$$U^{**}_{ij} = U^*_{ij} + \frac{\Delta t}{2\sigma_{ij}} \begin{pmatrix} 0 \\ L^m_{ij}(\mathbf{V}^{\overline{n+1}}) \\ L^d_{ij}(\mathbf{V}^{\overline{n+1}}) \end{pmatrix} \qquad (12)$$

This gives final value of velocity, *i. e.*, $\mathbf{V}^{n+1}_{ij} = \mathbf{V}^{**}_{ij}$

(4) Compute intermediate values of temperature, $T^{\overline{n+1}}_{ij}$, by relaxing Eq. (10) until sufficient convergence is achieved (note, here T^{**} can be computed directly from U^{**}, using Eq. (2)).

(5) Compute final value of total energy as

$$(\rho e)^{n+1}_{ij} = (\rho e)^{**}_{ij} + \frac{\Delta t}{2\sigma_{ij}} L^k_{ij}(T^{\overline{n+1}}) \qquad (13)$$

This completes the computation of U^{n+1}_{ij}.

Adaptive Mesh Refinement

In this work, solution adaptive mesh refinement (AMR) is used to enhance both accuracy of the computed solution and the efficiency of the algorithm. The use of solution adaptive mesh refinement enhances accuracy by ensuring sufficient resolution of small scale features in the flow field, including shocks and boundary layers, even when it is not known beforehand where such features may occur and when they are moving or

changing in time. Simultaneously, the efficiency of the solution algorithm is enhanced as the flow solver is enabled to focus its efforts in regions where it is most needed, using appropriately coarser grids in other regions.

The mesh refinement algorithm used here is based on the AMR algorithm of Berger and Colella.[5] In this algorithm, the solution exists on several *levels* of finer and finer meshes. Each mesh level, excluding the coarsest which covers the entire domain, is embedded within the next coarser level (see *e.g.,* Fig. 1) and typically covers only a small fraction of the domain. The finer levels are created by refining cells on the coarser levels by an integer refinement ratio, r, where high resolution is required. On every mesh level, the cells are organized into a small number of topologically rectangular *blocks*, each containing a few thousand points. The advantage of this organization is that simple and efficient array data structures can be used for storage of the data in each grid block. The relatively small number of these data structures leads to small overhead in manipulation of data during computations. In addition, the regular arrays lend themselves to a high level of optimization on vector computers (*e.g.,* Cray computers).

For unsteady problems, refinement is done in time as well as in space. Thus—to synchronize in time the solutions on different levels—whenever the solution on one mesh level is advanced in time by Δt, the solution on the next finer level is advanced r times by a time step $\Delta t/r$. This leads to a recursive time stepping scheme for the mesh hierarchy. At interfaces between coarse and fine grids, boundary conditions are needed for the fine grid. These boundary conditions are simply obtained by interpolation in space and time on the coarser grids. This interpolation produces no loss in accuracy provided care is taken in the creation of the mesh hierarchy to ensure that such interpolation takes place where the flow field is sufficiently resolved by the coarse grid. To achieve this in a general case, the boundary of the coarse mesh level must lie a certain distance (measured in number of cells) from the boundary of the embedded fine mesh 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 and is required so that sufficiently accurate boundary conditions can be supplied to the fine grid (see *e.g.,* Ref 5, 7). In addition to proper nesting, the fine mesh level itself is made to extend slightly beyond the region where high resolution is needed so that small scale feature in the flow field do not lie immediately adjacent to coarse-fine grid interfaces where the interpolation takes place. This is done by refining a layer of coarse grid cells surrounding the region to be refined and including with the fine mesh level. This is called *buffering.* Finally, to ensure conservation of mass, momentum and total energy on the mesh hierarchy, a special *refluxing operation* is needed at the coarse-fine interfaces after the completion of the r time steps on the fine grid, and before the next time step on the coarse grid. This refluxing operation imposes on the underlying coarse grid cells the aggregate of the fluxes "seen" by the adjacent fine grid cells and ensures convergence of the solution to a weak solution of the governing equations as the mesh is refined.[8] At the end of the r time steps on the fine grid, the solution on the fine grid must be *restricted* to the next coarser grid to ensure accuracy of the coarse grid solution.

In the AMR algorithm, the generation and maintenance of the hierarchy of mesh levels takes place essentially as follows: Assume there exist a hierarchy of mesh levels and solutions on those levels (in the beginning, the hierarchy can simply be the starting coarsest grid with the initial conditions as the starting solution). Every few time steps, the error in each cell on a given level is estimated using some suitable measure (see *e.g.* Ref.5, 9). Cells where the error exceeds a specified limit are tagged for refinement and buffer cells are added. 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 (*i.e.,* if a cell is tagged for refinement, the corresponding cells on the coarser levels are also tagged), 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. 5) fits a small number of topolgically rectangular blocks around the tagged cells on each level. The mesh within these blocks is then refined to create *new* fine mesh levels. The solution in the *old* mesh levels is then copied onto the *new* mesh levels of same resolution 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. 5).

Refinement of Curvilinear Grids

Special care must be taken when the adaptive mesh refinement is applied to body-fitted, curvilinear

grid systems. In this case, the refined grids must be generated from an existing coarser curvilinear grid. This grid refinement must ensure sufficiently smooth grids on all levels of refinement. In this work, grid refinement for curvilinear grids is done by combining parametric cubic spline interpolation and simplified Hermite interpolation. The cubic splines, here natural cubic splines, are used to "reconstruct" 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. For further details of the AMR algorithm used here for curvilinear structured grids, see Ref. 4. For other work on adaptive mesh refinement in structured grids see *e.g.* Ref. 10-13.

Implementation—Object-Oriented Mixed Language Programming

The methodology described in this paper has been implemented in two computer codes, one designed for structured body-fitted grid systems, the other for uniform Cartesian grids. Both codes are designed to compute two-dimensional flows. The codes were written using mixed language programming, *i.e.*, a driver module for the semi-implicit adaptive algorithm was written in the C++ programming language while all routines performing floating point intensive parts of the algorithm (e.g., the approximate Riemann solver) were written in FORTRAN. This implementation is possible due to the modularity of the AMR algorithm and allows one to take advantage of the strengths of the different programming languages. Here, the strengths of C++, including object-oriented capability, flexible data structures and dynamic memory allocation, make that language very effective for the implementation of a driver for the AMR algorithm while the extensive optimization by FORTRAN compilers of floating point operations on array data structures makes FORTRAN the language of choice for most of the compute-intensive parts. Both codes make extensive use of the AMR *library* developed by Crutchfield and Welcome.[14] The AMR library is written in C++ and is a collection of space-dimension independent *classes* specially designed to aid in implementation of schemes employing the AMR algorithm.

Results

Results obtained using the semi-implicit discretization of the Navier-Stokes equations and AMR are presented for two test problems. The first problem is a super sonic flow over a 26.56° wedge with a blunt leading edge. The second problem is a shock reflection of an inclined surface.

Supersonic flow over a blunt body at $M = 5$—unadapted mesh

To demonstrate the applicability of the algorithm in computing viscous flows on traditional unadapted, body-fitted structured grids with dense clustering of grid points in the boundary layer, the flow over an impulsively started wedge with a blunt leading edge was computed. The flow conditions assumed here are a free-stream of M=5 and a Reynolds number Re=62500 (based on free stream flow speed and a length scale equal to twice the radius of curvature of the leading edge of the wedge).

Figure 2 shows the grid system used for the computation. The grid has 64 cells from the wedge to the free-stream boundary and 48 cells from the symmetry line to the exit boundary. Grid points are clustered towards the wedge to resolve the boundary layer.

Figures 3-5 show the computed solution for this problem. Figure 3 shows contours of Mach number around the wedge shortly after the start of the simulation and before the solution reaches a steady state. This figure shows the formation of the bow shock in front of the impulsively started wedge and illustrates the unsteady nature of the simulation. Figure 4 shows the steady state solution for this problem on the unadapted grid system. Figure 4a shows contours of Mach

number around the wedge. Figure 4b shows a close up view of pressure at the leading edge. Finally, Fig. 4c shows a plot of pressure along the stagnation stream line. As Fig. 4c shows, the detached bow shock is captured mainly within two cells and stands between 0.058 and 0.062 from the leading edge. This corresponds to a ratio of stand-off distance to two times the radius of curvature at the leading edge of 0.232 to 0.25. This corresponds well to experimental value of 0.24 shown in Ref. 15 for a flow M=5 over a cylinder at $M = 5$.

Figure 5 shows the flow in the boundary layer on the wedge. The boundary layer is well captured by the current grid system. Note, for the location shown in Fig. 5b, the cell Reynolds number for the cell next to the wall is about 10^{-3}. Thus, if the same flow were computed using a pure explicit discretization like the MacCormac scheme, a 2000 fold increase in the number of time steps would be required over that used by the current semi-implicit scheme to reach steady state. For the current problem and flow solver, the cost of the semi- implicit scheme is about triple that of a pure explicit scheme. Thus, CPU time is reduced by a factor of over 600.

Supersonic flow over a blunt body at $M = 5$— adapted mesh

Figures 6-8 show a solution for the same problem as above except at $Re = 125000$. This time the results were obtained using adaptive mesh refinement to improve the resolution of the flow. A single level of refinement was used, with a refinement ratio of four. Fig. 6a shows the adapted mesh near the leading edge of the wedge. Figure 7 shows Mach number in the same region. Figure 8 shows the velocity field in the stagnation region and the formation of the boundary layer on the leading edge of the wedge. In this simulation, the refined regions were restricted to the near-wall region, effectively preventing refinement around the shock. Note, as the mesh in the boundary layer has been refined by a factor of four, the largest allowable time step for stability of the implicit scheme has been reduced by a factor of four compared to that for the unadapted grid. However, the cell Reynolds numbers for refined cells in the boundary layer on the wedge are of the order of 10^{-4}. Thus for an explicit scheme the maximum allowable time step would have been reduced by a factor of 40.

Shock reflection off an isothermal non-slip ramp

The second sample solution is for the interaction of a shock in Argon with an isothermal non-slip ramp at an angle near transition between Mach and regular reflection. The computational domain was taken to be 2.5 centimeters long horizontally. The computations were done using uniform rectangular grids—no clustering of grid points near the solid wall boundary was used in the initial grid. The solution was computed using the coarsest level grid and four refined levels, each a factor of four more refined than its predecessor.

The results for the shock–inclined ramp interaction are shown in Fig. 9 and 10. Figure 9 shows the magnitude of the horizontal velocity. Lighter colors indicate lower velocity. Figure 9a shows the entire computational domain which is covered by the coarsest grid. The figure shows well the overall shock structure. However, a Mach stem or boundary layer are not visible at this level of resolution. Figures 9b–e show the subregion near the intersection of the incident shock with the ramp at finer levels of refinement, which in this calculation were restricted to the near wall region. At the finest level of refinement, (e), the grid spacing is 0.41 microns. Fine levels of refinement clearly resolve the Mach stem and viscous boundary layer on the ramp surface, which are not apparent at coarser levels.

Figure 10 shows a plot of horizontal velocity profiles in the viscous boundary layer at several locations behind the Mach stem. Note, all distances have been normalized by the grid spacing on the finest mesh level. The figures show that the boundary layer is well captured with the present method.

Conclusions

Due to the severe restriction on time step size resulting from small cell Reynolds numbers, pure explicit discretizations of the Navier-Stokes equations are typically not practical for simulations of unsteady, medium or high Reynolds number flows. To overcome this limitation, the discretization must be at least in part implicit. In Ref. 4 new methodology was proposed for such simulation, in which the viscous terms are integrated implicitly, using the trapezoidal rule, while the inviscid terms are treated explicitly. In this paper, which can be considered a continuation of Ref. 4, the discretization is revisited and an efficient solution algorithm for the implicit system of equations is described in detail. Also, the use of adaptive mesh refinement, which is an integral part of the proposed methodology, is discussed. The effectiveness of the algorithm was demonstrated by computing both steady and unsteady flows on traditional body-fitted structured grids as well as adaptively refined Cartesian grids.

Acknowledgements

The authors wish to thank Dr. L. A. Povinelli, of the Internal Fluid Mechanics Division of NASA Lewis

Research Center and Director of ICOMP, for facilitating the cooperation between ICOMP, Lawrence Livermore National Laboratory and University of California at Berkeley that led to this paper. His contributions and support are greately appreciated.

Research at the Lawrence Livermore National Laboratory was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W–7405–Eng–48. Support under contract No. W–7405–Eng–48 was provided by the Applied Mathematical Sciences Program and the High Performance Computing and Com munications Program of the DOE Office of Scientific Computing and by the Defense Nuclear Agency under IACRO 93–817 and IACRO 94–831.

Research supported at UC Berkeley by the US Department of Energy Office of Scientific Computing under grants FDDE–FG03–94–ER25205 and FDDE–FG03–92–ER25140, and by a National Science Foundation Presiditional Young Investigator award under grant ACS-8958522.

References

1 MacCormack, R.W., "Numerical Solution of the Interaction of a Shock Wave with a Laminar Boundary Layer," Proc. Second International Conference on Numerical Methods in Fluid Dynamics, *Lecture Notes in Physics*, vol. 8, Springer-Verlag, New York, pp. 151-163, 1971.

2 Tannehill, J.C., Holst, T.L., Rakich, J.V., "Numerical Computation of Two-dimensional Viscous Blunt Body Flows with an Impinging Shock," AIAA-75-154, 1975.

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

4 Steinthorsson, E., Modiano, D. and Colella, P.; "Computations of Unsteady Viscous Flows Using Solution Adaptive Mesh Refinement in Curvilinear Body-Fitted Grid Systems" AIAA-94-2330, June, 1994. Also, ICOMP-94-17, NASA TM 106704.

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

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

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

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

9 Warren, G.P., Anderson, W.K., Thomas, J.L., and Krist, S.L., "Grid Convergence for Adaptive Methods," AIAA-92-1592; AIAA 10th Computational Fluid Dynamics Conference, June 24-26, 1991.

10 Berger, M. J. and Oliger, J.; "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *J. of Comp. Physics,* Vol. 53, pp. 484-512, 1984.

11 Berger, M. J. and Jameson, A..; "Automatic Adaptive Grid Refinement for the Euler Equations," *AIAA Journal,* Vol. 23, No. 4, pp. 561-568, April 1985.

12 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.

13 Davis, R. L. and Dannenhoffer, J. F., "Three -Dimensional Adaptive Grid Embedding Euler Technique," AIAA 93-0330, January 1993.

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

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

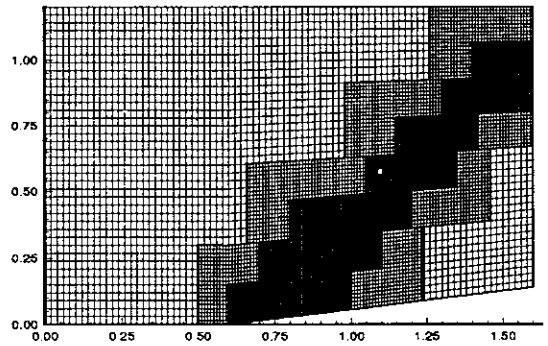


Figure 1. A grid system on three levels—a coarse grid covering the entire physical domain and two properly nested fine grid levels.

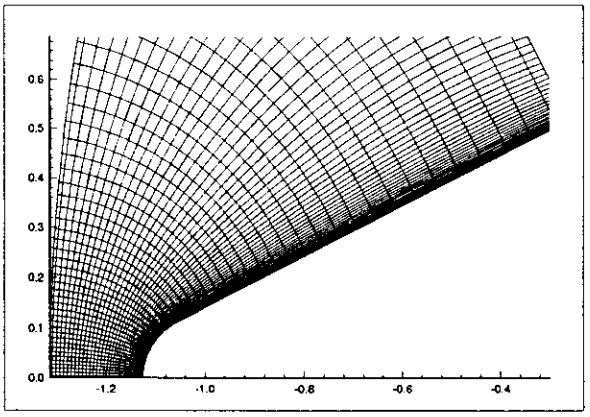


Figure 2. A body-fitted curvilinear grid system for a blunt-edge wedge.

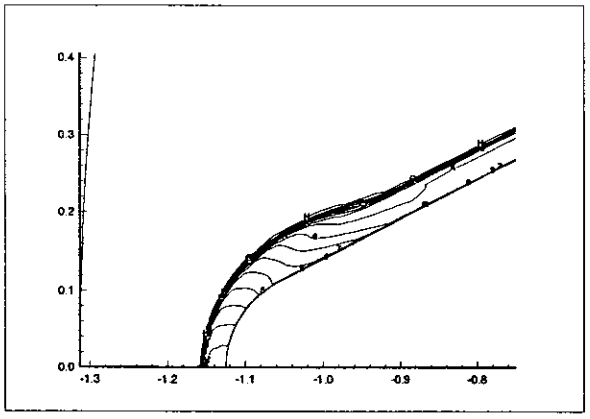
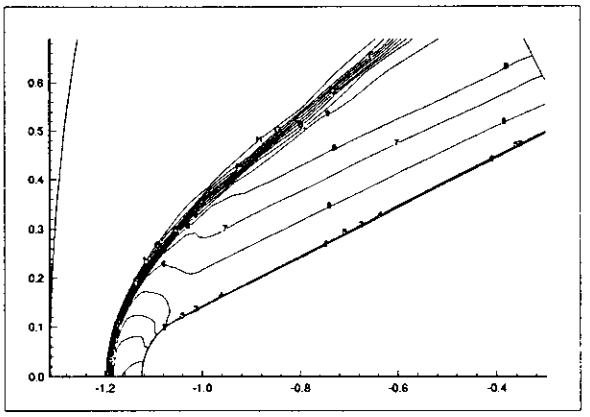
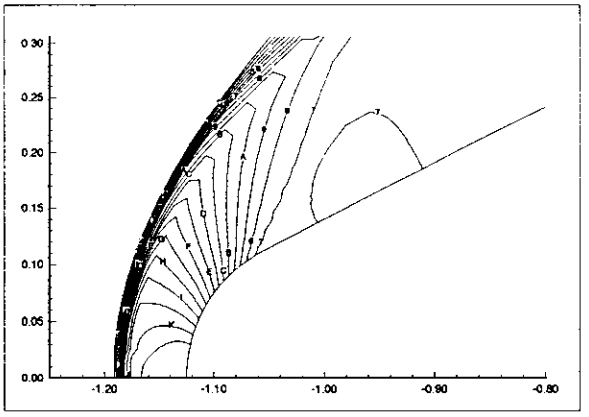


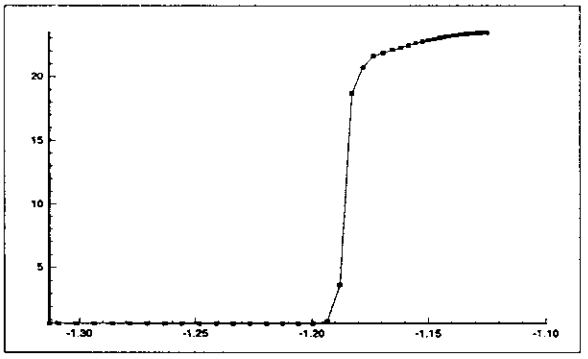
Figure 3. Supersonic flow at $M = 5$ over a blunt-edge wedge—contours of Mach number showing formation of the bow shock: $M_1 = 0.$, $\Delta M = 0.3$.



(a)

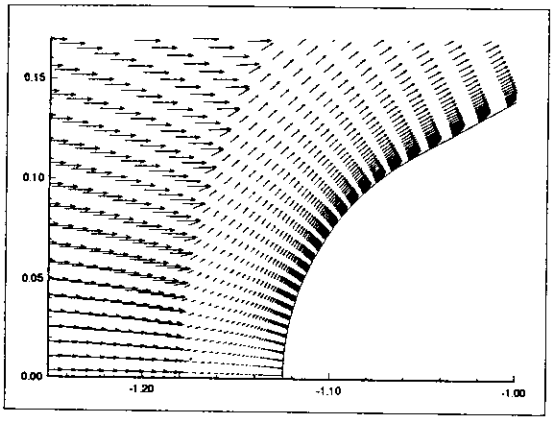


(b)

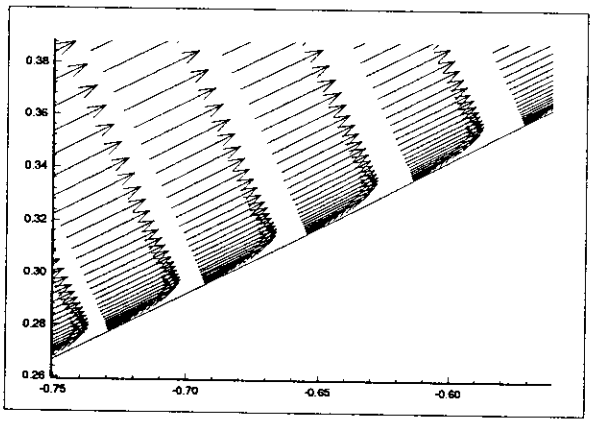


(c)

Figure 4. Supersonic flow at $M = 5$ over a blunt-edge wedge: (a) contours of Mach number; $M_1 = 0.$, $\Delta M = 0.3$, (b) contours of pressure; $p_1 = 2$, $\Delta p = 1$, (c) pressure on stagnation stream line.



(a)



(b)

Figure 5. Supersonic flow at $M = 5$ over a blunt-edge wedge: (a) velocity field near leading edge, (b) boundary layer on the wedge.

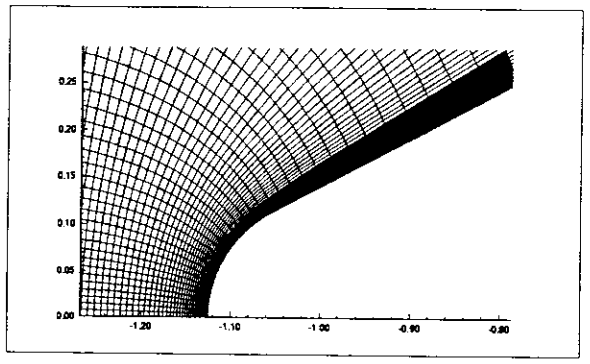


Figure 6. A body-fitted, adapted curvilinear grid system for a blunt-edge wedge.

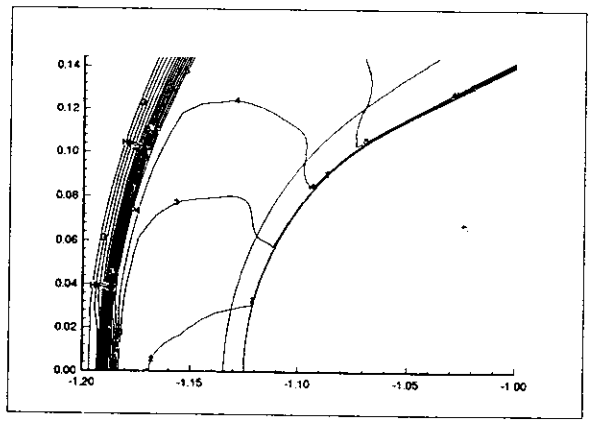


Figure 7. Supersonic flow at $M = 5$ and $Re = 125000$ over a blunt-edge wedge—contours of Mach number; $M_1 = 0.$, $\Delta M = 0.3$.

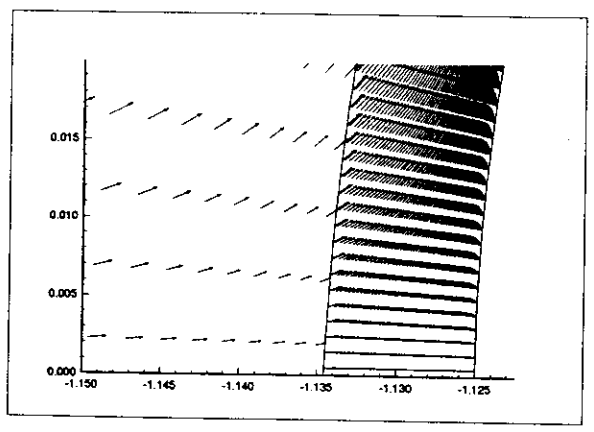
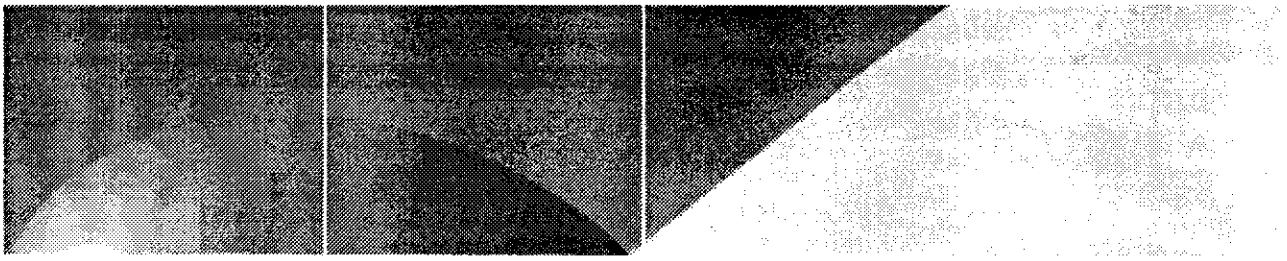
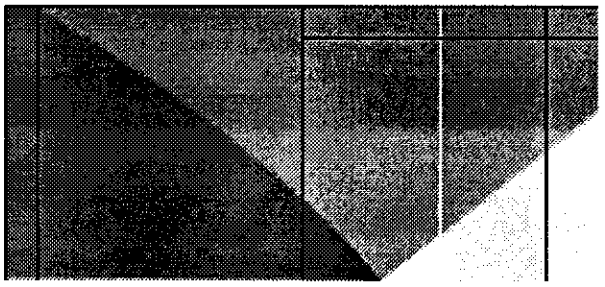


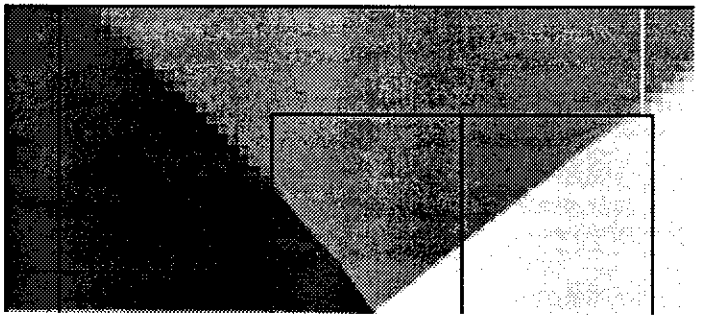
Figure 8. Supersonic flow at $M = 5$ and $Re = 125000$ over a blunt-edge wedge—velocity field near stagnation point.



(a)



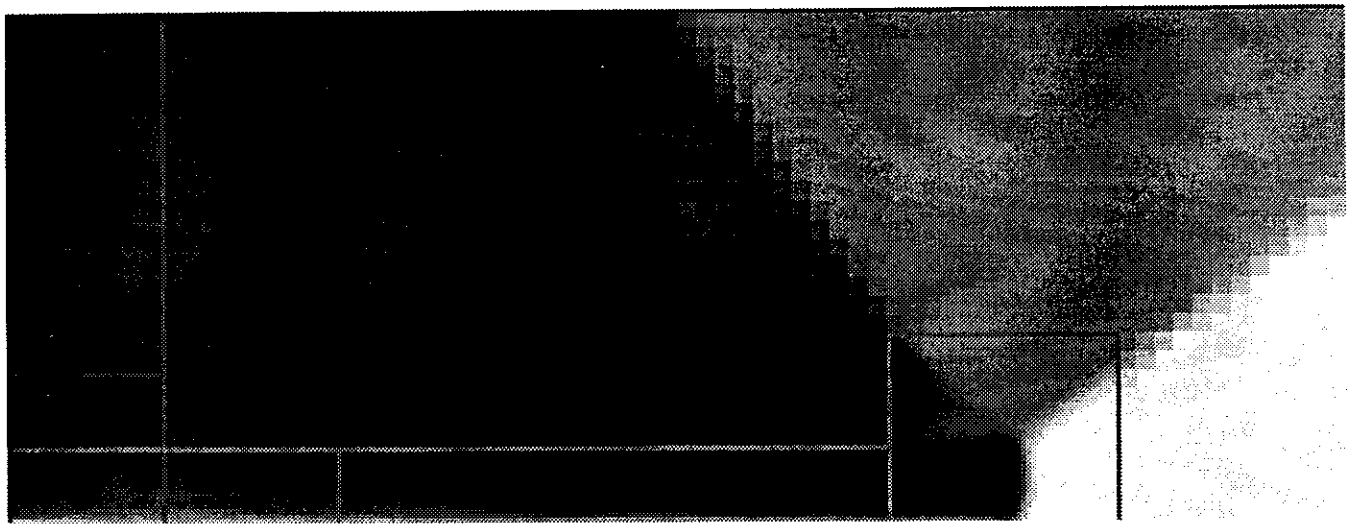
(b)



(c)

(d)

Figure 9. See caption on next page.



(e)

Figure 9 cont. Interaction of a shock in Argon with isothermal non-slip ramp at angle near transition between Mach and regular reflection—level of horizontal velocity: (a) entire computational domain at coarsest level of refinement, (b)-(e) subregion near the Mach stem at finer and finer levels of refinement.

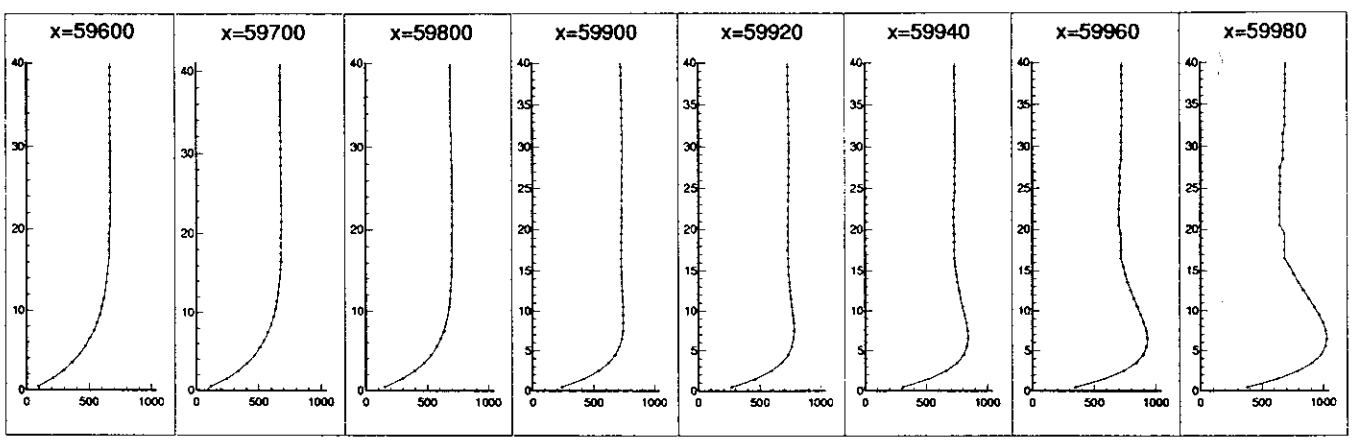


Figure 10. Interaction of a shock in Argon with isothermal non-slip ramp—profiles of horizontal velocity at eight stations behind Mach stem; distances normalized by grid spacing on the finest grid.