Automatic Adaptive Grid Refinement for the Euler Equations

Marsha J. Berger*
Courant Institute of Mathematical Sciences, New York University, New York, New York
and
Antony Jameson†
Princeton University, Princeton, New Jersey

A method of adaptive grid refinement for the solution of the steady Euler equations for transonic flow is presented. The algorithm automatically decides where the coarse grid accuracy is insufficient, and creates locally uniform refined grids in these regions. This typically occurs at the leading and trailing edges. The solution is then integrated to steady state using the same integrator (FLO52) in the interior of each grid. The boundary conditions needed on the fine grids are examined, and the importance of treating the fine/coarse grid interface conservatively is discussed. Numerical results indicate substantial computational savings for the same solution accuracy can be achieved.

I. Introduction

In computing transonic flowfields about complex geometries, it is difficult to resolve all features of the solution to the same accuracy with a uniform grid. As much as possible, the regions where the solution needs finer grid resolution are finely zoned in the initial (pre-solution) grid-generation phase. However, it is not always known in advance where those regions are, or how finely zoned they must be made. The location of the inaccurate regions changes with different flow parameters, Mach number, angle of attack, etc.

Algorithms are commonly found in the literature where the user computes a solution, regrids, and re-solves.[1] In this paper, an algorithm for automatic local grid refinement is presented. The authors describe a simple procedure to discover the regions of high error (typically the leading and trailing edges and in the neighborhood of shock waves), and to regrid by introducing any number of local rectangular fine grids. This both removes the guesswork and obtains comparable solutions at less cost than those obtained by uniformly refining the grid over the entire flowfield.

A wide variety of approaches to adapting the grid for better solution resolution have been tried. Rai and Anderson[2] use a method of clustering the grid lines in the neighborhood of a shock by "attracting" the lines into the region. Harten and Hyman[3] use an algorithm where each grid point can move within a base grid cell which stays fixed. In one dimension this method can have the same sharp resolution as a shock-fitting scheme. Recent work by Usab and Murman[4] proposed grid refinement procedures similar to those presented here, but did not incorporate the automatic error estimation in our approach.

In Sec. II of this paper, the algorithm for local grid refinement, that is, the error estimation and grid generation, is described. We also describe how to integrate the solution on these multiple grids to steady state using FLO52.[5] Since the refined grids are locally uniform patches in the same coordinate system as the coarse grid, an existing integration routine can be used with very little modification. Section III deals with the boundary conditions needed on the fine grid. The strategy is to solve an initial boundary-value problem on each grid. If a fine grid touches the airfoil, or has a far-field boundary, the same boundary conditions are applied as for a single-grid computation. The only new type of boundary that arises is the fine/coarse grid interface. The importance of treating this interface conservatively, even if the interface is in a smooth flow region, is discussed, and the procedure implemented is described in some detail. Finally, Sec. IV compares the multiple-grid results to a single-grid finely zoned run.

The same issues that arise in the interfaces between fine and coarse grids (conservation, the data structures and bookkeeping needed for this information), arise in the solution of a problem with complex geometry by component grids. By the latter is meant multiple grids in different coordinate systems. In the future it is our intention to develop these results further in that direction. Also, since the algorithm presented here keeps grids locally uniform, a simple user interface is possible. This allows, for example, the use of a vectorized integrator. The method does not have the major drawbacks of moving grid-point methods, namely, grid skewness and the "all points to the worst zone" problem, and thus seems very suitable for three-dimensional calculations as well. In this paper, a systematic study to verify that this method works is presented. We demonstrate that with no loss in the convergence rate, we can capture the accuracy of the solution on a grid twice as fine by using a coarse, global grid, and adaptively refining only those regions where the error is high.

II. Multiple-Grid Method of Adaptive Refinement

This section presents the algorithm used to solve the two-dimensional Euler equations for steady flow about an airfoil. The overall algorithm is described before going into detail about the main steps. A more detailed discussion of the structure of this algorithm is found in Ref. 6.

The solution procedure (see Sec. II.C) starts by time stepping on a single global grid. Since the initial conditions are uniform flow, we wait until the solution has settled down to, say, a residual $\approx 10^{-2}$ before applying the error estimator and subsequent adaptive strategy. The error estimator (described in Sec. II.A) is then applied at every point on the coarse grid. Those grid points where the estimate is high are flagged as needing finer grid resolution. The grid-generation algorithm creates fine grids in the same coordinate system as the coarse grid, so that every flagged point is contained in a fine grid. An important point is that the fine grids are rectangles in the computational plane. For example, the refinement at the leading edge in Fig. 1a is the center rectangle in the computational domain shown in Fig. 1b. Since the grid is periodic in the ξ direction, with the break at the trailing edge, the trailing-edge refined grids are the left- and rightmost rectangles in Fig. 1b.

Received Nov. 17, 1983; revision received April 7, 1984.

*Research Assistant Professor.

†Professor. Member AIAA.

The use of rectangles as the basis for refinement is a crucial decision. First, it allows for a very simple user interface. Any method that can be used to integrate the global coarse grid can also be used without change on each fine grid. Second, the use of rectangles makes the data structure problem tractable, since only four corner points are needed to fix the subgrid location. The storage overhead is thus on a per grid basis, rather than on a per grid point basis, and is negligible. Other methods typically use pointers for each refined cell of the coarse grid, or possibly each row. Finally, this approach to adaptive grid refinement does not suffer from the two main problems in moving grid point methods. These problems are the difficulty in controlling grid skewness, and the problem of adequately resolving several features of the solution when all points rush to the strongest feature.[7] It is clear that some mechanism of adding points in a simple way as well as moving them is called for. In the present method, if a refined grid is found to need further refinement (the error estimate at fine grid points is still too high), another, finer rectangle is added which will be nested in the existing subgrid in the same way the subgrid is nested in the coarse grid.

It is emphasized that these grids are not patched into one global grid, but are kept independently, each with its own solution vector. This means that some coarse grid solution storage is wasted (unless it participates in the solution process itself, as in a multigrid method), since the fine grid solution will always be used when it exists. The benefits seem to greatly outweigh this waste, since by preventing fragmentation the solution process on each grid can still be vectorized, and the loss in computing some extra coarse grid points is offset by the gain in efficiency due to regularity and the simplicity of the data structures.

Given this grid structure, the solution on each grid is initialized by interpolation from the coarse grid, and the time stepping continues. Section II.C describes the integration strategy for multiple grids, and reviews both the finite volume discretization scheme and the generalized Runge-Kutta time-stepping method used to advance the solution on each grid.

A. Error Estimation

In regions of smooth flow, the criterion used for refining the grid is an estimate of the error in the solution on the finest existing grid in that region. Although there is no theory for equations of mixed type, in the purely elliptic or purely hyperbolic case there are estimates for the global error in the solution in terms of the local truncation error.[8] Accordingly, the local truncation error in the solution will be estimated using ideas similar to Richardson extrapolation or deferred correction.[6] To solve

$$f(u)_x + g(u)_y = 0$$

compute

$$Q(h)U = 0$$

where U is the numerical approximation to u, and the difference operators approximating f_x and g_y based on a step size h are in Q. The local truncation error is

$$Q(h)u = \tau h^p$$

where $p=2$ for a second-order method. The term τ contains derivatives of the solution u. The goal of refining is to determine when τ is big, and reduce h, so that the same accuracy is attained over the entire flowfield. The idea is to estimate the error using Richardson extrapolation-type estimates by differencing on a grid with mesh spacing $2h$ using every other point of the computed solution U. Compute

$$Q(2h)U \approx (2^p - 1)\tau h^p$$

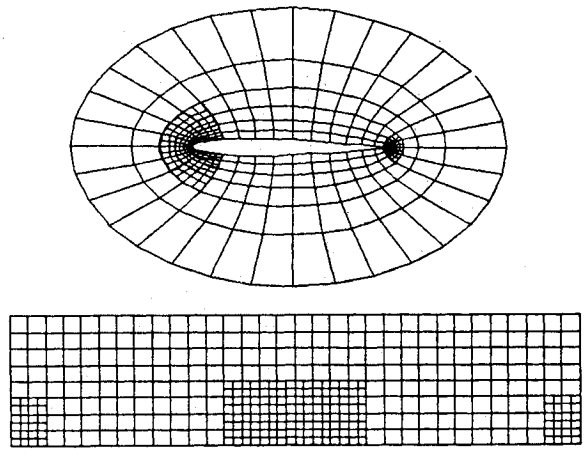

Fig. 1 Fine grids at the leading and trailing edges.

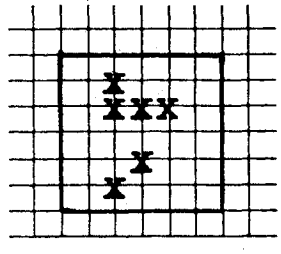

Fig. 2 Fine grid generation around flagged points.

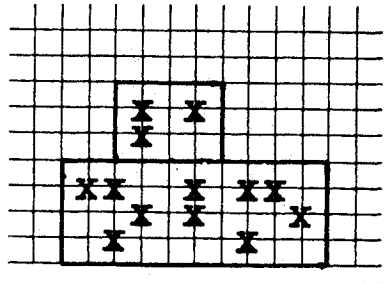

Fig. 3 Two fine grids generated around one group of flagged points.

In the steady-state calculation, the residual $Q(h)U$ is driven to zero, but the coarsened grid residual will not be zero. Thus,

$$\frac{Q(2h)U}{2^p - 1} \approx \tau h^p$$

can be used as an estimate of the error at each point.

Notice that it is unnecessary to know the exact form of the truncation error τ for this method. Also, this residual calculation is identical to the first stage of the regular Runge-Kutta integration step but on a $2h$ grid. For the error estimation step, the computer code is merely changed to read $i+2$ instead of $i+1$ (and so on) when updating the ith point. (In fact, this can sometimes be done automatically in Fortran by changing the dimension statements when declaring the arrays in an integration subroutine.) The computational overhead of this estimator is thus less than one extra integration step for the whole computation.

In regions where the flow is discontinuous, this procedure no longer gives a valid error estimate. However, it acts as a trigger for mesh refinement in the presence of a strong shock. It is found in these results, however, that the largest errors are at the leading edge and then trailing edge of the airfoil. Of

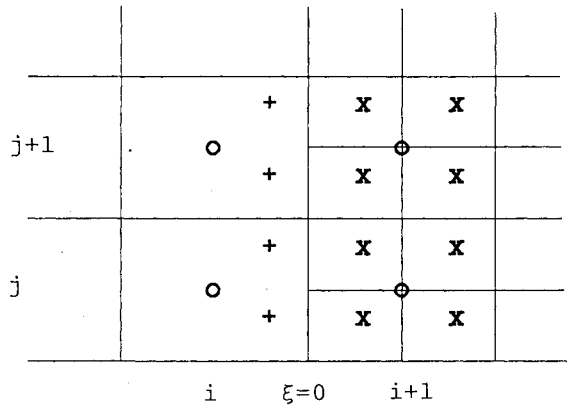

Fig. 4 Fine/coarse grid interface.

course, this depends on the underlying global grid resolution. For strong shocks, a fine grid is also created in the region of the shock. This general procedure has the advantage that it is not necessary to know the location of the shock before the start of the computation.

B. Grid Generation

The output from the error estimation routine is a list of (coarse) grid points with high error estimates, indicating that a refined zone is needed in that region. The grid-generation routine separates the points into appropriate groups so that a (logically) rectangular grid can be placed around each group. This proceeds in two phases. First, the points that are in different parts of the domain (such as leading and trailing edges) are separated into different groups. Around each group, a rectangle is formed that is large enough to include all points in that group. This new grid is then slightly enlarged (by one or two coarse grid points), to ensure that the fine grid boundary, where special interpolation formulas will be used for the boundary differencing, is in a region with a small error estimate. Figure 2 illustrates this procedure schematically.

The only possible exception to this procedure is shown in Fig. 3. It may happen that two different grids are created around one cluster of points to minimize the size of the (unnecessarily) refined region. Details of this exceptional case can be found in Ref. 9.

One last remark about the rectangular grids should be made. It may happen that the zone needing refinement is oblique to the underlying grid, for example, for an oblique shock. It could be advantageous to be able to align the fine grid so that the coordinates are approximately normal and tangent to the discontinuity. A rotated difference scheme has been used by Jameson[10] for the potential equation. Recent results by Davis[11] for the Euler equations show much better performance for first-order upwind schemes if they are rotated to align with the shock. The grid-generation procedure has the capability to produce grids with this alignment property. However, interpolation procedures have not yet been developed that treat the fine/coarse grid interface conservatively. Work is still in progress on this point.

C. Integration Procedure

It is very easy to solve the equations on the grid structure described above. One step should be taken on all grids using the Runge-Kutta finite volume integrator described below. Since we are interested in the steady-state solution, we use pseudo-time steps with a fixed Courant number. For time-dependent calculations, for reasons of both stability and efficiency, it is best to take several smaller time steps on the fine grids for every one coarse grid step. We have experimented with taking several steps on a large fine grid for every coarse grid step. The optimal strategy for the number of iterations to be done on each grid at a time is still an open question. This will be an especially important consideration in large computations where secondary storage is used.

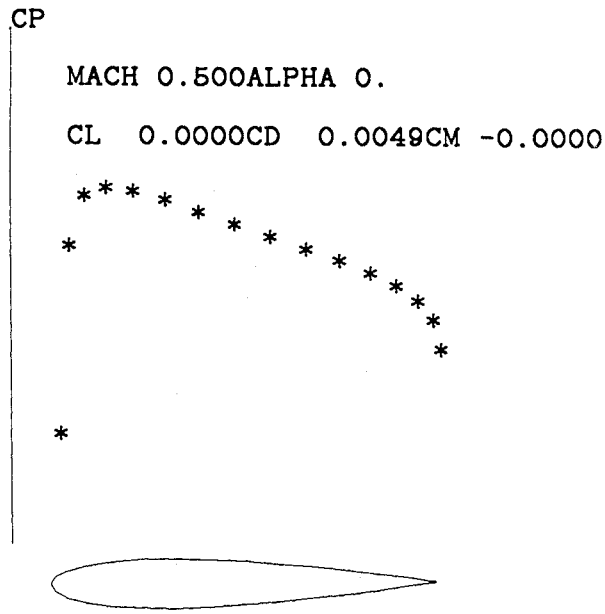

Fig. 5 Pressure coefficient using a 32×8 coarse grid.

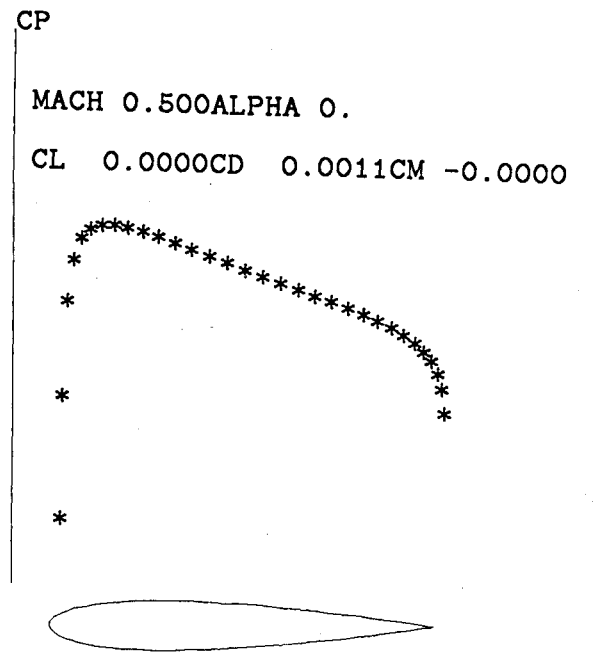

Fig. 6 Pressure coefficient using a 64×16 coarse grid.

We briefly review the finite volume Runge-Kutta time-stepping procedure which is the integrator for this adaptive algorithm. For details, see Ref. 12 and the discussion of the FLO52 program. The full Euler equations in two dimensions are written in integral form as

$$\frac{\partial}{\partial t}\int\int w\,dx\,dy + \int_{\partial\Omega} f\,dy - g\,dx = 0$$

where

$$w = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix} \quad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{bmatrix} \quad g = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{bmatrix}$$

The equations are approximated in a computational domain where the variables are cell-centered. The flux is evaluated at the boundary of a given cell using the average of the values in the adjacent cells. This spatial discretization procedure leads to a system of ordinary differential equations of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}(hw) + Qw - Dw = 0$$

where Qw is the approximation to the Euler terms, Dw an added dissipative term, and h the cell area. The dissipation is introduced by a combination of second- and fourth-order differences which are switched on by pressure gradients. The same dissipation formulas are used in the integration step on each grid, except at the boundaries of the fine grids, where the fourth-order stencil is too large. In this case only the second-order dissipation is used.

The ordinary differential equations are integrated using a modified four stage Runge-Kutta scheme in which the dissipative terms are only evaluated once. At each time step the solution is updated by the following sequence:

$$w^{(1)} = w^{(0)} - \frac{\Delta t}{4h}(Qw^{(0)} - Dw^{(0)})$$

$$w^{(2)} = w^{(0)} - \frac{\Delta t}{3h}(Qw^{(1)} - Dw^{(0)})$$

$$w^{(3)} = w^{(0)} - \frac{\Delta t}{2h}(Qw^{(2)} - Dw^{(0)})$$

$$w^{(4)} = w^{(0)} - \frac{\Delta t}{h}(Qw^{(3)} - Dw^{(0)})$$

where $w^{(0)}$ is the value at the beginning of the time step, and $w^{(4)}$ the final updated value. The time-step limit for this scheme is almost the same as that of a standard fourth-order Runge-Kutta scheme.

The far-field boundary values are partially specified from freestream values, and partially extrapolated from the Riemann invariants, depending on whether the flow is supersonic or subsonic, and whether the boundary is an inflow or outflow boundary. At the body, only the pressure is needed to advance the variables in the cells nearest the wall. The pressure is computed by an extrapolation formula based on the normal momentum equations. The slight modification of this required at the coarse/fine interface is described in Sec. III.

III. Fine-Grid Boundary Conditions

In this section the difference equations used at the interface between a coarse and a fine grid are discussed. The procedure developed herein is designed for use with the finite volume difference scheme. More general procedures are described in Ref. 13.

The procedure is illustrated using a refinement ratio of 2 between grids. In Fig. 4, the coarse grid variables are marked with o , the fine-grid variables with an X. Notice that there are coarse points "underneath" the fine grid. The solution is computed over the entire coarse grid including these points when a coarse grid integration step is taken. However, the coarse grid points underneath the fine grid are updated after each coarse step by replacing the solution there with the volume-weighted average of the solution at the four nearest fine points. The final output uses the values from the finest grid covering each region.

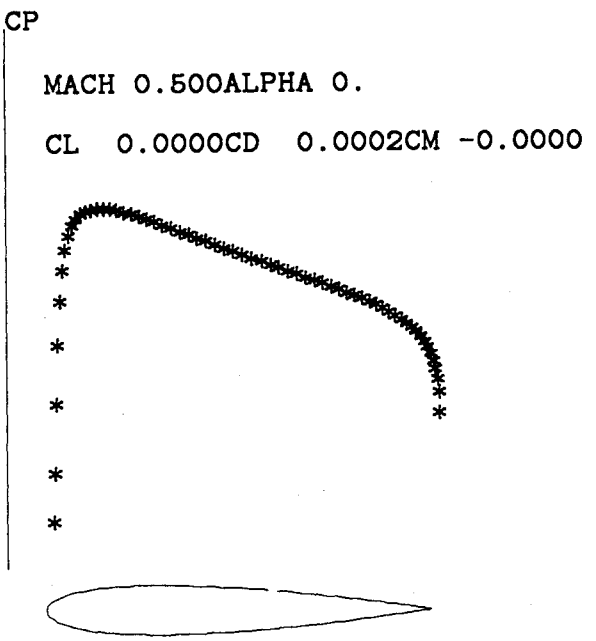

Fig. 7 Pressure coefficient using a 128×32 coarse grid.

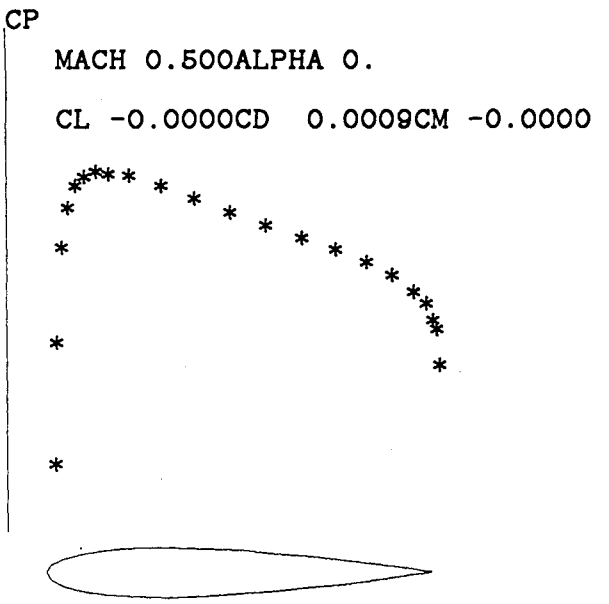

Fig. 8 Pressure coefficient using the grid in Fig. 9.

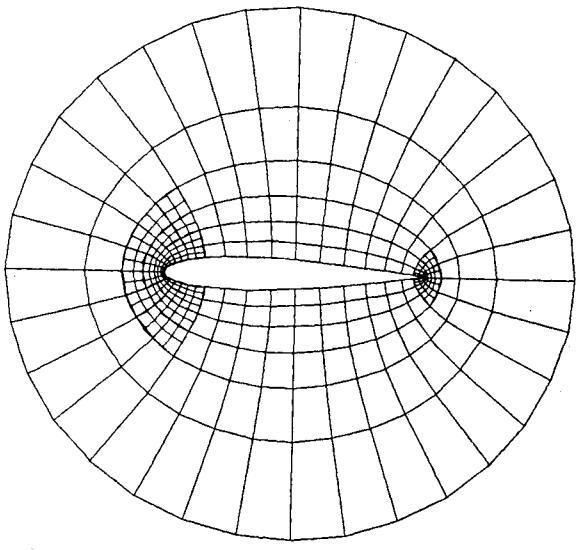

Fig. 9 32×8 coarse grid with one level of refinement.

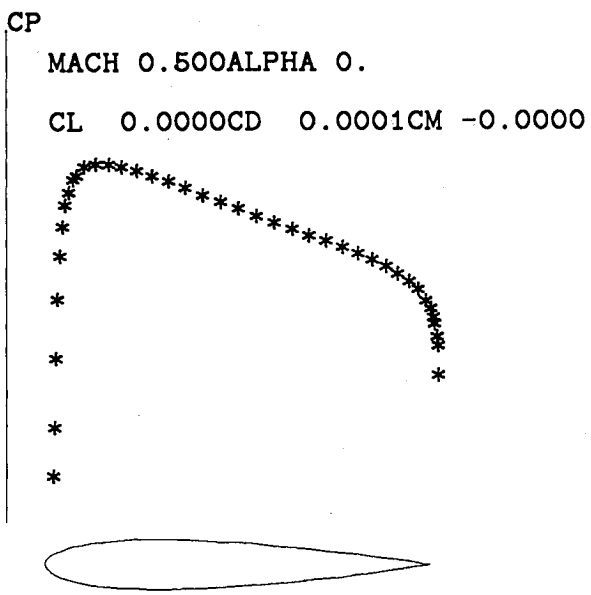

Fig. 10 Pressure coefficient using the grid in Fig. 11.

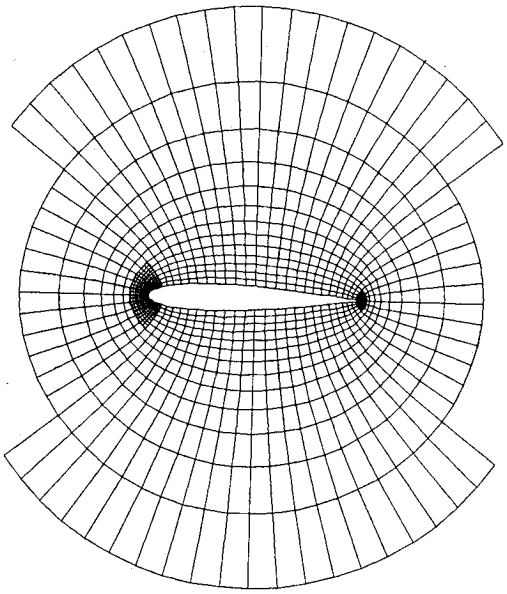

Fig. 11 64×16 coarse grid with one level of refinement.

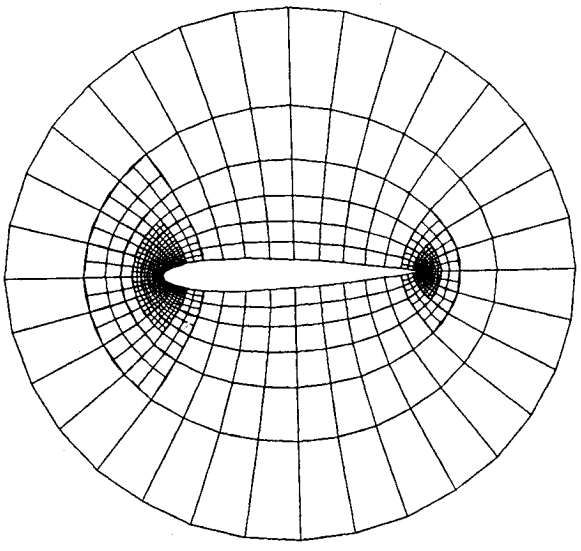

Fig. 12 32×8 coarse grid with two levels of refinement.

To take a step on the fine grid, the flux across the line $\xi = 0$ into the fine grid must be calculated. It is algorithmically advantageous to introduce an "outside" column of solution values (denoted by the plus signs in Fig. 4). The flux can then be calculated in the same way for the first cell and the interior cells of the fine grid. This also mimics the coarse-grid setup, where an extra column is kept on each side of the periodic boundary. The easiest way to determine this column of fine grid values is by interpolation from the coarse grid. Point p would be set by linear interpolation from coarse grid points v_{ij}, $v_{i,j+1}$, $v_{i+1,j}$, and $v_{i+1,j+1}$. To maintain accuracy, the values at $v_{i+1,j}$ and $v_{i+1,j+1}$ on the coarse grid would be replaced by the volume-weighted average of the four neighboring fine grid points after every coarse grid integration step. In this way, each grid can still be integrated independently, in a manner that still vectorizes, and only a small amount of "fix-up" work along the boundaries of the fine grids need be done.

Unfortunately, there is no reason for this procedure to be conservative, which in this case means that the sum of the fluxes into the coarse cells at the interface [computed for example using the value $(v_{i,j} + v_{i+1,j})/2$] is equal to that computed on the fine grid into the fine cells on the right of the interface. It is important to maintain conservation in order to guarantee the correct shock location in transonic flowfields. This is especially relevant since there will often be a fine grid in the region of a shock, and so the interface between the fine and coarse grids will be near the shock. When this nonconservative procedure is used, the computed drag coefficient can differ by as much as 20% from the more accurate conservative calculation described later.

An alternative interface procedure which is conservative is to calculate the flux on the coarse grid, and divide it in half for the two adjacent fine cells. This would bypass setting the outside variables, and calculate a boundary flux for the fine grid directly. Unfortunately, this is unstable, as can be seen from a linear analysis of the interface. The treatment in this case yields the linear relationship

$$v_{0,1} + v_{1,1} + v_{0,2} + v_{1,2} = 2(u_{i,j} + u_{i+1,j})$$

where u approximates the solution on the coarse grid, and v approximates the solution on the fine grid. It turns out that any boundary scheme that couples the fine points across the interface can give rise to an oscillatory wave emanating from the interface into the fine grid, and supported by the central differenced (linearized finite volume) scheme. Another alter-

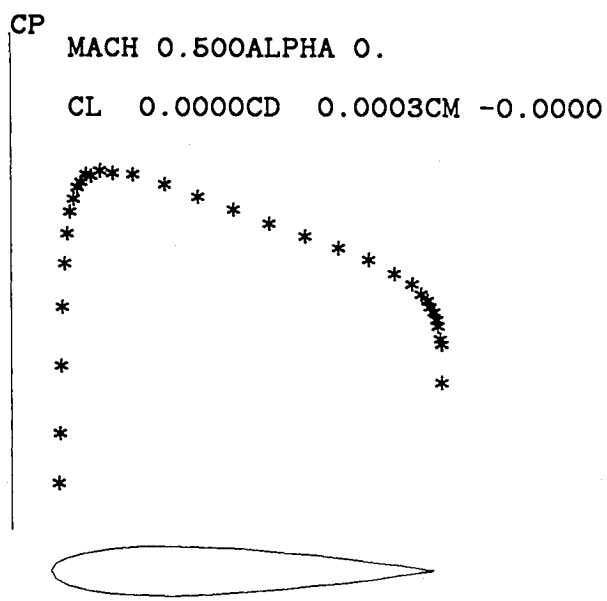

Fig. 13 Pressure coefficient using the grid in Fig. 12.

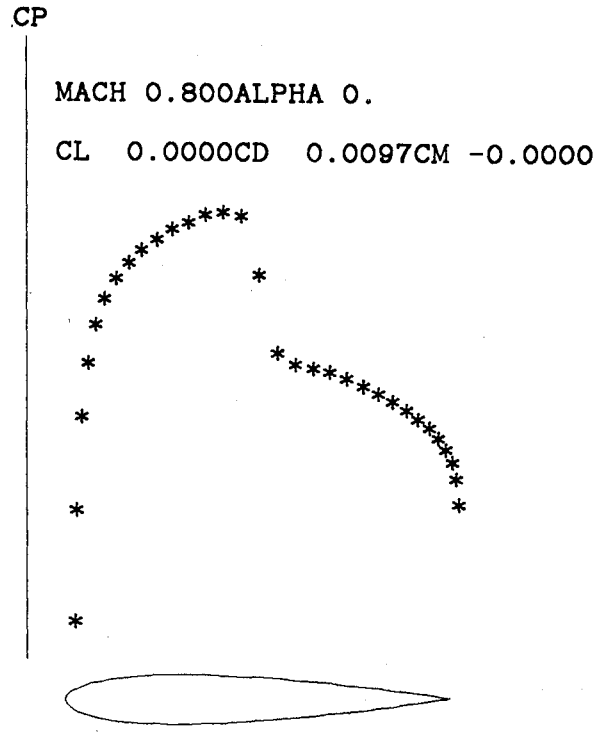

Fig. 14 Pressure coefficient using a 64×16 coarse grid.

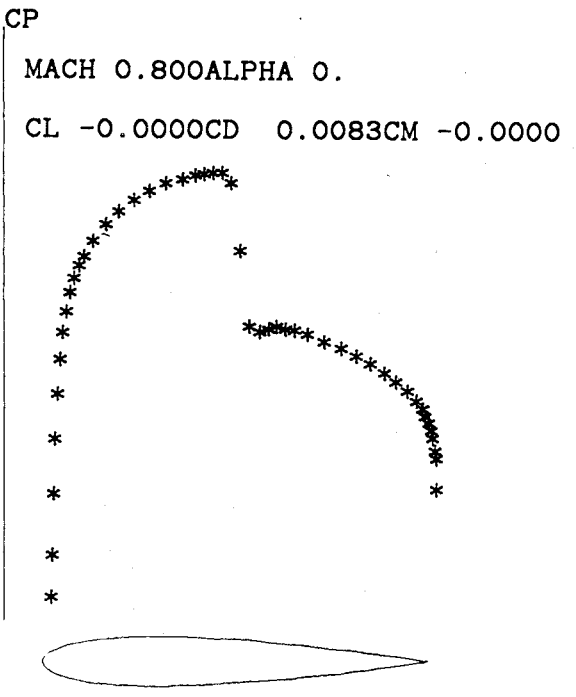

Fig. 16 Pressure coefficient using the grid in Fig. 15.

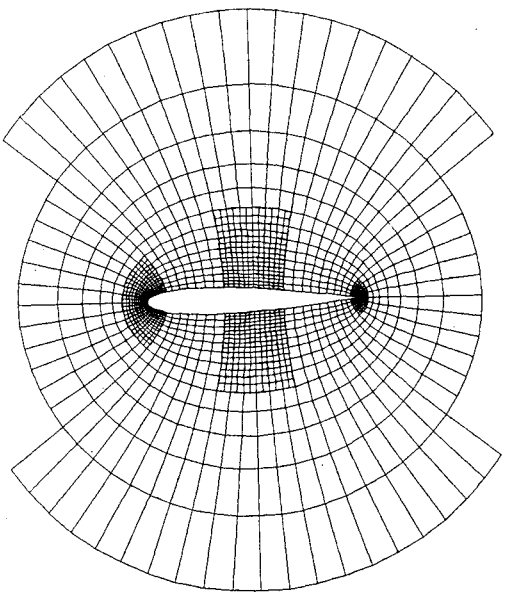

Fig. 15 64×16 coarse grid with one level of refinement.

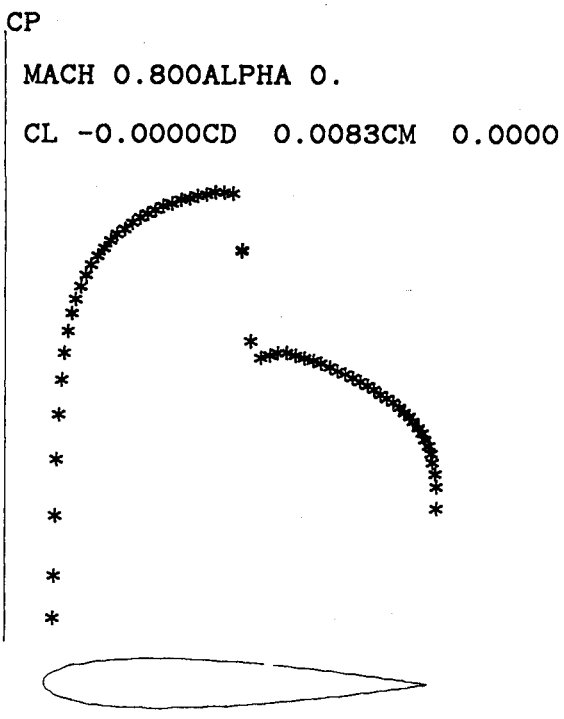

Fig. 17 Pressure coefficient using a 128×32 coarse grid.

native, that of calculating the flux directly from the one coarse point and adjacent two fine points, is stable, but of lower order accuracy. In effect, it treats the solution in the column of plus signs as piecewise constant in each coarse cell, instead of linear.

The procedure chosen is a variation of interpolation. The cells with plus signs are obtained from interpolation from the coarse grid, and the fine grids are then advanced. The coarse grid fluxes are determined as usual during the regular integration step, but then the value at each coarse grid point nearest the interface is "fixed" so that the flux at the interface equals the sum of the two fine cell fluxes. In this way, conservation is enforced, second-order accuracy is maintained, and the only extra computational work is done along the boundary. (This method thus avoids having to check every coarse point to determine whether it is located at an interface, which would be unacceptable overhead.) In experiments with these interface equations, when the interface is forced to be right at the location of a shock, no loss of accuracy is observed in the solution. This has also been observed by Rai,[14] who computed discontinuous solutions passing through zonal boundaries where the ratio of the grid spacing is as much as 10:1. No loss of accuracy was observed using conservative interface conditions.

A special treatment is required when the interface coincides with the body. In order to interpolate for a fine point value at the body, a coarse grid value is needed at the body as well. Since only the pressure is computed at the body, values are

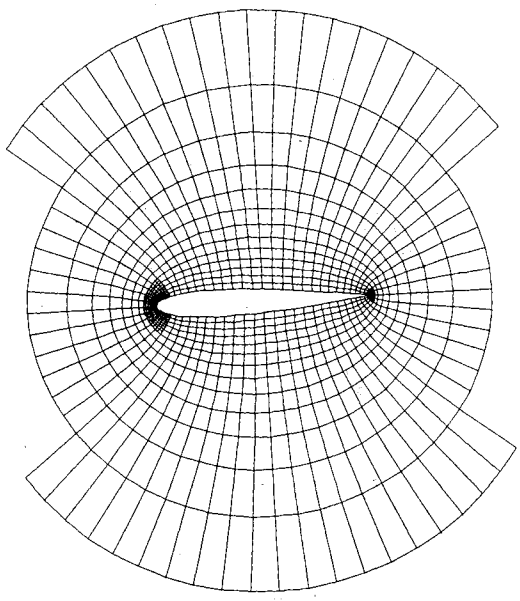

Fig. 18 64×16 coarse grid with one level of refinement, using a less stringent error tolerance.

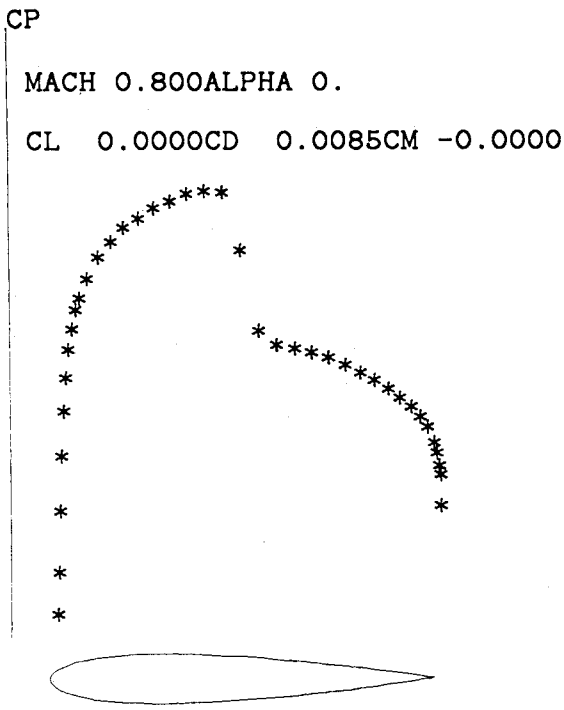

Fig. 19 Pressure coefficient using the grid in Fig. 18.

needed for the density, x and y momentum, and energy. These are computed by setting the momentum normal to the body to zero, setting the tangential momentum to be identical to that one cell over, and setting the energy to its steady-state constant value. In practice, there is little difference between this procedure and simple linear extrapolation of the missing coarse-grid values from the interior.

IV. Numerical Results

Results comparing the solution computed on a coarse grid, on a coarse grid with patched fine grids, and on a uniformly refined grid are presented. In all cases, by refining a fraction of the grid, the accuracy of the solution on a uniformly fine grid is recovered at less than one-half the cost, and sometimes as little as 1/20 the cost.

The first test case is for nonlifting subsonic flow over an NACA 0012 airfoil. The Mach number is 0.500 with a 0-deg angle of attack. Figure 5 shows the pressure coefficient calculated on a 32×8 coarse grid. The drag coefficient is 0.0049. Figure 6 shows the pressure coefficient calculated on a 64×16 grid. The drag coefficient is 0.0011. Figure 7 shows the solution using a grid 128×32, with a drag coefficient of 0.0002. The drag coefficient is converging like h^2 to its expected value of zero. Figure 8 shows a refined grid solution based on a 32×8 underlying coarse grid, with refined grid patches as shown in Fig. 9. The drag coefficient in this case is 0.0009. Figure 10 shows a refined grid solution based on a 64×16 underlying coarse grid. The refined grid for this case is shown in Fig. 11. The drag coefficient is reduced to 0.0001. In both cases, the accuracy of the solution on the uniformly next finer level grid is recovered by using small grid patches at the leading and trailing edges of the airfoil. More dramatic savings can be achieved by using three nested levels of grid refinement, as shown in Fig. 12. The solution, with a drag coefficient of 0.0003, is shown in Fig. 13. If the number of iterations on each grid multiplied by the number of grid points in each grid are counted, the computational work using three levels of grid refinement is 1/20 the work on a 128×32 grid, for comparable solution accuracy.

The second test case is transonic flow containing a shock wave. Figure 14 shows the pressure coefficient for an NACA 0012 airfoil at Mach 0.8 with a 0-deg angle of attack. The mesh used for this computation is 64×16 cells. When the grid is refined (using an error tolerance of 0.005), as shown in Fig. 15, the solution obtained is almost identical to the solution computed on a 128×32 mesh (compare Figs. 16 and 17). In the coarse-grid run, the entropy behind the shock was computed to be 0.0072, in the multiple grid run it was 0.0052, and in the fine grid run the entropy was 0.0054. In this mesh refined solution, 21% of the coarse grid was refined by a factor of 2 in both coordinate directions. The cost of integrating the mesh refined run was thus roughly one-half the cost of the 128×32 grid run. If the error tolerance for mesh refinement is less stringent (0.025), so that only the leading and trailing edges are refined (as in Fig. 18), the solution is only slightly worse (Fig. 19). The entropy production across the shock is 0.0046 in this case. In this run only 10% of the coarse grid is refined, and so the overall cost of the solution using two levels of refinement is roughly 35% of the fine grid cost.

V. Conclusions

A general adaptive mesh refinement algorithm is applied to the computation of the solution of the Euler equations for transonic flow. The adaptive gridding is automatic. The user supplies only the subroutine to integrate the solution on a computational rectangle. The algorithm decides where the coarse grid is insufficient, and places finer grid patches in those regions. Since the work presented herein is with rectangular grids, the data structures that keep track of the grids are simple. Numerical results show that significant computational savings are possible.

Acknowledgments

The work of the first author was supported in part by Department of Energy Contract DEAC0276ER03077-V. The work of the second author was supported in part by the Office of Naval Research under Grant N00014-81-K-0379 and by NASA Langley Research Center under Grant NAG-1-186.

References

[1]Nakamura, S. and Holst, T. L., "A New Solution-Adaptive Grid Generation Method for Transonic Airfoil Flow Calculations," NASA TM 81330, Oct. 1981.

[2]Rai, M. M. and Anderson, D., "The Use of Adaptive Grids in Conjunction with Shock-Capturing Methods," AIAA Paper 81-1012, June 1981.

[3]Harten, A. and Hyman, J. M., "Self-Adjusting Grid Methods for One-Dimensional Hyperbolic Conservation Laws," Los Alamos Report LA-9105, 1981.

[4]Usab, W. Jr. and Murman, E. M., "Embedded Mesh Solutions of the Euler Equation Using a Multiple-Grid Method," *Proceedings of the AIAA Computational Fluid Dynamics Conference*, Danvers, Mass., Paper 83-1946, July 1983.

[5]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," AIAA Paper 81-1259, 1981.

[6]Berger, M. and Oliger, J., "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," *Journal of Computational Physics*, Vol. 53, March 1984, pp. 484-512.

[7]Anderson, D., "Adaptive Mesh Schemes Based on Grid Speeds," AIAA Paper 83-1931, 1983.

[8]Isaacson, E. and Keller, H., *Analysis of Numerical Methods*, John Wiley & Sons, New York, 1966.

[9]Berger, M., "Data Structures for Adaptive Grid Generation," submitted to *SIAM J. Sci. and Stat. Comp.*

[10]Jameson, A., "Iterative Solutions of Transonic Flows over Airfoils and Wings, Including Flows at Mach 1," *Communications on Pure and Applied Mathematics*, Vol. 27, 1974, pp. 283-309.

[11]Davis, S., "A Rotationally Biased Upwind Difference Scheme for the Euler Equations," ICASE Tech. Memo. 172179, July 1983.

[12]Jameson, A., "Steady-State Solution of the Euler Equations for Transonic Flow," *Transonic, Shock and Multidimensional Flows: Advances in Scientific Computing*, Academic Press, New York, 1982, pp. 37-70.

[13]Berger, M., "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations," Ph.D. Thesis, Computer Science Department, Stanford University, Stanford, Calif., 1982.

[14]Rai, M., "Conservative Treatment of Zonal Boundaries for Euler Equation Calculations," AIAA Paper 84-0164, 1984.