Automatic Adaptive Grid Refinement for the Euler Equations

*Marsha J. Berger**
Courant Institute of Mathematical Sciences
251 Mercer St.
New York University
New York, NY 10012

Antony Jameson+
Princeton University
Dept. of Mechanical and Aerospace Engineering
Princeton, NJ 08544

Abstract

We present a method of adaptive grid refinement for the solution of the steady Euler equations for transonic flow. Our 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. We examine the boundary conditions needed on the fine grids, and discuss the importance of treating the fine/coarse grid interface conservatively. Numerical results are presented.

1. Introduction

In computing transonic flow fields 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 arc, or how finely zoned to make them. 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, re-grids, and re-solves [1]. In this paper, we present an algorithm for automatic local grid refinement. We 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 re-grid 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 flow field.

*Supported in part by Department of Energy Contract No. DEAC0276ER03077-V.
+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

A wide variety of approaches to adapting the grid for better solution resolution have been tried. Rai and Anderson [2] have a method of clustering the grid lines in the neighborhood of a shock by "attracting" the lines into the region. Harten and Hyman [3] have 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 Mur man [4] proposed grid refinement procedures similar to ours, but does not incorporate the automatic error estimation in our approach.

In section 2 of this paper, we describe the algorithm for local grid refinement, that is, the error estimation and grid generation. We also describe how to integrate the solution on these multiple grids to steady state, which we do using FLOS2 [5]. Since the refined grids are locally uniform patches in the same coordinate system as the coarse grid, we are able to use an existing integration routine with very little modification. Section 3 deals with the boundary conditions needed on the fine grid. Our strategy is to solve an Initial boundary value problem on each grid. If a fine grid touches the airfoil, or has a farfield boundary, the same boundary conditions are applied as would be used for a single grid computation. The only new type boundary that arises is the fine/coarse grid interface. We discuss the importance of treating this interface conservatively, even if the Interface is in a smooth flow region, and describe in some detail the procedure which we implement. Finally, section 4 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 bookeeping needed for this information), arise in the solution of a problem with complex geometry by component grids. By the latter we mean multiple grids in different coordinate systems. In the future we intend to apply our results that direction. Also, since our algorithm 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 3D calculations well. In this paper, we present a systematic study to verify that this method works. 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.

2. Multiple Grid Method of Achieve Refinement

This section presents the algorithm we use to solve the 2D Euler equations for steady flow about an airfoil. We describe the overall algorithm before going into detail about the main steps. A more detailed discussion of the structure of this algorithm is found in [6].

The solution procedure (described in section 2.3) 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 section 2.1) 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 figure 2.la is the center rectangle In the computational domain shown In figure 2.lb. Since the grid is periodic in the ξ direction, with the break at the trailing edge, the trailing edge refined grids are the left most and rightmost rectangles in figure 2.lb.

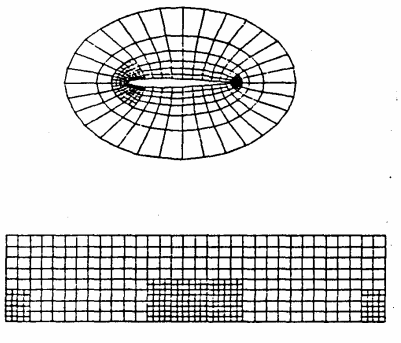

Figure 2.1 Fine grids at the leading and trailing edges.

The use of rectangles as the basis for refinement is a crucial decision. First, It allows for a very simple user interface. The integrator which is used on the global coarse grid can also be used without change on each fine grid. Secondly, the use of rectangles makes the data structure problem tractable, since only four corner points are needed to fix the sub-grid location. The storage overhead is thus on a per grid basis, rather than on a per grid point basis, an4 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 to add points in a simple way as well as moving them is called for. In our 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 sub-grid in the same way the sub-grid is nested in the coarse grid.

We emphasize 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, such as in a multi-grid method), since we will always use the fine grid solution 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 2.3 describes the integration strategy for multiple grids, and reviews both the finite volume discretization scheme and the generalized Runge Kutta time stepping method which is used to advance the solution on each grid.

2.1. Error Estimation

In regions of smooth flow, the cirteria we use 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, we will estimate the local truncation error in the solution using ideas similar to Richardson extrapolation or deferred correction [6]. To solve

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

we 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 stcpsize h am 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 flow field. The idea is to estimate the error using Richardson extrapolaton-type estimates by differencing on a grid with mesh spacing 2h using every other point of the computed solution U. We compute

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

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

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

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. Secondly, 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 i^{th} 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. We have found in our results however, that the largest error is at the leading edge and then trailing edge of the airfoil. Of 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.

2.2. Gild 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 edge and trailing edge) are separated into different groups. Around each group, a rectangle is formed which 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 estimate. Figure 2.2 illustrates procedure schematically.

The only possible exception to this above procedure is shown in Figure 2.3. It may happen that two different grids are created around one cluster of points, in order to minimize the size of the (unnecessarily) refined region. Details of this exceptional case can be found in [9].

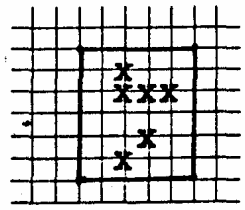

Figure 2.2 Fine grid generation around flagged points.

We make one last remark about the rectangular grids. 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 were approximately nor mal and tangent to the discontinuity. A rotated difference scheme has been used by Jarneson [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

gilds with this alignment property. However, interpolation procedures have not yet been developed which treat the fine/coarse grid Interface conservatively. Work is still in progress on this point.

2.3. Integration Procedure

It is very easy to solve the equations on the grid structure described above. We take one step on all grids using the Runge Kutta finite 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

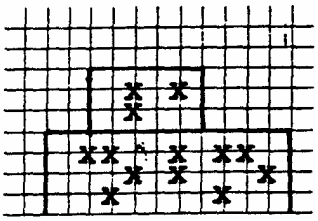

Figure 2.3 Two fine grids generated around one group of flagged points.

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.

We briefly review the finite volume Runge Kurta time stepping procedure which is the integrator for this adaptive algorithm. For details, see [12] and the discussion of the FLO52 program. The full Euler equations in 2D are written in integral form,

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

where

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

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 aver age of the values in the adjacent cells. This spatial discretization procedure leads to a system of ordinary differential equations. These have the form

$$\frac{d}{dt}(hw) + Qw - Dw = 0,$$

where Qw is the approximation to the term, and where h is the cell area. The dissipation is introduced by a combination of second and fourth order differences, which arc switched on by pressure gradients. The same dissipation formulas arc 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 ease we use only the second order dissipation.

The ODE 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:

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

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

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

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

where $w^{(0)}$ is the value a; the beginning of the time step, and $w^{(4)}$ is 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 farfield boundary values are partially specified from freestream values, and partially extrapolated from the Riemann invariants, depending on whether the flow is supersonic or sub sonic, 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 nor mal momentum equations. The slight modification of this required at the coarse/fine interface is described in section 3.

3. Fine Grid Boundary Conditions

In this section we discuss the difference equations used at the interface between a coarse and fine grid. The procedure which we have

developed is designed for use with the finite volume difference scheme. More general procedures arc described in [13].

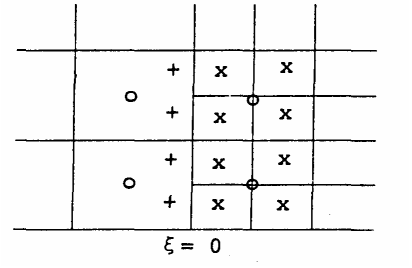

Figure 3.1 Fine/coarse grid inter face.

We illustrate the procedure using a refinement ratio of 2 between grids. In figure 3.1, the coarse grid variables arc 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.

To take a step on the fine grid, the flux across the line $\xi = 0$ into the fine grid must be calculated. It is advantageous to introduce an outside column of solution values (denoted by the plus signs in figure 3.1). 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 $v_{j,j}$ $v_{i,\,j+1}$, $v_{i+1,\,j}$. $v_{i+1,\,j+1}$. To maintain accuracy, the values at v_{j+1j} 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 "fixup" 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 $\dfrac{v_{i,\,j} + v_{i+1,\,j}}{2}$ 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 flow fields. 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.

An alternative interface procedure which is conservative is to calculate the flux on the coarse grid, and divide it in half for the adjacent two 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

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

where u approximates the solution on the coarse grid, and v approximates the solution on the fine grid. It turns out that boundary scheme which 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 alternative, 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 a cell, instead of linear.

The procedure we have 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 arc 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 he 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.

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 needed for the density, x and y momentum, and energy. These are computed by setting the momentum normal to the body to 0,

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.

4. Numerical Results

We present results comparing the solution computed on a coarse grid, on a coarse grid with patched fine grids, and on a uniformly refined grid. 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 half the

The first test case is for non-lifting subsonic flow over a NACA 0012. The Mach number is .500 with zero degrees angle of attack. Figure 4.1 shows the pressure coefficient for a run with a grid size 32 by 8, shown In Figure 4.2. The drag coefficient is .0049. Figure 4.3 is from a grid of size 64 by 16 with drag coefficient .0011. The grid is shown in Figure 4.4. Figure 4.5 is the solution on a grid of size 128 by 32 grid, shown In Figure 4.6. The drag coefficient here is .0002. The drag coefficient is converging like h^2 to its expected value of zero. Figure 4.7 shows a refined grid solution based on a 32 by 8 underlying coarse grid, with refined grid patches as shown in Figure 4.8. The drag coefficient in this case is .0009. Figure 4.9 shows a refined grid solution based on a 64 by 16 under lying coarse grid. The refined grid for this ease is shown in Figure 4.10. The drag coefficient is reduced to .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.

The second test case is transonic flow containing a shock wave. Figure 4.11 shows the pressure coefficient for a NACA 0012 airfoil at Mach .8 with zero degrees angle of attack. The mesh used for this computation is 64 by 16 cells. When the grid is refined (using an error tolerance of .005), as shown in Figure 4.12, the solution obtained is almost identical to the solution computed on a 128 by 32 mesh (compare Figures 4.13 and 4.14). In the coarse grid run, the entropy behind the shock was computed to be .0072, In the multiple grid run it was .0052, and in the fine grid run the entropy was .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 half the cost of the 128 by 32 grid run. If the error tolerance for mesh refinement is less stringent (.025), so that only the leading and trailing edges are refined (Figure 4.15), the solution is only slightly worse across the shock is .0046 in this case. In this run only 10% of the coarse grid is refined, and so the overall cost of the solution is roughly 35% of the fine grid cost.

5. References

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

[2] M.M Rai and D. Anderson, "The Use of Adaptive Grids in Conjunction with Shock-Capturing Method", AIAA Paper 81-1012. Presented at the AIAA 5th Computational Fluid Dynamics Conference, Palo, Alto, California, June 1981.

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

[4] W. Usab, Jr. and Earl M. Murman, "Embedded Mesh Solutions of the Euler Equation Using a Multiple- grid Method", AIAA Paper 83-1946-CP. Presented at the 6th AIAA Computational Fluid Dynamics Conferenocs, Danvers, Mass. July, 1983.

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

[6] M. Berger arid J. Oliger, Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, To appear in J. Comp. Phys.

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

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

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

[10] A. Jameson, "Iterative Solution of Transonic Flows over Airfoils and Wings, Including Flows at Mach 1", Comm Pure Appl. Math. XXVII (1974), 283-309.

[11] S. Davis, “A Rotationally Biased Upwind Difference Scheme for the Euler Equations”. ICASE Technical Memo 112179, July, 1983.

[12] A. Jameson, “Steady-State Solution of the Euler Equations for Transonic Flow”, Transonic, Shock and Multidimensional Flows Advances in Scientific Computing, Academic Press, 1982.

[13] M. Berger, On Conservation at Grid Interfaces. In preparation.