AIAA-89-1930

An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries

M. Berger
Courant Institute of Mathematical Sciences
New York, NY
R. LeVeque
University of Washington
Seattle, WA

AIAA 9th Computational Fluid Dynamics Conference
Buffalo, New York / June 13–15, 1989

An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries

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

Randall J. LeVeque
Mathematics Department
University of Washington
Seattle, WA 98195

Abstract

We present a Cartesian mesh algorithm with adaptive refinement to compute flows around arbitrary geometries. Cartesian meshes have been less popular than unstructured or body-fitted meshes because of several technical difficulties. We present an approach that resolves many of these problems. Cartesian meshes have the advantage of allowing the use of high resolution methods that are difficult to develop on unstructured grids. They also allow for efficient implementation on vector computers without using gather-scatter operations except at boundary cells. Some preliminary computational results using lower order boundary conditions are presented.

1. Introduction

The construction of logically rectangular body-fitted grids for complicated geometries is notoriously difficult. One alternative is to use an unstructured mesh, so that the cell volumes are not derived by a smooth mapping from a rectangular domain, as in [16] for example. Another possibility is to simply use a Cartesian grid over the entire flowfield. This introduces the difficulty of imposing solid wall boundary conditions on a grid that is not aligned with the body.

Nonetheless, there are several reasons to prefer the Cartesian grid approach, in addition to the ease of grid generation. First, it allows the use of higher order accurate shock capturing methods that are difficult to achieve on an unstructured mesh with no coordinate directions. A Cartesian grid integrator is highly vectorizable. Gather-scatter type vector operations need to be performed only in a lower dimensional region, and not over the entire flowfield. The basic solver on a Cartesian mesh is also simpler than on a body-fitted mesh, since there are no metric terms. Finally, there is some evidence [20] that for strong shock calculations an unstructured mesh has larger phase errors, and thus poorer shock-capturing abilities, than structured grids.

There are also several difficulties in using Cartesian grids. The main difficulty is the small cell problem. Arbitrarily small cells arise at the edge of the domain where the grid intersects a body. Stable, accurate and conservative difference schemes are needed for these cells. We would like the time step for a time-accurate computation to be based on the cell volume of the regular cells away from the body, and not be restricted by small cells at the boundary. The time step appropriate for the regular grid cells can give a Courant number that is orders of magnitude larger than 1 for the smaller, irregular cells. This will lead to stability problems with standard explicit methods. In this work, we use an approach based on wave propagation that essentially increases the size of the stencil near these small cells and maintains stability for arbitrary time steps. This large time step approach was studied in one space dimension in [13], and has been applied in the present context of small boundary cells in [14,15].

Another problem with Cartesian grids is one of accuracy. Grid stretching, used in body-fitted grids to cluster the grid points in regions where they are needed, cannot be used. Moreover, since the grid is not aligned with the boundary, a loss of resolution may occur near the boundary. To improve the accuracy we use an adaptive mesh refinement algorithm developed in [8]. Rectangular refined grids are superimposed on the coarse grid, so that the efficiency of the integrator on each grid is maintained. The time step is refined along with the mesh width on the fine grids, so that the CFL condition is maintained while allowing larger time steps on the coarser grids. This further concentrates the computational work where it is needed.

Cartesian mesh methods have received increased attention recently. Cartesian grids were used in [19] to solve the full potential equations. This was extended to the Euler equations in two space dimensions in [11], and to three dimensions in [12]. Cartesian grids have also been used in conjunction with an implicit, flux-vector split method for the Euler equations [10]. These calculations, however, suffer from the lack of resolution of a Cartesian mesh; none of the calculations used a local mesh refinement algorithm. The use of a global, tensor product grid to concentrate grid lines near the leading edge of an airfoil, for example, can be very wasteful. In those computations, the small, irregular cells near the body were merged into their neighboring cells to create a cell that was large enough to satisfy a stability constraint. This procedure loses resolution.

In the next section, we describe our overall computational method, including the organization of the calculation and the boundary representation. Section 3 describes the large time step method and the implementation of the solid wall boundary conditions. Section 4 describes the mesh refinement algorithm. In Section 5 we show applications of our method to the inviscid Euler equations in two geometries: shocked flow around two cylinders, and a curved channel calculation.

2. Algorithm and Data Structures

We consider the Euler equations in two space dimensions,

$$U_t + F(U)_x + G(U)_y = 0,$$

where

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \quad F(U) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uE + up \end{bmatrix}, \quad G(U) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vE + vp \end{bmatrix}.$$

and

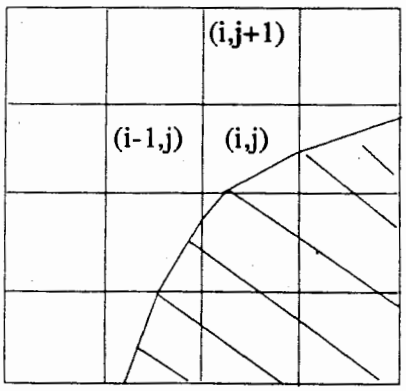
$$p = (\gamma - 1)(\rho E - \rho(u^2 + v^2)/2)).$$

The boundary of a solid object is approximated by piecewise linear segments at shown in Figure 1a. We assume that the boundary cuts through each cell at most once, so that the resulting irregular cell is a polygon with at most five sides. We use a finite volume method, with $U_{i,j}$ representing the cell average of the vector of conserved quantities in cell (i,j), i.e.

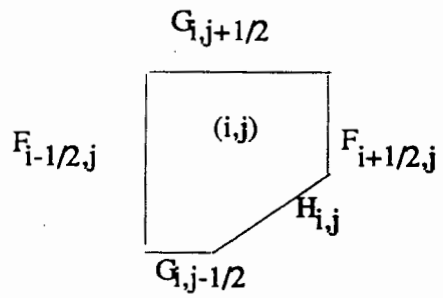
$$U^n_{i,j} \approx \frac{1}{A_{i,j}} \int_{cell(i,j)} u(x,y,t_n)\, dx\, dy,$$

where $A_{i,j}$ is the area of the cell. For regular cells, away from the boundary, $A_{i,j} = \Delta x \Delta y$, but $A_{i,j}$ may be orders of magnitude smaller for cells on the boundary.

All grid cells are indexed using the rectangular Cartesian structure. Additional information about the irregular cells is kept in a linked list data structure that is easily traversed in implementing the boundary conditions. Necessary information includes the cell area and list of vertices for each irregular cell, as well as a pointer to its location in the Cartesian grid. In the other direction, a two dimensional integer array indicates whether a Cartesian cell is regular, and for irregular cells contains a pointer to the corresponding location in the linked list. When grid refinement is used there may be several rectangular grids, each with its own solution storage and irregular points list.



(a)



(b)

Figure 1 (a) a Cartesian grid. The shaded region represents a solid body. (b) Blowup of cell (i,j) showing the fluxes.

In each time step, the cell values are updated by differencing fluxes at the cell sides, as illustrated in Figure 1b. The updating formula is

$$U^{n+1}_{i,j} = U^n_{i,j} - \frac{\Delta t}{A_{i,j}} [F_{i+1/2,j} - F_{i-1/2,j} + G_{i,j+1/2} - G_{i,j-1/2} + H_{i,j}]. \qquad (1)$$

boundary of the cell. For example, with Godunov's

method we would obtain $F_{i+1/2,j}$ by solving a Riemann problem $u_t + f(u)_x = 0$ with left and right states $U_{i,j}$ and $U_{i+1,j}$ to determine the correct intermediate state u^*. We then set $F_{i+1/2,j} = h_{i+1/2,j} f(u^*)$, where $h_{i+1/2,j}$ is the length of the interface between cells (i,j) and $(i+1,j)$. For regular cells this length is just Δy. Similarly, $G_{i,j+1/2}$ is the flux per unit time through the top of the cell. Finally, $H_{i,j}$ is the flux per unit time through the irregular side of the cell, which represents the solid wall boundary of the fluid domain. In regular cells, $H_{i,j} = 0$. With higher order Godunov schemes such as MUSCL, the left and right states are modified using slope information to achieve second order accuracy.

In irregular cells there are two difficulties with this approach. First, the neighboring cell values needed to define appropriate slopes may not be present. This is currently handled by setting the slopes to zero, so that the flux reduces to the Godunov flux at these interfaces. Improvements to this algorithm are currently under study.

Even with first order fluxes, there is still a stability problem. Use of these fluxes in updating formula (1) will give instabilities in cells where $A_{i,j}$ is very small relative to Δt. A wave propagation interpretation of this is given in Section 3, where we present a way to modify the fluxes to account for the reflection of waves at the boundary and obtain a much more stable algorithm. First we present an outline of the overall algorithm.

Step 1: Initial flux computation. In the first pass, fluxes at all cell boundaries are computed assuming that the grid is regular, even at interfaces where both neighboring cells lie outside of the actual fluid domain. This is done for ease of vectorization, but the calculations outside the domain will have no influence on the final solution. Also, the interface lengths are always assumed to be Δx or Δy in computing the flux per unit time, regardless of the true length of the side. This will be corrected in step 2 as required.

Step 2: Flux Modification Near the Boundary. In the second step, we march around the solid boundary of the fluid domain, modifying the fluxes of the irregular cells. First we adjust the fluxes F and G at each interface to incorporate the correct length rather than the standard length Δx or Δy. Next, we modify the fluxes to improve the stability of the small cells and incorporate the solid wall boundary conditions. This will be described in more detail in the next section. Finally, we calculate the fluxes $H_{i,j}$ at the irregular side of each boundary cell. This is also described in section 3.

Step 3: Updating $U_{i,j}$. The grid values $U_{i,j}$ are now updated using the flux differencing formula (1).

Step 4: Smoothing at the Boundary. Although the flux modification of Step 2 is intended to give stability for arbitrarily small cells, in practice very small cells still cause difficulties in some computations. The exact causes of this are currently under study. In the present code, stability is restored by means of an averaging process. In very small cells (those with area less than 3% of the regular cell size, typically), the value of $U_{i,j}^{n+1}$ is replaced by a weighted average of the original value and the value in one or more neighboring cells. The choice of cells depends on the local geometry. The value in the neighboring cell(s) is also modified in such a way that conservation is maintained. The weights used are proportional to the cell areas and hence the value in the small cell is replaced by a value that is essentially equal to that of the cell's primary neighbor (or a weighted averaged of two or three neighboring cells). This procedure has been found to eliminate any remaining instabilities. This algorithm is similar to the flux redistribution algorithm in [9].

3. Boundary Conditions and Flux Modification

For irregular cells at the boundary, the fluxes that are calculated in the first stage of the algorithm are simply the Godunov fluxes obtained by solving the Riemann problem between this cell and each neighbor. In order to understand how these fluxes should be modified for small cells, we first consider Godunov's method on a regular grid cell. The method can be interpreted in the following way: Solve the Riemann problem at each interface to obtain waves propagating away from the interface. For each wave that propagates into the cell, let ΔU represent the jump in the conserved quantities across the wave. Suppose that in time Δt this wave sweeps through a certain fraction α of the cell. Then the cell average $U_{i,j}$ is updated by the quantity $\alpha\Delta U$. Note that the CFL condition requires $\alpha \le 1$, and that α is the ratio of the area swept out by the wave to the total cell area (see Figure 2). The wave shown in Figure 2 propagates with speed $s > 0$ from the left side of the cell, and so

$$\alpha = \frac{s\Delta t \Delta y}{A_{i,j}} = \frac{s\Delta t}{\Delta x}.$$

Now consider the same wave but suppose that the cell in question is an irregular cell as shown in Figure 3a. We now have $\alpha = s\Delta t / A_{i,j} > 1$, and updating $U_{i,j}$ by $\alpha\,\Delta U$ would lead to instability. However, an alternative approach that is physically more reasonable is to update $U_{i,j}$ by only $1.0 \cdot \Delta U$, since the wave overlaps the entire cell, and then to update cells further to the right by the remainder of the wave $(\alpha - 1)\Delta U$. This is the basis of the large time step method originally described in [13]. In the present context however, there are no cells to the right. Instead, there is a solid wall boundary, off which the wave should reflect. This is illustrated in Figure 3b. The portion of the wave that lies outside the domain is

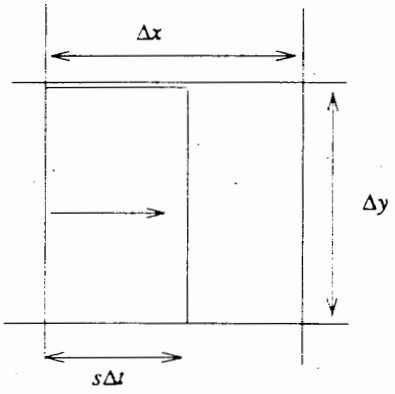


Figure 2 The wave containing the jump in conserved quantities travels to the right a distance $s\Delta t$ away from the interface.

reflected normal to the boundary segment of this cell. Linearization of the solid wall boundary conditions suggests that the reflected wave should carry a jump $\Delta\tilde{U}$, which has the same jump as ΔU in density, pressure and tangential velocity, but which has the jump in normal velocity negated. (Normal and tangential refer to the orientation of the solid wall boundary).

This reflected wave overlaps some fraction $\beta \leq 1$ of the cell, and so $U_{i,j}$ is further updated by $\beta\Delta\tilde{U}$. In addition, the reflected wave may overlap neighboring cells, and each of these is also updated by the fraction of the cell overlapped multiplied by $\Delta\tilde{U}$. In Figure 3b, three neighboring cells are affected by the reflected wave. This is the maximum number possible, so the amount of computational work required is bounded.

As just described, the waves are used to update cell values directly. In the actual implementation, the waves are used to update the fluxes at the cell interfaces by calculating the flux through each interface due to the wave. This makes these boundary conditions easier to use in conjunction with an arbitrary flux differencing method away from the boundaries. The flux at each interface is modified by any wave that crosses the interface. For example, the reflected wave in Figure 3b crosses four interfaces and would modify the flux at each of these interfaces.

Finally, we must calculate the flux $H_{i,j}$ at the solid wall boundary itself. The basic flux is computed by solving a Riemann problem at the wall with data given by $U_{i,j}$ and $\tilde{U}_{i,j}$. The vector $\tilde{U}_{i,j}$ agrees with $U_{i,j}$ in density, pressure and tangential velocity but again has the normal velocity negated. This flux is then modified by any wave that reflects off the wall, once by the outgoing flux of the wave ΔU and then again by the incoming flux of the reflected wave.

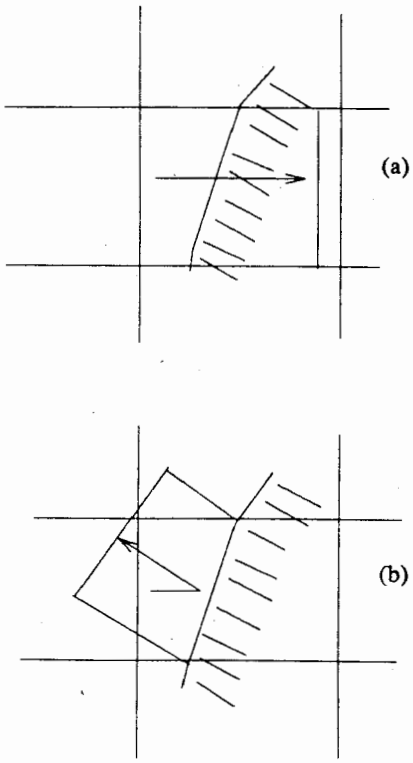


Figure 3 (a) The wave completely sweeps through the small boundary cell. (b) The wave reflects off the boundary back into the domain.

4. Mesh Refinement

The adaptive mesh refinement algorithm (henceforth AMR) is based on the use of uniform, local grid refinements superimposed on an underlying coarse grid. These embedded grid refinements can be recursively nested to maintain a fixed level of accuracy in the calculation. Unlike other embedded grid refinement methods, (e.g. [18]), in this method the grid cells requiring refinement in each level are grouped together into rectangular blocks which are uniformly refined. This means that some coarse grid cells may be unnecessarily refined, but has the advantage that all grids are uniform and rectangular. This allows us to maintain vectorization without using gather/scatter operations. It also allows for a simple user interface, since a finite difference scheme can be written for a uniform rectangular grid without concern for the connectivity of each cell. The use of fine grids instead of unstructured grid points also reduces the storage overhead, which is on a per grid basis for our method, rather than the overhead per cell found in unstructured mesh calculations. The additional complications introduced by this approach occur at the interfaces of the fine and coarse grids (see below).

In addition to refining the spatial grid for time-accurate computations we use a smaller time step on the fine grids as well. This keeps the mesh ratio of time step to space step the same on all grids, and so the same explicit finite difference scheme is stable on all grids. The computational work is thus further concentrated on the fine grids, where it should be. In contrast, some adaptive methods for transient flows use the same time step for the whole mesh [16,17]. This can be less efficient, since the resulting Courant number may be far smaller than necessary over the unrefined portion of the grid.

AMR uses an automatic error estimation procedure, based on Richardson extrapolation, to determine the regions in the domain where the resolution in the solution is insufficient. These coarse grid cells are "flagged" as needing refinement. In addition, the irregular grid cells at solid bodies should be flagged as needing refinement if the geometry of the boundary is under-resolved. An automatic grid generation algorithm groups these flagged cells into rectangular grid patches. We have developed heuristic procedures that are quite successful at this type of grid generation [4]. We try to balance the conflicting goals of minimizing the number of fine grids and minimizing the area that is unnecessarily refined.

The time accurate integration algorithm proceeds by taking one step on the coarsest grid, and as many steps as necessary on the finer level grids until they catch up to the coarse grid time. If there are several levels of fine grids, this is applied recursively. At this point the grids are advanced independently of each other, except that fine grids require boundary values from adjacent fine grids or interpolated from the coarser grids. For a five point stencil, a fine grid will need 2 points all around the outside of the grid in order to advance the solution one step. If there is an adjacent fine grid, it can supply the missing points. Otherwise, these so-called dummy points are obtained using bilinear interpolation in space and linear interpolation in time from the coarse grid.

Since we will be computing discontinuous solutions of hyperbolic conservation laws, the adaptive mesh refinement algorithm needs to be conservative. This is complicated by the use of different time steps on the different grids. Conservation is ensured in three different parts of the mesh refinement algorithm. When two adjacent levels of grids are at same time, the fine grid updates the coarse grid, performing the conservative averaging procedure

$$U^{coarse}_{i,j} = \sum_{m=0}^{r-1} \sum_{n=0}^{r-1} U^{fine}_{k+m,l+n},$$

where r is the mesh refinement ratio, for each coarse cell (i,j) containing a fine grid cell (k,l) in the lower left corner. If a fine grid is then removed, the total mass in the domain is conserved. Secondly, after every integration step the solution is post-processed at all coarse grid points adjacent to a fine grid. The initial coarse flux (computed ignoring the fine grids) is subtracted, and the sum of the fine grid fluxes over space and time is added in its place. Thirdly, we use conservative interpolation procedures to initialize the solution when a fine grid is created. A more complete description of the algorithm for time-dependent pdes is in [6].

5. Numerical Results

We illustrate the method on two time-dependent problems involving shock waves. In the first example we compute flow around two cylinders. An incident shock travels at Mach number 2.81. One cylinder is slightly ahead of the other. This leads to an interesting pattern of wave reflections between the two cylinders. In addition there is a reflected bow shock, and complicated wave structures behind the cylinders after the shocks pass by. All of these regions use the adaptive refinement as the solution develops. Figure 4 shows the incident shock with the location of the refined grids indicated. The initial coarse grid is 64 by 64. Two levels of grids are used, with a refinement factor of 4. Figure 5 shows density contours of the solution at later stages of the simulation.

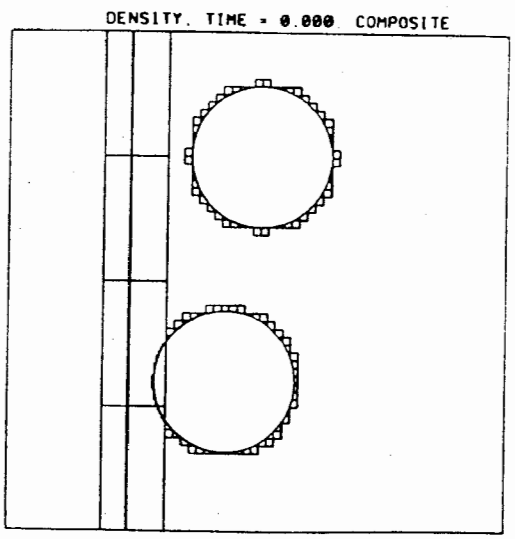
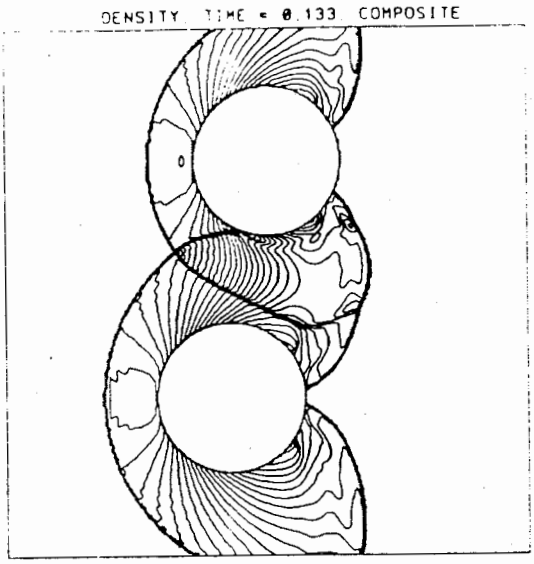


Figure 4 Density contours and grid locations at initial time for shock impinging on two cylinders.

The second example we consider is a Mach 2.2 shock travelling in a channel with a 90 degree bend. This has previously been studied in [1,21]. We use a very coarse initial grid, since much of the initial rec-



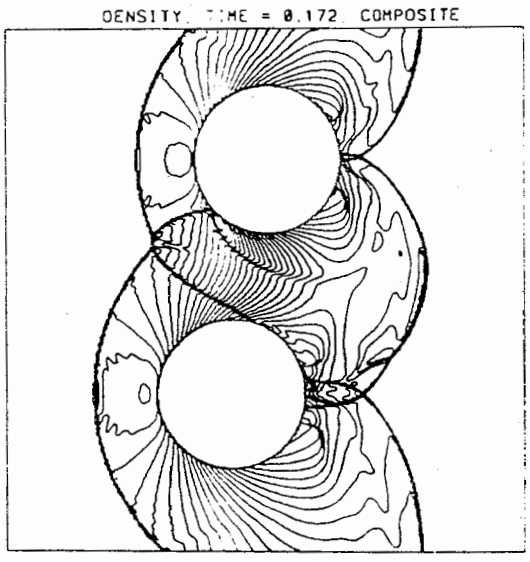


Figure 5 Density contours at later times.

tangular grid is outside the computational domain. We use two additional levels of refinement by a factor of 4 in each case in order to obtain good resolution of the shock and induced wave pattern. Figure 6 shows density contours when the shock has passed most of the way through the channel. In this figure, the location of the three levels of refined grids is indicated on the contour plots.

6. Conclusions

This work demonstrates the feasibility of an adaptive Cartesian grid approach for fluid problems in complicated geometries. Several aspects of this approach are still under development. We would like to improve the boundary scheme to make it second order along with the

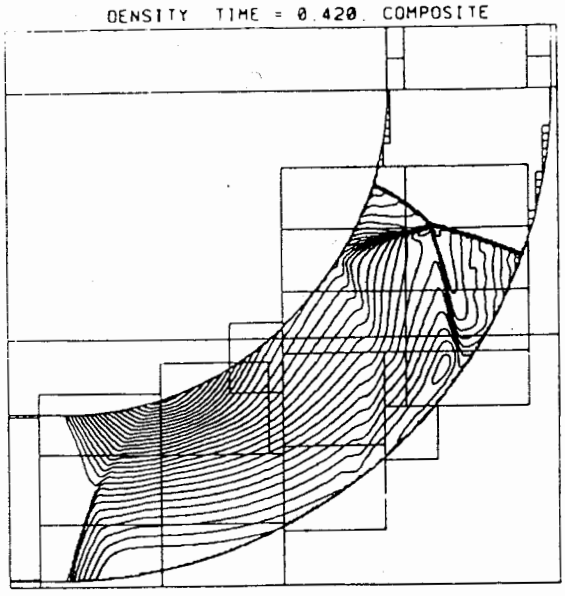


Figure 6 Density contours for shock in a channel with 90 degree bend.

interior scheme. We also hope to eliminate the smoothing step now used to insure stability in the very small cells.

We also plan to include an acceleration procedure for steady state calculations. For nested Cartesian grids multigrid is particularly attractive, since the data structures and grid transfer operations in the adaptive grid refinement algorithm make up almost all of those needed in multigrid [5]. Local grid refinement and multigrid have already been combined using a logically rectangular body-fitted grid in [7].

An important consideration is whether these techniques will extend to Navier-Stokes calculations. For problems with boundary layers, it is usually desirable to use body-fitted grids and refine heavily in the direction normal to the boundary. Refining Cartesian grid cells near such a boundary may be highly inefficient. In such cases a component grid approach may be useful, in which there are several grids with distinct coordinate systems. For example, there may be a thin body-fitted boundary layer region in addition to an underlying Cartesian grid. These multiple components will overlap in an arbitrary way, creating small irregular cells as in the Cartesian mesh method above. Again, stable and conservative difference equations are needed to compute the flow at these mesh junctions. The techniques used here should be directly applicable to this situation. Such an approach has previously been considered by others, e.g. [2,3]. However, these previous efforts did not treat the interface conditions between the different grids in an accurate or conservative way.

Acknowledgements

We thank Jeff Saltzman for the use of his high order Godunov integrator. The first author's work was supported in part by DOE Contract No. DEAC0276ER03077-V, by the AFOSR under Contract No. 86-0148. The authors were supported by NSF Presidential Young Investigator Awards ASC-8858101 and DMS-8657319.

6. References

[1] T. Aki, "A Numerical Study of Shock Propagation in Channels with 90 Degree Bends", National Aerospace Laboratory Technical Report, Tokyo, Japan, 1987.

[2] E. Atta, "Component-Adaptive Grid Interfacing", AIAA Paper No. 81-0382. Proc. AIAA 19th Aerospace Sciences Meeting, 1981.

[3] J. Benek, J. Steger and F. Dougherty, "A Flexible Grid Embedding Technique with Application to the Euler Equations", AIAA Paper No. 83-1944. Proc. 6th Computational Fluid Dynamics Conference, Danvers, Mass. July, 1983.

[4] M.J.Berger, "Data Structures for Adaptive Grid Generation", SIAM J. Sci. Stat. Comp. 7, (1986), pp. 904-916.

[5] M.J.Berger, "Adaptive Finite Difference Methods in Fluid Dynamics". Lecture series 1987-04 in Computational Fluid Dynamics at the von Karman Institute for Fluid Dynamics, Rhode Saint Genese, Belgium, March, 1987.

[6] M. Berger and P. Colella, "Local Adaptive Mesh Refinement for Shock Hydrodynamics". To appear in J. Comp. Phys.

[7] M.J. Berger and A. Jameson, "Automatic Adaptive Grid Refinement for the Euler Equations", AIAA J. 23, (1985), pp. 561-568.

[8] M. Berger and J. Oliger, "Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations", J. Comp. Phys. 53 (1984), pp. 484-512.

[9] I. Chern and P. Colella. "A Conservative Front Tracking Method for Hyperbolic Conservation Laws". Submitted to J. Comp. Phys. UCRL-97200, July 1987.

[10] S. Choi and B. Grossman, "A Flux-Vector Split, Finite-Volume Method for Euler's Equations on Non-Mapped Grids", AIAA Paper 88-0227, Proc. 26th Aerospace Sciences Meeting, Reno, Nevada.

[11] D. Clarke, H. Hassan, and M. Salas, "Euler Calculations for Multielement Airfoils Using Cartesian Grids, AIAA Paper 85-0291, Jan. 1985.

[12] R. Gaffney, H. Hassan and M. Salas, "Euler Calculations for Wings Using Cartesian Grids", AIAA Paper 87-0356, January 1987.

[13] R.J. LeVeque , "A Large Time Step Generalization of Godunov's Method for Conservation Laws", SIAM J. Num. Anal. 22 (1985), pp. 1051-1073.

[14] R.J. LeVeque, "High Resolution Finite Volume Methods on Arbitrary Grids via Wave Propagation", J. Comp. Phys. 78 (1988), pp. 36-63.

[15] R.J. LeVeque, "Cartesian Grid Methods for Flow in Irregular Regions", in Numerical Methods for Fluid Dynamics III, K.W. Morton and M.J. Baines, editors, Clarendon Press (1988), pp. 375-382.

[16] Lohner, "Finite Elements in CFD: What Lies Ahead". Proc. World Congress on Computational Mechanics, Austin, Texas. Sept. 1986.

[17] J. Saltzman and J. Brackbill, "Applications and Generalizations of Variational Methods for Generating Adaptive Meshes", in Numerical Grid Generation, J. Thompson, ed., North-Holland, 1982.

[18] Usab and Murman, "Embedded Mesh Solutions of the Euler Equations Using a Multiple-grid Method", AIAA Paper 83-1946-CP, 6th AIAA Computational Fluid Dynamics Conference, Danvers, Mass. July 1983.

[19] B. Wedan and J. South, "A Method for Solving the Transonic Full-Potential Equations for General Configurations", Proc. AIAA Computational Fluid Dynamics Conf., July 1983.

[20] P. Woodward, Proc. Nato Workshop in Astrophysical Radiation Hydrodynamics , Munich, W. Germany, Nov. 1983. Also Lawrence Livermore Report UCRL-90009, August, 1982, (unpublished).

[21] H. Yee, "Upwind and Symmetric Shock-Capturing Schemes", NASA Technical Memorandum 89464, May 1987.