THE ADAPTIVE, CUT-CELL CARTESIAN APPROACH (WARTS AND ALL)

Kenneth G. Powell
The W. M. Keck Foundation CFD Laboratory
Department of Aerospace Engineering
The University of Michigan
Ann Arbor, MI

INTRODUCTION

Solution-adaptive methods based on cutting bodies out of Cartesian grids are gaining popularity now that the ways of circumventing the accuracy problems associated with small cut cells have been developed. Researchers are applying Cartesian-based schemes to a broad class of problems now, and, although there is still development work to be done, it is becoming clearer which problems are best suited to the approach (and which are not). The purpose of this paper is to give a candid assessment, based on applying Cartesian schemes to a variety of problems, of the strengths and weaknesses of the approach as it is currently implemented.

BASIC ELEMENTS OF THE APPROACH

In the adaptive, cut-cell Cartesian approach, as in many adaptive-grid methods, the grid-generation and flow-solution algorithms are strongly linked. The basic pieces of the grid-generation are:

- A cell-based tree data structure;
- A geometry-based adaptive refinement scheme for generating an initial grid;
- A solution-based adaptive refinement/coarsening scheme for generating the final grid.

The basic pieces of the flow-solution algorithm are:

- A limited linear reconstruction scheme;
- A flux function based on an approximate Riemann solver;
- A multi-stage time-stepping scheme.

Each of the basic pieces of the grid-generation and the flow-solution algorithms are described briefly in the following paragraphs. Additional pieces, such as a viscous-term discretization, cell-merging for moving boundary problems, and multigrid acceleration have also been implemented and used in

obtaining results presented in this paper; readers are directed to other papers for details on these techniques.

The data structure

In the work presented in this paper, a cell-based tree data structure is used. In this approach, the grid is represented as a tree in which cells in the grid correspond to nodes of the tree. The root of the tree is a large cell that covers the entire solution domain. This root cell is then divided, yielding a number of children cells which depends on the type of tree being used: two children cells for a binary-tree-based grid; four for a quadtree-based grid; eight for an octree-based grid. In the work presented in this paper, a quadtree data structure was used for the two-dimensional Euler results, a binary-tree data structure was used for the Navier-Stokes results, and an octree data structure was used for the three-dimensional Euler results.

Tree-based structures are well-suited to an adaptive Cartesian-grid approach for several reasons. One reason is that a tree-based structure is memory-efficient; connectivity information relating cells to neighboring cells is unnecessary, as this information can be inferred from the tree structure. Another reason is the ease with which the grid can be adapted: local refinement simply adds children cells to one of the nodes of the tree; local coarsening simply deletes the children cells of one of the nodes of the tree. It should be noted that cell-based trees are not the only data structure suited to the adaptive Cartesian approach. While the current work and the work of the TRANAIR group at Boeing [1] are based on tree structures, the work of Berger [2] and of Quirk [3] are based on local patches of refined grids, with each patch addressed in a structured-grid manner.

Geometry-Based Adaptive Refinement

Unquestionably the primary attraction of the cut-cell Cartesian approach arises from the quest for "hands-off" grid generation. In the geometry-based adaptive-refinement scheme used to generate the initial grids for calculations, both the body-surface discretization and the solution-domain volume discretization are carried out automatically, based on a minimum of user input. The inputs to this step are:

- A length scale for the root cell (typical value 100 chords);
- A maximum length scale for cells that intersect a body (typical value 0.01 chords);
- A maximum angle deviation between successive faces on a body (typical value 5°);
- A minimum length scale of interest (typical value 0.001 chords).

In addition, a location for the centroid of the root cell (typically the origin) is required, as is a spline representation of the bodies in the flow.

The grid-generation algorithm makes use of these input parameters by carrying out the following steps;

1. A root cell is constructed based on the input size and location;

2. The root cell is recursively refined until each body has at least one cell intersecting it;
3. Cells that intersect bodies are recursively refined until the maximum body length-scale constraint is met (or the minimum length-scale is hit);
4. Cells that intersect bodies are recursively refined until the angle-deviation constraint is met (or the minimum length-scale is hit).

This approach is robust, and simple to code, although it relies on recursion. Most of the time is spent querying the spline representations of the bodies, and computing intersections of cells in the grid with the splines. Efficient and robust coding of those intersection calculations is extremely important. Quirk [4] uses integer arithmetic to compute these intersections, which guarantees robustness and can resolve arbitrarily fine geometric details.

Solution-Based Adaptive Refinement

Solution-based adaptation is carried out by flagging cells for refinement or coarsening based on heuristic criteria. The criteria used in this work are tuned to capturing regions in which compressibility and/or vorticity are appreciable, and are scaled in the manner suggested by Warren et al [5] so as to avoid over-refining high-gradient regions at the cost of smooth regions. The criteria are

$$\tau_c = |\nabla \cdot \mathbf{u}|\, h_i^{1.5} \qquad \sigma_c = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \tau_{c_i}^2} \tag{1}$$

$$\tau_v = |\nabla \times \mathbf{u}|\, h_i^{1.5} \qquad \sigma_c = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \tau_{v_i}^2} \tag{2}$$

where n is the number of cells in the grid, and h_i is a length-scale associated with cell i. Based on these criteria, cells are flagged for refinement if

$$|\tau_c| > \sigma_c \text{ or } |\tau_v| > \sigma_v \tag{3}$$

and flagged for coarsening if

$$|\tau_c| < \frac{1}{10}\sigma_c \text{ and } |\tau_v| < \frac{1}{10}\sigma_v \,. \tag{4}$$

Limited Linear Reconstruction

In order for the scheme to be more than first-order accurate, a local reconstruction must be done; in order for the scheme to yield oscillation-free results, the reconstruction must be limited. The limited linear reconstruction here is due to Barth [6]. A least-squares gradient is calculated, using neighboring cells, by locally solving the following non-square system for the gradient of the primitive variable vector W by a least-squares approach

$$\mathcal{L}\nabla W^{(k)} = f \tag{5}$$

$$\mathcal{L} = \begin{pmatrix} \Delta x_1 & \Delta y_1 \\ \vdots & \vdots \\ \Delta x_N & \Delta y_N \end{pmatrix} \qquad f = \begin{pmatrix} \Delta W_1^{(k)} \\ \vdots \\ \Delta W_N^{(k)} \end{pmatrix} \tag{6}$$

where

$$\Delta x_i = x_i - x_0 \quad (7)$$

$$\Delta y_i = y_i - y_0 \quad (8)$$

$$\Delta u_i = u_i - u_0 \quad (9)$$

and the points are numbered so that 0 is cell in which the gradient is being calculated, and i is one of N neighboring cells used in the reconstruction. A cell-based minmod-type limiter is implemented by reconstructing the solution as

$$\mathbf{W}(\mathbf{x}) = \bar{\mathbf{W}} + \phi(\mathbf{x} - \bar{\mathbf{x}}) \cdot \nabla \mathbf{W} \quad (1)$$

where ϕ is given by

$$\phi = \min \begin{cases} 1 \\ \min_k \left(\frac{\left| \bar{W}^{(k)} - \max_{neighbors}\left(W^{(k)}\right) \right|}{\left| \bar{W}^{(k)} - \max_{cell}\left(W^{(k)}\right) \right|} \right) \\ \min_k \left(\frac{\left| \bar{W}^{(k)} - \min_{neighbors}\left(W^{(k)}\right) \right|}{\left| \bar{W}^{(k)} - \min_{cell}\left(W^{(k)}\right) \right|} \right) \end{cases} \quad (2)$$

Flux Function

The flux function used is Roe's approximate Riemann solver, described in more detail in many references, including [7]. The inputs to the flux function are the left and right states resulting from the limited reconstruction step described above.

Time-Stepping

A multi-stage scheme is used to advance the solution in time (in the unsteady calculations) or to a steady state (in the steady calculations). Local time stepping is used in the steady calculations; a cell-merging procedure is used in the unsteady calculations.

RESULTS AND OUTLOOK FOR VARIOUS CLASSES OF FLOWS

Steady, Inviscid Flows

The original drawbacks to cut-cell Cartesian approaches for solving the Euler equations were two-fold: stability and accuracy problems associated with the small cut cells; and resolution problems due to the regularity of Cartesian grids. With the introduction of local refinement and coarsening, the resolution problem for these flows is basically solved for two-dimensional flows. For three-dimensional flows, resolution problems remain, as will be described later in this section.

The accuracy and stability problems associated with the small cut cells that occur in the Cartesian approach have been the topic of several papers. Several techniques have been developed, all of which

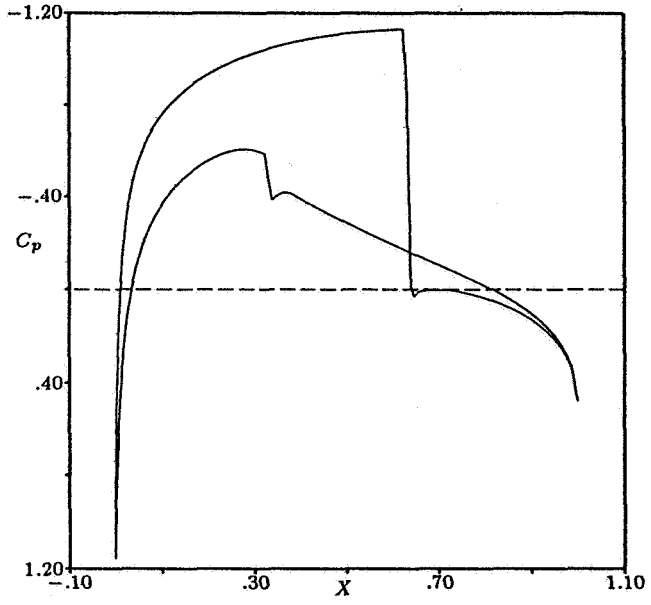

Figure 1: AGARD 01 Benchmark Case — C_p

alleviate the problems. Berger and Leveque [8] use a wave-propagation-based technique that allows the time step in small cells to be computed from a CFD criterion based on the entire uncut cell. Chern and Colella [9] use a simpler but less accurate technique based on similar concepts. DeZeeuw and Powell [7] treat the mesh with cut cells as an unstructured mesh, and use linear reconstruction to ensure accuracy and local time stepping to ensure stability in steady flows. For unsteady flows, Quirk [4] and Bayyuk, Powell and Van Leer [10] use a cell-merging technique that alleviates the small time-step problem, and also lends itself well to computing flows in which the boundaries are moving.

Results for the AGARD-1 and AGARD-3 benchmark cases are shown in Figures 1–7. The first was run on a grid of 11,366 cells; the second on a grid of 15,234 cells. As can be seen, the adaptation criteria capture the shocks and wakes, without overresolving other flow regions. The C_p on the wing for the first case, and the Mach number on the wing for the second case, show that the cut cells on the wing do not lead to non-smooth values of the flow variables there.

Results for the four-element Suddhoo-Hall benchmark case, are shown in the paper by Coirier and Powell in this volume. The comparison of computed and exact C_p show that, once sufficient adaptation has been done to resolve the flow features, the solution is smooth and matches the analytical solution well.

Results for the "Jameson non-unique" benchmark case are shown in Figures 8–11. The grid, shown in Figure 8, was used for all of the calculations; it has 13,613 cells. The free-stream Mach number was set at $M_\infty = 0.78$, and the angle of attack was varied incrementally, converging the code to machine zero residuals at each angle of attack. The hysteresis effect is shown in Figure 9; the two solutions obtained at $\alpha = -0.45°$ are shown in Figures 10 (obtained by decreasing the angle of attack incrementally) and 11 (obtained by increasing the angle of attack incrementally). While the two solutions highly resemble those originally obtained by Jameson, and the hysteresis effect is evident, the range of M and α in which the solution was non-unique was different from that reported

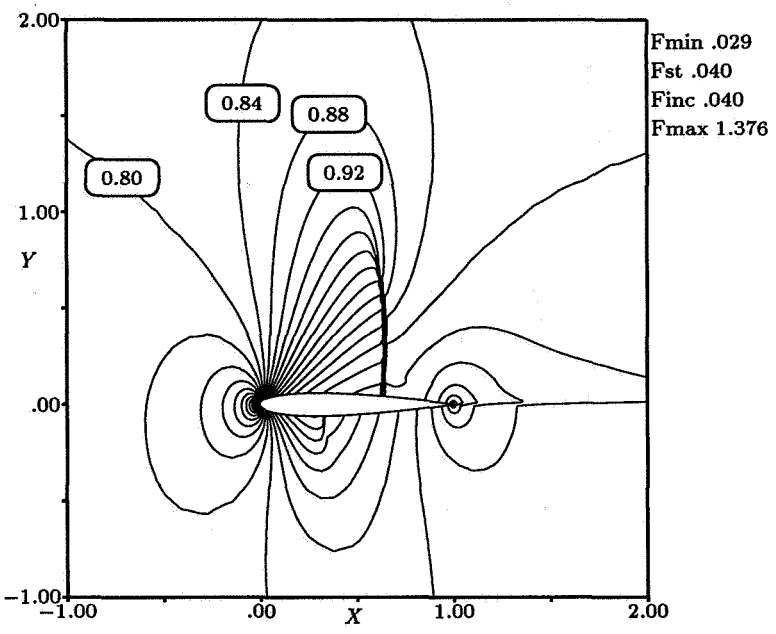

Figure 2: AGARD 01 Benchmark Case — M contours

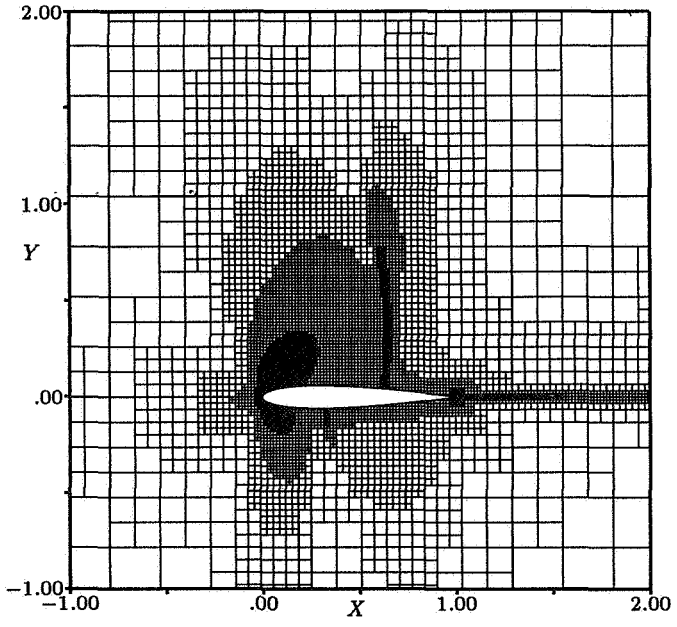

Figure 3: AGARD 01 Benchmark Case — Grid

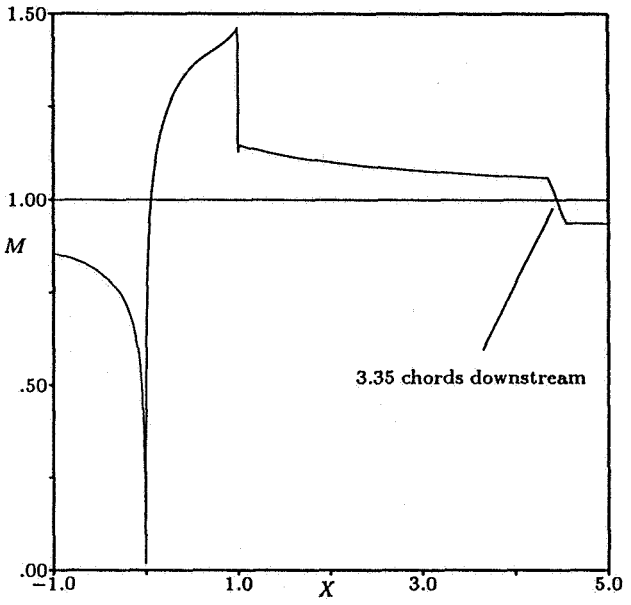

Figure 4: AGARD 03 Benchmark Case — Mach Number on axis

by Jameson. Running this case raised more questions than it answered for us. Further studies of the sensitivity of the non-unique range to computational parameters will have to be carried out before anything definitive can be said about this airfoil.

Extension of the two-dimensional scheme above to three dimensions adds complexity to the geometric algorithms (such as calculating the locations at which the cells are cut by the bodies) but adds very little to the flow solver. It is in the three-dimensional cases that the advantages of the automated grid-generation procedure are truly seen. Preliminary results for the double-ellipsoid benchmark case are shown in Figures 12–14.

The simplicity of the double-ellipsoid geometry hides an important inefficiency of the three-dimensional Cartesian-based scheme, however. For problems in which, due to the geometry, gradients are higher in one direction than another, the cost of resolving the one direction is the over-resolution of the other direction. An example is a high-$\mathcal{AR}$ wing: resolving the chordwise direction well leads to gross over-resolution of the spanwise direction, due to the isotropic nature of the grid refinement. Allowing directional refinement will in general not help; for instance, a swept-back high-$\mathcal{AR}$ wing require a large number of cells to resolve, regardless of the refinement approach used.

Memory and CPU usage for the steady-flow codes are presentied in Table 1. The times reported are for one single-grid iteration of a four-stage time-stepping scheme. The data structure used for the cut cells in the 3D code is a preliminary one; a code with lower memory overhead could be easily implemented.

With the incorporation of local refinement/coarsening into Cartesian codes, and with any of the various solutions to the small-cell problem, adaptive cut-cell Cartesian codes have reached a point where they can compete favorably with other approaches for two-dimensional steady, inviscid flows. The grid-generation is as automatic as totally unstructured approaches, and in some cases more automatic, since the surface discretization of the boundaries is carried out at the same time as the

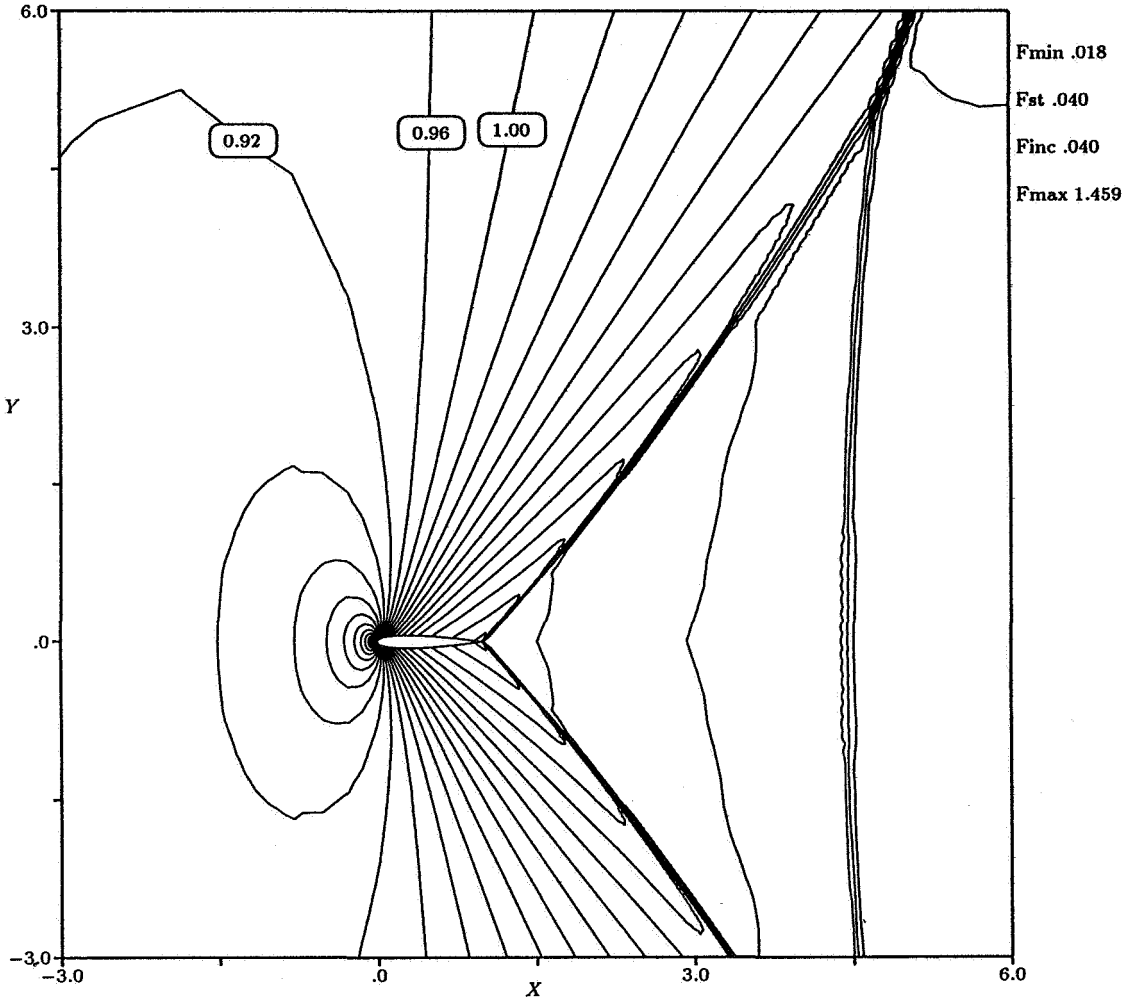

Figure 5: AGARD 03 Benchmark Case — M contours

	Memory Usage	CPU usage (μseconds/cell/iteration)
2D	(30 reals + 8 ints)*nCells + (2 reals + 3 ints)*nCutCells	370 (HP 735/99)
3D	(35 reals + 16 ints)*nCells + (38 reals + 35 ints)*nCutCells	470 (IBM RS 6000/590)

Table 1: Memory and CPU usage for Cut-Cell Cartesian Euler Codes (steady)

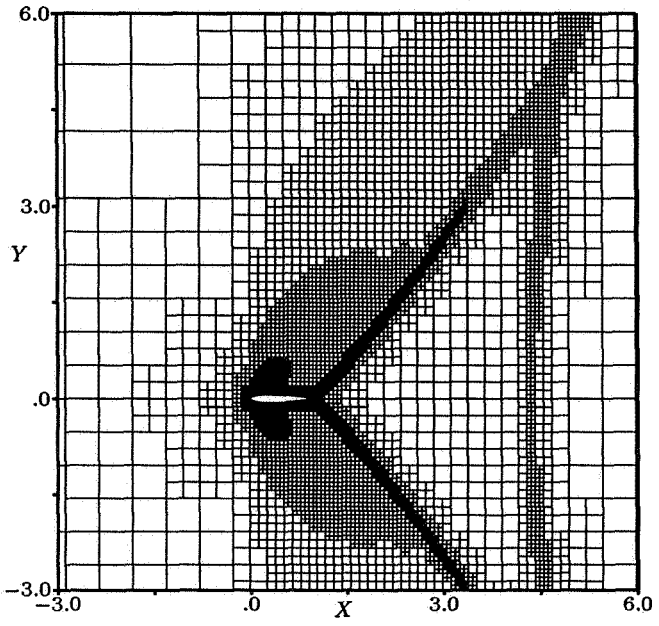

Figure 6: AGARD 03 Benchmark Case — Grid

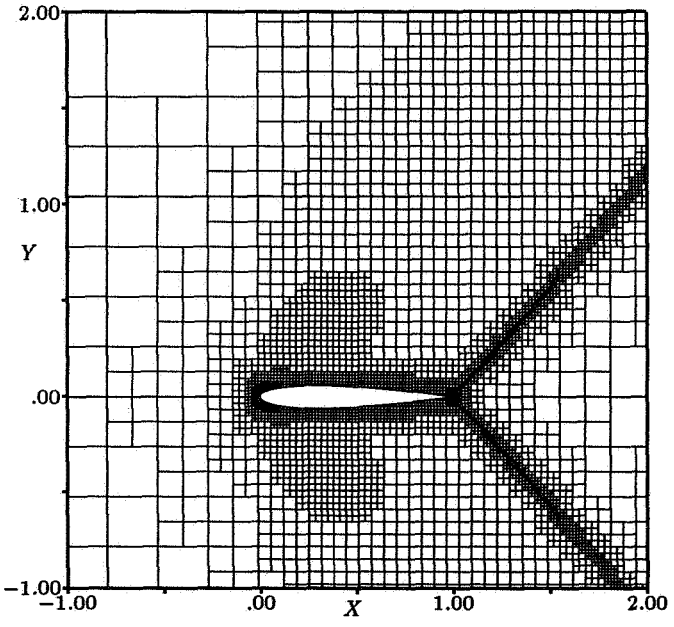

Figure 7: AGARD 03 Benchmark Case — Grid Close-up

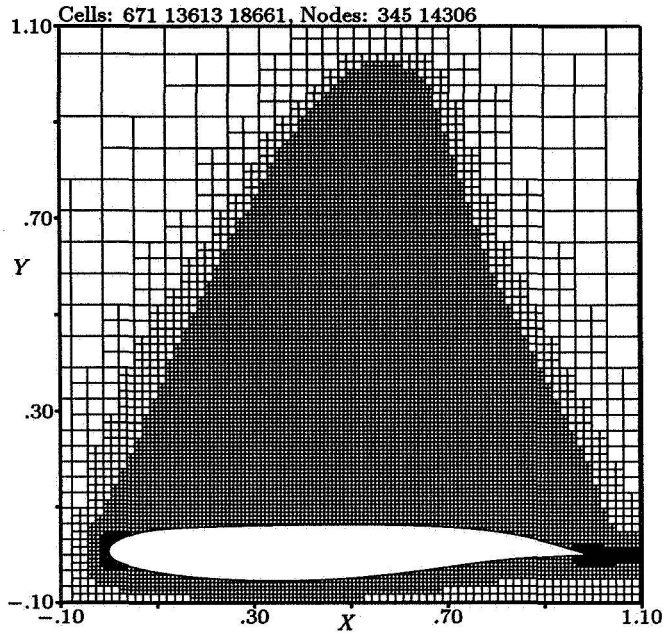

Figure 8: Jameson Airfoil Benchmark Case — Grid

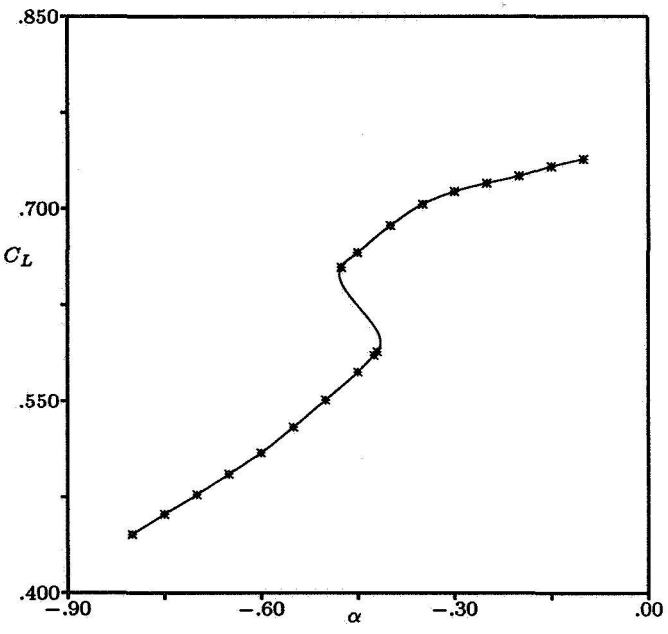

Figure 9: Jameson Airfoil Benchmark Case — C_L versus α curve

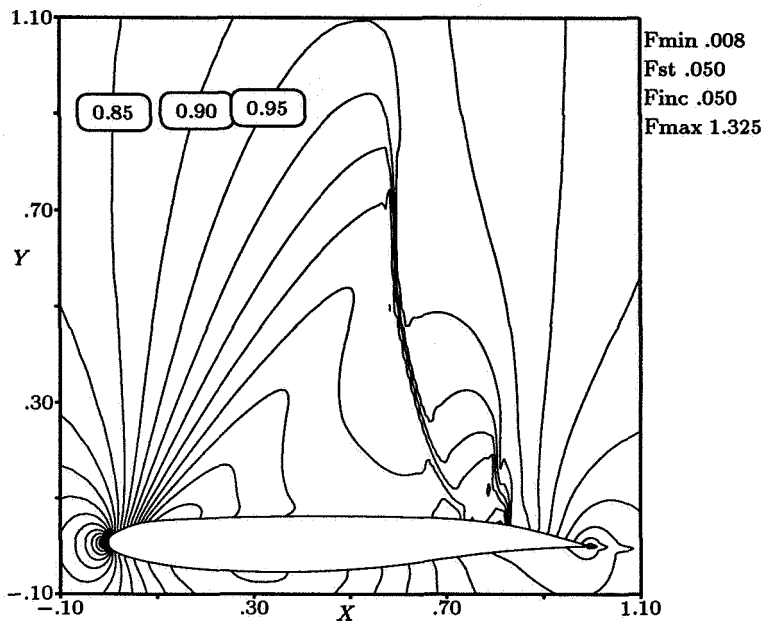

Figure 10: Jameson Airfoil Benchmark Case — Mach contours

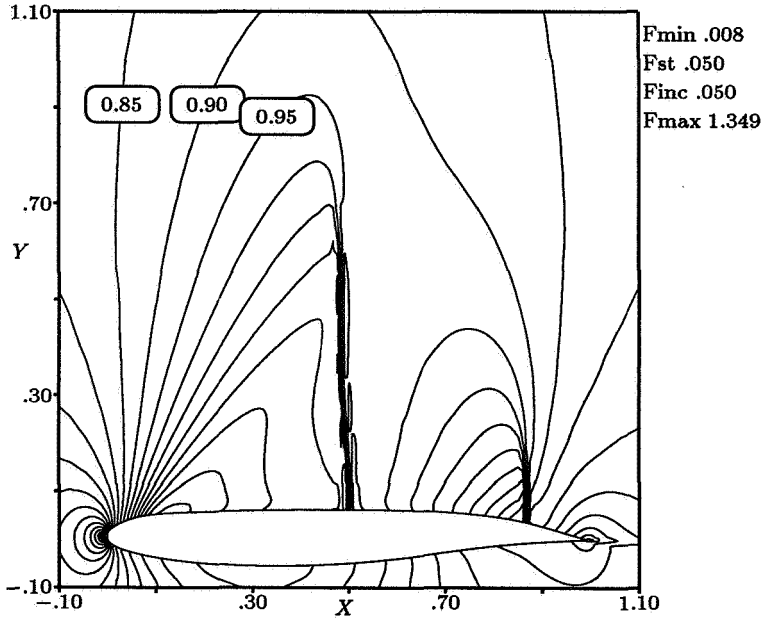

Figure 11: Jameson Airfoil Benchmark Case — Mach contours

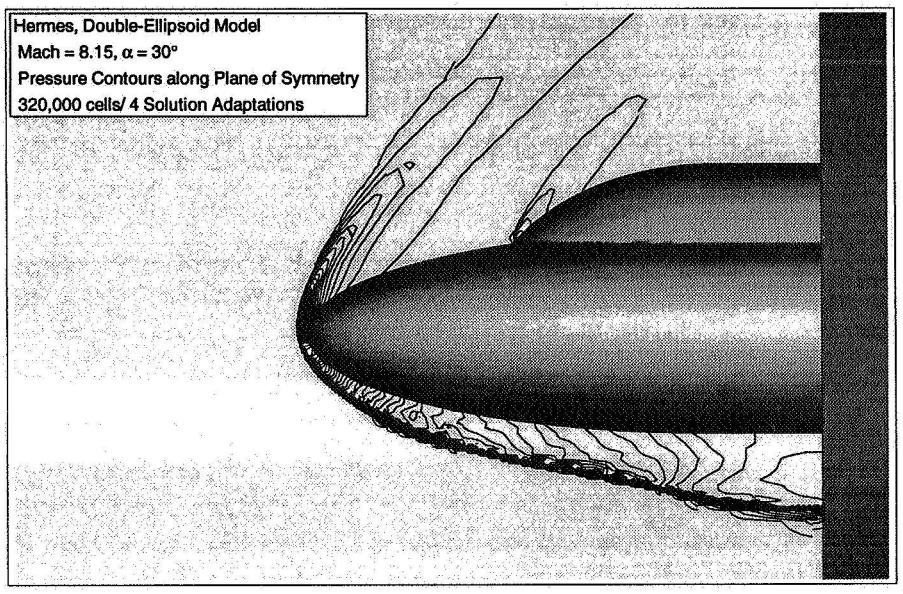

Figure 12: Solution for Hermes Problem

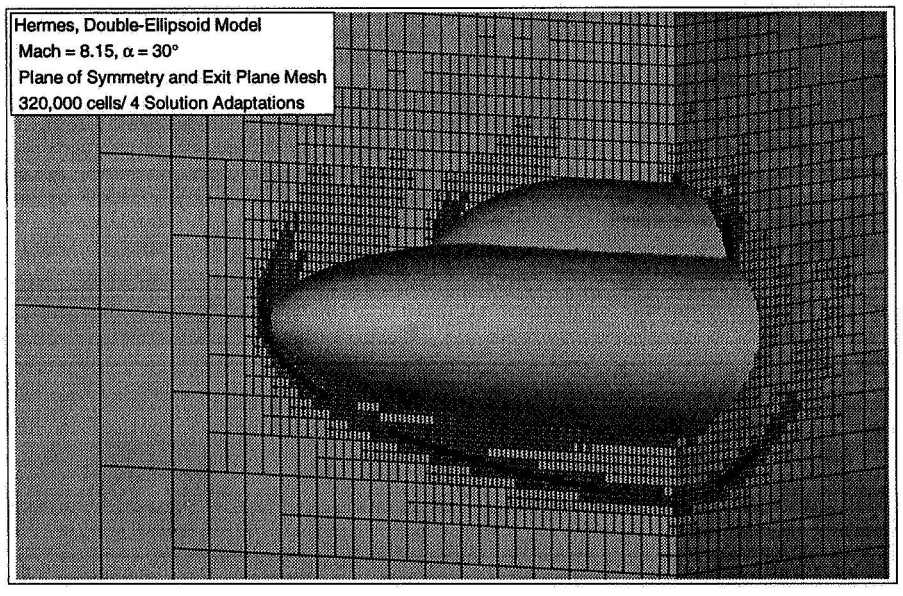

Figure 13: Grid for Hermes Problem

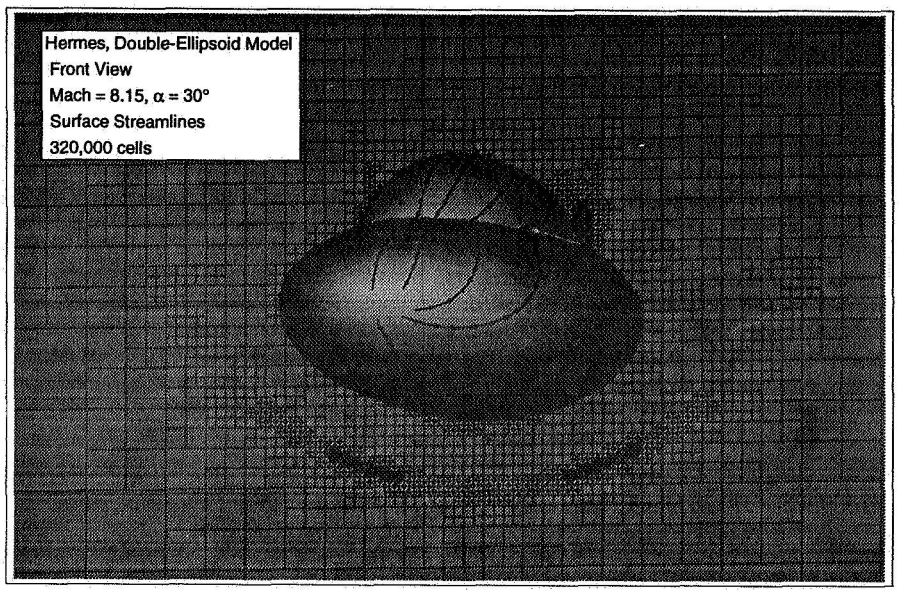

Figure 14: Streamlines for Hermes Problem

volume discretization of the solution domain. Typically, approximately 50% more cells are required to get the same accuracy out of a Cartesian scheme as on a body-conforming grid (see, for example [11]).

In three dimensions, the payoff of automated grid generation is even greater. However, since the Cartesian approach is geared towards solving isotropic problems, many geometries are difficult to resolve properly without an extremely high number of cells. This trade-off of ease of grid generation versus efficient use of computational resources is one that is almost certainly worthwhile in the early stages of a design process; very high-caliber calculations for use in detailed analysis of three-dimensional flows will probably always have to be done on body-conforming grids.

Viscous Flows

The difficulties in applying Cartesian-based schemes to viscous flows are detailed in the paper by Coirier and Powell in this volume. The two fundamental issues are:

- The difficulty in defining a viscous discretization that is positive and consistent (let alone accurate) on the very non-smooth meshes produced by the cut-cell Cartesian approach;
- The inherent inefficiency in isotropic refinment of one cell into four for highly anisotropic (e.g. high-Reynolds-number) problems.

It is interesting to note that the first problem is most acute when the Reynolds number is low; the second is most acute when the Reynods number is high. At moderate Reynolds number, a stable, reasonably accurate scheme that makes reasonably efficient use of the computational resources can be constructed. Flow quantities such as density, pressure and velocities can be obtained to good accuracy; derivative quantities such as pressure gradient and C_f can not.

Taking these issues into account, Cartesian Navier-Stokes codes will probably never play more than a very preliminary design role in low-to-moderate Reynolds number flows, and probably have no role in high-Re flows. For high Reynolds number flows, the best route is a viscous-inviscid coupling approach, in which a Cartesian Euler code is coupled to an integral or finite-difference boundary-layer code (this approach is currently under development in collaboration with Marsha Berger). Another approach, that will require much more development but seems very promising, is to rewrite the Navier-Stokes equations as a first-order hyperbolic system, and solve the resulting equations by methods similar to those used for the Euler equations [12].

Unsteady Flows

The ability of the Cartesian approach to model shock physics accurately and efficiently has been shown most impressively by Berger and Colella [13], Berger and Leveque [8] and Quirk [4, 3]. One very promising arena for Cartesian approaches is that of problems with moving boundaries. In this approach, the boundary is "cut" from the Cartesian mesh at each time step. An unsteady Euler solver is implemented, that accounts for the changing areas of the cut cells of the mesh. In order to alleviate the small time-step problem, and the problems occuring when the body covers or uncovers a cell in one time step, a merging procedure is used [4, 10].

Results from an idealized inlet case are presented in Figures 15–17. This case is a time-accurate simulation of an evolving flow pattern. The evolution is induced by a continuous and smooth deformation of boundaries resembling a supersonic inlet. The figures show the Mach number contours and the associated grids at three points during the geometric excursion. The inflow Mach number is 2.54 throughout. Bewteen the first and second positions, the inlet sides open up at the back and the front and rear edges of these sides become more tapered. The sides also move apart and the spike moves forward and contracts in the vertical direction. The effect of this change on the flow-field can clearly be seen. Between the second and third positions, the spike decreases in length and its geometry changes as shown in the figures. Also the sides flatten and they move towards each other. The final flow pattern achieves the required objective of the inlet (decelaration of the flow to approximately Mach 1.4 with minimal losses in stagnation pressure). The location and shape of the rear of the spike in the third position is critical: the impinging reflected shock must fall in a region of increasing cross-sectional area, otherwise shock system will be disgorged. The number of cells in the mesh ranges from 17,000 at the initial time to 83,000 at the final time.

Results from a case of a body expelled from a high-pressure chamber are presented in Figures 18 and 19. In this case, the two rigid bodies and the gas are all initially stationary. The gas in the enclosure of the larger body has a density and a pressure ten times those of the outside gas. The boundary separating the high-pressure gas from the low-pressure gas is initially half-way down the channel in the larger body and coincides with a plane of symmetry of the smaller body. The simulation shows the evolution of the flow and the motion of the bodies as the compressed gas flows through the channel. The motion of each body is computed by integrating the accelaration due to the net (inviscid) aerodynamic force acting on it. The grid begins with 17,000 cells; that number has increased to 175,000 by the final time.

The stretching and shearing that a body-conforming mesh would undergo during these calculations would necessitate frequent remeshing. In some sense, in the Cartesian calculation remeshing is carried out every time step; this procedure is inexpensive in the Cartesian case, however.

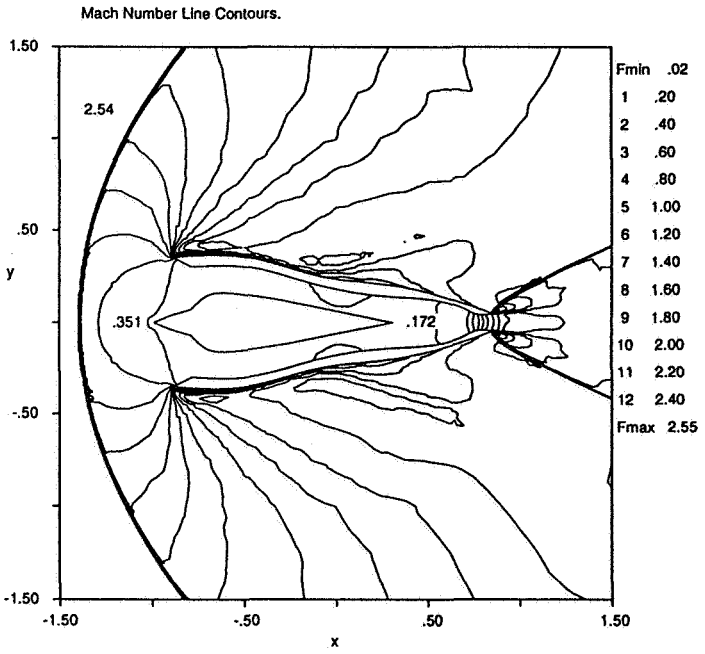

Figure 15: Mach Contours for Inlet Problem — Initial Time

	Memory Usage	CPU usage (μseconds/cell/iteration)
2D	(15 reals + 11 ints)*nCells	300 (2 stage scheme, HP 735/99)

Table 2: Memory and CPU usage for Cut-Cell Cartesian Euler Code (unsteady)

Memory and CPU usage for the unsteady code are reported in Table 2.

THE BOTTOM LINE

The siren-song of totally automated grid-generation has lured a growing number of researchers to Cartesian-based schemes.

Two-dimensional Euler solvers, both steady and unsteady, have reached a level of sophistication and maturity comparable to that of structured quadrilateral- and unstructured triangular-mesh schemes. For these problems, the Cartesian approach is competitive with the best structured and unstructured schemes. Grid generation is automated, user input to the codes is minimal, and the only penalty is that more cells must be used than in a more traditional approach.

One set of problems for which the Cartesian approach promises to show real advantages over more traditional approaches is that of inviscid flows with moving boundaries. The preliminary results shown here, and those of Quirk [4] and Pember et al [14], for moving-boundary flows suggest that

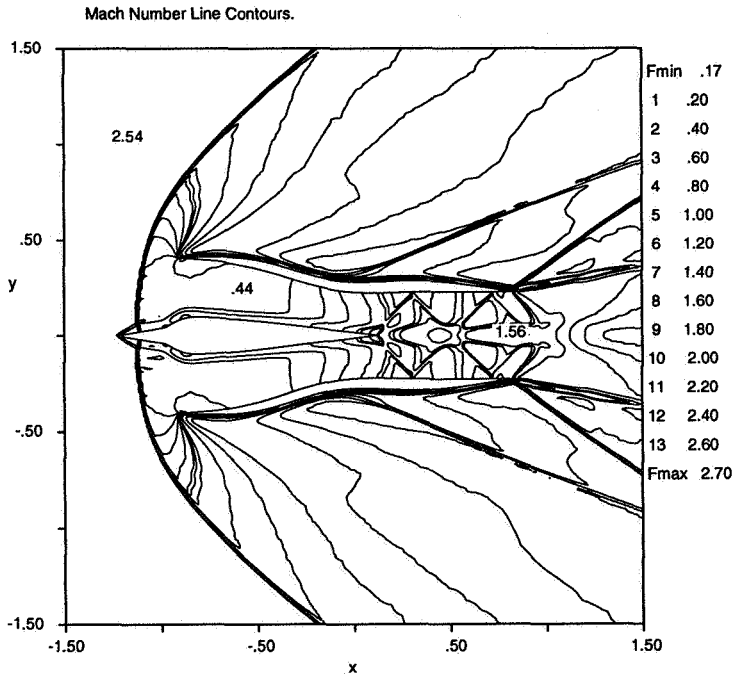

Figure 16: Mach Contours for Inlet Problem — Intermediate Time

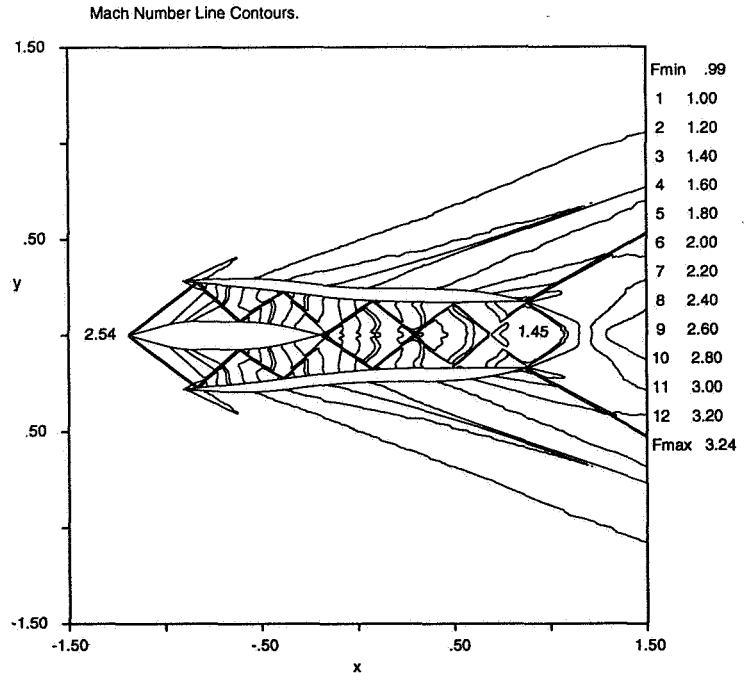

Figure 17: Mach Contours for Inlet Problem — Final Time

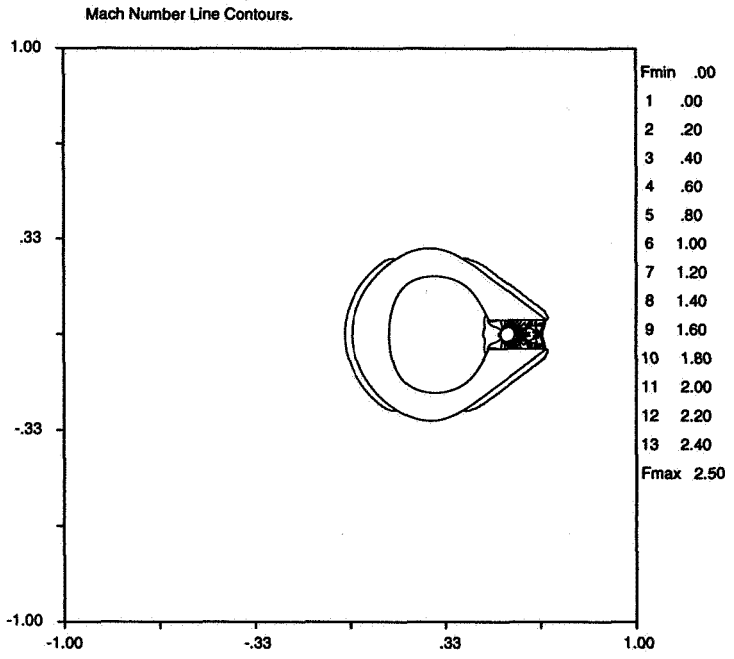

Figure 18: Mach Contours for Propulsion Problem — Initial Time

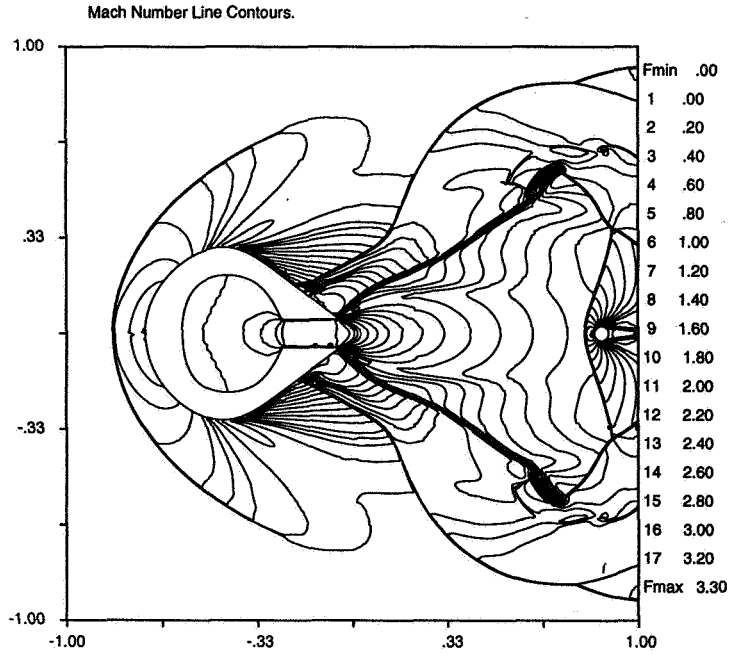

Figure 19: Mach Contours for Propulsion Problem — Final Time

this class of problems may be one for which the Cartesian approach is the best suited.

For three-dimensional inviscid flows, the automated grid generation of the Cartesian approach is even more attractive, but for one drawback. That drawback is that resolving one coordinate direction along a body typically means over-resolving another coordinate direction, for all but the simplest bodies. Judging by the popularity of Cartesian potential-flow solvers, however, for which this drawback also exists, the benefit of automating the grid generation may outweigh the inefficiency when computing geometrically complex cases.

Application of the Cartesian approach to solution of the Navier-Stokes equations is more problematic. At low Reynolds number, the difficulty in constructing an accurate, positive approximation to the viscous terms is the primary problem. At high Reynolds number, the inherent inefficiency of adapting what started out as a Cartesian mesh to non-coordinate-aligned features is the primary problem. At moderate Reynolds numbers, Cartesian Navier-Stokes codes may be useful in preliminary design, but high-caliber results will rely on a method that uses a body-conforming grid with special treatment of the boundary-layer. For high Reynolds number flows, Cartesian Euler coupled with a boundary-layer solver may be the most useful approach.

More work remains to be done. Issues of accuracy and efficiency for Cartesian-based schemes are still being resolved. However, while there are some classes of flows for which these codes will never compete favorably with more traditional approaches, it is clear that, for other classes of flows, Cartesian codes are not only competitive, but offer unique advantages.

ACKNOWLEDGMENTS

Most of the work presented in this paper was carried out by four of the author's graduate students:

- Darren DeZeeuw
- Sami Bayyuk
- Bill Coirier
- Jeff Benko

The work was funded by NASA Langley Research Center, the McDonnell Aircraft Corporation, and the National Science Foundation.

REFERENCES

[1] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, and S. S. Samant. A locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 62:1–66, 1991.

[2] M. J. Berger. Adaptive finite difference methods in fluid dynamics. Technical Report DOE/ER/0377-277, Courant Mathematics and Computing Laboratory, 1987.

[3] J. Quirk. *An Adaptive Grid Algorithm for Computational Shock Hydrodynamics.* PhD thesis, Cranfield Institute of Technology, 1991.

[4] J. J. Quirk. An alternative to unstructured grids for computing gas dynamic flows around arbitrariliy complex two-dimensional bodies. ICASE Report 92-7, 1992.

[5] G. Warren, W. K. Anderson, J. Thomas, and S. Krist. Grid convergence for adaptive methods. In *AIAA 10th Computational Fluid Dynamics Conference*, 1991.

[6] T. J. Barth. On unstructured grids and solvers. In *Computational Fluid Dynamics*. Von Kármán Institute for Fluid Dynamics, Lecture Series 1990-04, 1990.

[7] D. De Zeeuw and K. G. Powell. An adaptively-refined cartesian mesh solver for the Euler equations. *Journal of Computational Physics*, 104(1):56–68, 1993.

[8] M. J. Berger and R. J. LeVeque. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. In *AIAA 9th Computational Fluid Dynamics Conference*, 1989.

[9] I. L. Chern and P. Colella. A conservative front tracking method for hyperbolic conservation laws. Technical Report UCRL-97200, Lawrence Livermore National Laboratory, 1987.

[10] S. Bayyuk, K. G. Powell, and B. van Leer. A simulation technique for two-dimensional unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry. AIAA Paper 93-3391-CP, 1993.

[11] W. J. Coirier and K. G. Powell. An accuracy assessment of Cartesian-mesh approaches for the Euler equations. Aiaa paper, 1995.

[12] T. I. Gombosi, C. P. T. Groth, P. L. Roe, and S. L. Brown. 35-moment closure for rarefied gases: Derivation, transport equations, and wave structure. submitted to *Physics of Fluids*, 1994.

[13] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. Technical Report UCRL-97196, Lawrence Livermore National Laboratory, 1987.

[14] R. B. Pember, J. G. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome. Adaptive cartesian grid methods for representing geometry in inviscid compressible flow. In *AIAA 11th Computational Fluid Dynamics Conference*, 1993.