AIAA Meeting Papers on Disc, January 1997
A9715279, F49620-94-1-0399, W-7405-Eng-82, AIAA Paper 97-0198

An octree solution to conservation laws over arbitrary regions (OSCAR)

Eric F. Charlton
Michigan Univ., Ann Arbor

Kenneth G. Powell
Michigan Univ., Ann Arbor

AIAA, Aerospace Sciences Meeting & Exhibit, 35th, Reno, NV, Jan. 6-9, 1997

An octree-based method is presented for the automatic grid generation and CFD solution of flows around complicated geometries. This method decouples the input surface and resultant volume grid so that the user need only be concerned with the actual input geometry and flow conditions. To encourage its use as a design tool, a complete parametric aircraft specification has been added to rapidly generated well-formed input geometries and model realistic aircraft shapes; support is included for embedded boundary conditions to handle jet engines and propellers. Object-oriented programming is used for the implementation, and coarse-grain parallel computing code has also been used to reduce the time required for the computation. (Author)

An Octree Solution to Conservation-laws over Arbitrary Regions (OSCAR)*

AIAA 97-0198

Eric F. Charlton† Kenneth G. Powell‡

January 6, 1997

Abstract

An octree-based method is presented for the automatic grid generation and computational fluid dynamics solution of flows around complicated geometries. This method decouples the input surface and resultant volume grid so that the user need only be concerned with the actual input geometry and flow conditions. To encourage its use as a design tool, a complete parametric aircraft specification has been added to rapidly generate well-formed input geometries and model realistic aircraft shapes; support is included for embedded boundary conditions to handle jet engines and propellors. Object-oriented programming is used for the implementation, and coarse-grain parallel computing code has also been used to reduce the time required for the computation.

1 Introduction

Computational fluid dynamics (CFD) has progressed very rapidly through the years, but the key difficulty of building a suitable grid for computation remains a significant cause of the delay between forming a flow problem to be solved and actually understanding the resultant flow field.

Structured grids are built with a repeating geometric and topological structure. Structured grids are usually formed from quadrilaterals (2D) or bricks (3D), as this is perhaps the simplest repeating topology to compute on; they are often simpler to deal with, specifically in terms of setup, computation, and visualization. This rigid topology may favor the computation, also, leading to simple and accurate computation of spatial derivatives and often cancellation of higher-order error terms. However, this rigid topology usually requires an inordinate amount of expert-type assistance in laying out a structured grid around any complicated geometry, such as an airplane.

Unstructured grids are best characterized by no such repeating geometry and topology that can be controlled only very locally. Unstructured grids are typically formed from simplices (triangles (2D) or tetrahedra (3D)), and the fact they have no repeating topology can make it very difficult to create and compute the necessary cell-to-cell connectivity for CFD. Also, the random orientation and volume inefficiency can lead to a very large number of cells being necessary to fill the computational domain.

Tree-based grids, including the octree, fall somewhere in between. There is a repeating geometric structure, but there is no rigid topology. This produces a combination with some of the best (and some of the worst) properties of both structured and unstructured grids.

The *oscar* project has several goals. The first is to automatically generate a computational volume grid for an arbitrarily complicated geome-

†Graduate Student Research Assistant, AIAA Student Member, charlton@umich.edu, http://www-personal.umich.edu/~charlton

‡Associate Professor, Senior AIAA Member, powell@umich.edu

try to facilitate the solution of complex flows. In the process, we study the necessary computational algorithms for the computational geometry, numerical flow solution, data structures, and parallel computing. The second goal is to create a tool that can be used for the study of complicated aerodynamics problems for use in an educational-design environment.

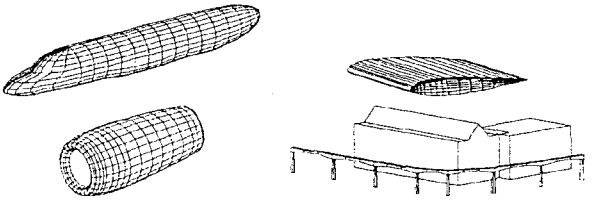

Figure 1: Simple shapes are combined into nearly complete aircraft (or building) configurations

2 Input Data

In general, a CFD problem consists of an equation model, geometry, and suitable boundary conditions (BC's). For *oscar*, the input geometry can be any number of closed, possibly intersecting, polyhedra made from planar polygons. To this list, *oscar* requires only a few parameters: minimum and maximum adaptation levels; the extent of the domain; adaptation and convergence criteria; run-time values for flow-solver control; and report and checkpoint frequency. This is one of the key advantages to a nearly-automatic method; the required input for a computational solution is little more than the physical problem!

While the ability to work with arbitrary configurations is significant and desirable, there are a few drawbacks to this high level of automation. First, there is very little feedback on the appropriateness of the geometric description or the resultant grid. A flat plate can be "over-resolved" with 10,000 triangles as easily as a smooth fuselage can be described far too coarsely, perhaps with more than one cell-per-facet on a curved surface. While the problem with the latter is easy to understand, the performance implications of the former can be just as unwelcome. Second, using the octree, the grid is isotropically refined; if the resultant solution has a region of two-dimensional character (e.g. a long straight wing), the resolution may be unnecessarily fine in the spanwise direction, leaving fewer resources available to solve for other significant features and leading to unnecessary computational expense. Finally, most automatic grid generation codes are extremely intolerant of any trouble in the geometric description. A skilled "grid-generation engineer" can often mask input problems while building the grid, but since there is no user involved in the grid generation, if *oscar* is given poorly defined input, the grid will come out ill defined. This can result in either an unnoticeable effect, a converged-to wrong answer, or even an instability leading to a divergent flow "solution."

2.1 Airplanes

Complicated (airplane) shapes are built by combining simpler basis shapes in appropriate locations, sizes, and orientations. As shown in Figure 1, these shapes include fuselages, wings, horizontal and vertical tails, pylons and struts, wheels, and engines. Each of these shapes may have boundary conditions embedded in the geometry itself. Also, all of the aircraft shapes are built from analytically-defined biparameter smooth surfaces, leaving open the possibility of eventually casting them into a non-uniform rational b-spline (NURBS) basis and allowing the shapes themselves to be manipulated with a computer aided design (CAD) system before flow analysis.

The analytic models used are described more completely in [1]. For a more detailed and realistic aircraft geometry model, consider ACSYNT [2], which uses a NURBS basis and features a comprehensive graphical user interface (GUI).

2.2 Engine BC's

In addition to the standard BC's of an applied freestream, hard wall, and symmetry plane, BC's

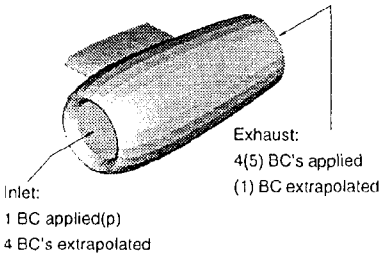

Figure 2: Applied Engine Boundary Conditions

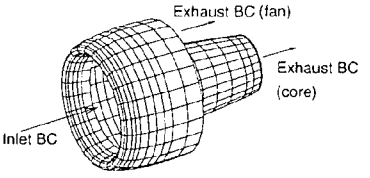

Figure 3: A high-bypass turbofan engine with engine BC's

may also be within the flow itself, either as a cut out set of "ghost cells" for a propellor or special BC's embedded in the surface definition, as for a jet engine. Since each facet of the input geometry is allowed its own BC, it is simple to specify a pressure condition on an engine inlet and specify an acceptable state for an exhaust condition. Based on the eigenvalues of the problem near the inlet and exhaust, conservation of mass, momentum, and energy, and a few simple assumptions (including purely axial outflow), three values must be chosen, e.g. inlet pressure (p_{in}), thrust (F), and fuel-mass flow rate ($\dot{m}_f$). Currently, this integration code has not been implemented, and *oscar* expects the exhaust state to be specified completely.

One remaining concern is that of robustness. Since the engine effect is achieved through the boundary, it is expected that the normal stability criterion should apply. This is usually the case, however, it was necessary to cap the pressure and density to prevent strong transients set up by this procedure from producing non-physical values and a divergent solution.

While the addition of the second exhaust condition will complicate the analysis and conservation-based parameter coupling, one logical extension of this method would be a high-bypass turbofan engine. Figure 3 shows a possible configuration; only two more geometric parameters (fan exhaust location and fan shroud scale) are necessary for the original engine geometric model to evolve into this one.

3 Grid Generation Procedure

Dating back to the late 1970's, octrees have been used for some time as a domain decomposition method for computer graphics and solid modeling[3, Section 12.6.3]. Early uses for quadtrees (the 2D Cartesian tree) included finite element analysis of structural problems in [4], and quad-trees have also been applied to the Euler equations [5], Navier-Stokes equations [6], and time-accurate Euler equations with moving bodies [7]. For 3D problems, octree research has been performed by Melton et al. (e.g. [8]) and Karman et al. (e.g. [9]).

Octrees are formed from a single cube-shaped cell known as "root" that encompasses the entire domain of the problem. This cell is then recursively divided into eight geometrically similar cells until some maximum refinement level is reached. The "leaf" cells are then either "Cartesian" or "Cut." Cartesian cells do not intersect the body and retain their cube basis shape. Cut cells intersect the body, and they are formed as the geometric subtraction of the input geometry from the basis cube cell; cut cells have "interfaces" between themselves and other cells and "boundary faces" upon which boundary conditions are applied. All of the geometric fidelity of the input body is preserved to allow the grid to model the problem as closely as possible; while this does require the computation of extra boundary fluxes, it is hoped that the ability to over-resolve the surface will improve the

accuracy of the solution. Also, a maximum refinement level difference of one is enforced across the face boundaries to minimize the loss of solution accuracy due to a non-smooth grid. The actual algorithm for computing these cut cells is described in detail in [10].

Oscar is implemented in ANSI C++ [11] using a (partial) object-oriented programming system (OOPS) model. While the use of OOPS facilitates simple code extension for the programmer, each cell is loaded with many pointers and other data, resulting in some computational resource overhead—particularly in terms of memory and time. For an example of the ease of extension, consider that hybrid-grid capability and the parallel flow solver were both one-week projects. However, the sub-cell geometry information is stored internally as an independent "object," which yields considerable flexibility in adding new code but carries an undesirable memory expense—each cut cell is a polyhedron, with some number of facets and an array of polygons; each polygon carries an array of three-vectors, each of which is built from three doubles. A production code can benefit greatly from a simpler bitwise encoding of the tree structure and a reference-based geometry data structure.

3.1 Hybrid Grids

Euler solutions can be very valuable to the analysis of a complicated compressible flow problem, but the real goal of most CFD simulations is the solution of the Navier-Stokes equations. The change from inviscid to viscous flow is not just a change of equations, but it also (often) requires a change in the method. Coirier showed in [6] that due primarily to the lack of grid-smoothness, stock Cartesian tree-based grids are simply not sufficient for an accurate solution to the Navier-Stokes equations, which demand grids to be smooth and fitted to the surface to accurately compute the necessary second-order derivatives of flow quantities.

While this seems disastrous, the use of a hybrid grid made from a surface-mapped prismatic grid combined with a tree-based Cartesian grid may be a very useful solution to this dilemma.

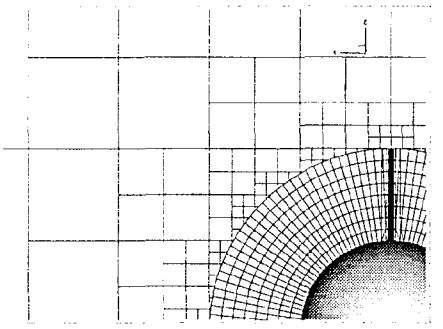

Figure 4: Hybrid Grid

As a preliminary plot shows in Figure 4, near the surface, a prismatic grid provides a suitable foundation for the solution of the viscous problem. Away from the surface where the grid smoothness is less significant, an octree can be used to quickly and efficiently fill the rest of the domain.

Karman's *splitflow* (e.g. [9]) provides support for prismatic grids created by marching normal vectors out from the surface. The actual combination of this near-surface grid with a Cartesian grid is very simple; internally, the Cartesian grid "sees" a modified body, and the only real difference is the inclusion of general-orientation interfaces in the flow solver. (No search is even necessary to build the connectivity between the Cartesian and prismatic grids with the simple programming trick of tagging the facets on the boundary of the prismatic grid with something that points back to the boundary cell.)

The difficulty turns out to be the actual computation of these normal vectors, which is a significant problem to those in the field of prismatic grid generation in general. While in smooth convex regions, the vertex-normals can be simple averages of the vertex's facet-normals, this usually leads to "crossed normals" near non-convex corners. While this has been problematic to CFD for some time, a true prismatic grid must extend to the edge of the computational domain, while a hybrid grid's prismatic grid needs only to reach the edge of the boundary layer. One drawback would be that the decoupling of the input surface and the volume grid is lost, as a user must now supply a smooth surface definition that is

adequate for a Navier-Stokes solution.

3.2 Parallel Grid Generation

With a recursive procedure and an available geometric boundary, one would expect that parallelization could provide a significant performance improvement. While faster production-level algorithms remove the need for performance improvement in the grid generation stage, *oscar* certainly benefits from a coarse-grain parallelization using multiple processors to speed it along. Given the convenient recursive definition (and code) and the lack of available knowledge about the required work at startup, physical domain decomposition is not well suited, and p-threads are the favored method. But, since a threaded system was not available for development, the grid generation is divided manually onto a handful of processors (usually four) and allowed to proceed from there with a little use of the "message passing interface" (MPI, [12]) for initial decomposition information.

There is a noticeable increase in performance through parallelization, but by relying on a coarse-grain user-driven domain decomposition, this increase is rarely linear. Also, some of the intermediate data handling must be done with the serial code, because it is not clear how to cast particular algorithms on a parallel system. In particular, grid smoothing and pruning, flow-solver domain decomposition, and post-processing must be done as intermediate serial steps. While these are actually interactive and do not require too much time, the job is removed from the parallel environment and interactive expert assistance is still required (both effects are undesirable, because parallel computers are often run in batch, and it is best to not leave the queue).

4 Flow Solution

An explicit second-order finite-volume unstructured flow solver is used for the flow field computation. Reconstruction of spatial derivatives is accomplished with a least-squares formulation, and a modified min-mod limiter is used to encourage monotonicity. Fluxes are computed with Roe's flux difference splitting. While the primary goal is to converge to a steady-state, a time-accurate flow solver is also provided, and it includes cell-merging to prevent the very small cells from forcing unusably small time-steps. Comparison to an exact solution for a cylindrical flow has been used to establish second-order accuracy.

4.1 Grid Adaptation

$$\begin{aligned} \tau_r &= |\nabla \times \vec{v}| L^{\frac{3}{2}}, \sigma_r = \sqrt{\frac{\sum_{i=1}^{n} \tau_{r_i}^2}{n}} \\ \tau_c &= |\nabla \bullet \vec{v}| L^{\frac{3}{2}}, \sigma_c = \sqrt{\frac{\sum_{i=1}^{n} \tau_{c_i}^2}{n}} \\ \tau_s &= |\nabla p - a^2 \nabla \rho| L^{\frac{3}{2}}, \sigma_s = \sqrt{\frac{\sum_{i=1}^{n} \tau_{s_i}^2}{n}} \end{aligned} \quad (1)$$

The grids can be adaptively refined based on the measures described in Equation 1, which use the local values of the curl and divergence of velocity ($\vec{v}$) and the strength of the numeric entropy wave [13]. Cells are flagged for refinement if:

$$|\tau_c| > \sigma_c \text{ or } |\tau_r| > \sigma_r \text{ or } |\tau_s| > \sigma_s \quad (2)$$

This seems very automatic, but adaptation rarely is this simple. Often, scalars are necessary to encourage adaptation around one feature instead of another. Also, "no-adaptation boxes" are supported and almost always required in the exhaust wake of an applied engine jet. In general, it appears that this criterion will adapt to flow features in a certain order: applied "man-made" features (jets and inlets), bluff-body effects, shock waves, stagnation points and lines, wingtip vortices, and finally softer features such as wing pressure peaks. If detailed resolution of a particular one of these is required, it is usually necessary to reduce the adaptation resources allotted to the others by scaling the adaptation criterion to favor the preferred mode. (This is a common problem; as mentioned by Allmaras in [14], support for no-adaptation regions is also found in Boeing's *tranair*.)

4.2 Parallel Flow Solution

An MPI unstructured fast flow-solver for the Euler equations (*muffee*) has also been built. *Muffee* is a much less sophisticated code than *oscar*, and it relies on many of *oscar*'s object libraries to simplify and speed development. *Muffee* has been used to accelerate flow solutions using The University of Michigan's IBM SP/2.

Muffee expects the same input as a true unstructured-grid flow-solver; each processor needs a description of the cells, interfaces, and boundary faces in its domain, as well as a set of files that describe the interfaces between processors. While this is a lot of data, *oscar* can quickly write out the setup files after doing a simple domain decomposition.

Since neither *oscar* nor *muffee* requires any special decomposition, it is relatively easy to break a large problem into several smaller parallel problems, either by using a decomposition based on the octree itself or a decomposition based on "computational nodes." The computational node decomposition assumes some number of computational centers and assigns the processors to the cell based on which center is nearest the cell's centroid. Then, simple centroid-based rules can be used interactively to move groups of cells from one processor to another before committing to the parallel run. Admittedly, this would be difficult to completely automate, but it is simple for someone knowledgeable about the general flow characteristics to choose these computational-centers; while expert assistance is required, it is very little, and the process is interactive on an engineering workstation.

5 Example Results

Plots from a few example runs are included here (the color plots are included at the end). While the emphasis is on computational aerodynamics analysis for aerospace engineering, it is important to note that this method could be extended to aerodynamic problems in other fields.

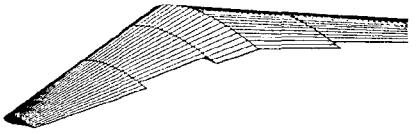

Figure 5: A swept, tapered wing with flap cutout

5.1 Reconnaissance Aircraft

The "U2-RG" case is based on Warren Eaton's experimental photo-reconnaissance aircraft, which is based on the Lockheed *U-2C*. This is a complicated geometry, particularly with the deployed landing gear and engine inlet and exhaust (with appropriate boundary conditions). The (half-) grid has adaptively refined to 270,000 cells (98,000 cells are cut, with 278,000 polygons defining the surface), and both the serial and parallel codes were used for the flow solution, using four processors on the SP/2.

Figure 6 shows pressure contours on the surface geometry with the grid through the symmetry plane and part of the grids through the engine inlet and the main landing gear rod. The configuration is comparable to a "go-around" situation (flaps up, gear down, high thrust, low angle-of-attack, and a moderate upstream speed, $M_\infty = 0.3$). Note the high-pressure buildup on the front of the landing gear and nose and the smooth pressure contours on the wing; the tail has very little change in pressure due to the very thin airfoil section. The adaptive refinement visible near the engine jet resulted from the failure to extend the "no-adaptation box" far enough aft. The jagged contours near the leading edge of the outboard wing are most likely the result of grid refinement boundaries; these contours should become smoother with more adaptation. This case will be rerun with a wing with flap cutout (as in Figure 5) and a deployed flap in a more realistic approach configuration.

5.2 Supercritical Business Jet

A surface plot for the "BizJet" case is shown in Figure 7. While this model is dimensionally based on the Cessna *Citation X*, the Cessna is a vastly superior aerodynamic design. The upstream Mach number is $M_\infty = 0.86$—definitely supercritical but well shy of the Cessna's $M_\infty = 0.92$ cruise; note the strong shock along the trailing edge of the wing and around the lower surface of the engine nacelle. Engine BC's were applied to the inlet and exhaust (the inlet effects are visible in the pressure contours on the wing). Several grid adaptations were applied, but note that the shock did not dominate the adaptation process. M_{max} was around 1.6 (under the engine nacelle), suggesting that a full-potential solution may not be sufficient for this case. Note that this case had both an over-resolved geometry (somewhere a cell has 200 facets, most likely the tip of one of the tail surfaces) and an under-resolved geometry (in a few places on curved sections of the fuselage, facets with multiple cells are visible). The final (half-) grid had 370,000 cells, with nearly 140,000 cut cells and 460,000 polygons on the surface, and the flow solution was computed on the SP/2 with eight nodes.

5.3 Building Interactions

The final case included is the computation of the flow around a simplified model of Disney's EPCOT Center. This is another very complicated geometry, particularly with the inclusion of the monorail track running through the model—the monorail track is arched and the support columns are tapered.

The freestream condition was kept fast ($M_\infty = 0.18$) to prevent the loss of accuracy associated with running Euler solvers at low speeds; this is about equal to a tropical storm (not including flying debris), which is a possible worst-case design condition for a real Florida theme park. The final goal would be to simulate wind effects with a lower Mach number.

One possibly significant extension would be to see if natural heat-sinks upwind of the prevailing wind direction could be used to produce a "free" cooling of the park itself; this would not only make the park patrons more comfortable, it could possibly save the company money on air-conditioning, which of course actually heats the surroundings. There are available breezes—appropriate ducting might be able to take advantage of them. Computational consideration of this would require preconditioning to accurately solve the flow at the much lower M_∞ and the addition of heat-transfer to the flow solver and boundary procedures, both of which should be possible.

One final consideration about this case is the similarity to a simulation of the wind flow over the crowded deck of an aircraft carrier in its "recovery" configuration. Carrier landings are made much more difficult by the unpredictable wakes of other craft parked on the deck [15]. While a turbulent viscous solution would be most desirable to truly understand the swirls and twists of the wind crossing the path of landing aircraft, valuable information could be obtained from even an advection-based Euler solution. Much like the EPCOT case, conventional grid generation would be impractical for this configuration, but *oscar* should be capable of gridding and solving without any substantial difficulty.

6 Conclusions

The use of an octree-based method for automatic grid generation and flow solution is practical even for complicated geometries in an academic environment. Many different problems of interest, including many different complete aircraft configurations and multiple building interactions, can be analyzed and studied. Also, while this work can be done on engineering workstations, parallelization turns out to be a useful option to reduce the final run time.

Oscar can ultimately be divided into four parts, each of which is a significant project in itself: geometry definition, grid generation, flow solution, and visualization and analysis. For each of these areas—despite significant progress—there remains much work to be done. In addition to the previously mentioned

model geometry improvements, current interests to note include:

- More complete integration of parallel algorithms into the production code: currently, parallel computing is used to accelerate both the grid generation and flow solution phases, which are the two most time-consuming. However, there are still intermediate steps that are serial and possibly could be parallelized. While no great speed advantage remains, the possibility does exist of simply leaving the whole job in a batch queue and coming back the next day to analyze the results.

- Accelerated flow solver: preconditioning should be particularly helpful with both high-lift-configuration wings and also very-low-speed flows, such has building-building interactions. Reed in [16] has successfully used a preconditioner on Cartesian grids, and work is underway to add this capability to *oscar*.

One concern still is that the automatic production of Cartesian volume grids is still very sensitive to the input geometry quality and the robustness of the computational geometry algorithms. Robustness does not simply mean producing a "correct" answer, but often includes producing an acceptable answer when the problem has been polluted with mistakes and numeric noise.

7 Future Work

Beyond extensions to better solve current problems, there are even more interesting possibilities through additions to *oscar*'s core components, the grid generator and flow solver. By extending each of these, even more difficult and significant problems could be studied, including:

- Free-surface flow solver: with a time-accurate free-surface flow-solver, *oscar*'s problem domain could be extended to consider natural water flow problems (rivers and lakes). This could be of particular value to flood prone areas, although ground-water boundary conditions would also be needed to accurately model the physical problem. Here, the ability to deal with an arbitrary geometry is significant, as one could use sounding data, for example, to build the input geometry; again, the complexity of the input is irrelevant.

- Time accurate solutions: for some time, it has appeared that the computational demands of the grid generation were too large to make the 3D extension to moving body problems such as Bayyuk's 2D work [7] practical. Discussions with Aftosmis have since convinced me otherwise—with some reservations. For a sufficiently fast geometry engine, the only serious computational boundary is the fact that 3D problems are inherently huge, so given the current computing climate, good parallel algorithms (i.e. automatic dynamic load balanced, etc...) would be necessary for the entire process; while the grid (re-)generation may not require parallel supercomputers, in the time-accurate moving-body case, the grid generation and flow solution can no longer be separated.

8 Acknowledgments

Since Fall, 1996, support is provided by the U.S. Air Force Office of Scientific Research under contract DOD-G-F49620-94-1-0399. Support from Fall, 1992 through Summer, 1996 was provided by the U.S. Department of Energy through its Computational Science Graduate Fellowship Program, under contract W-7405-Eng-82. Computing resources have been provided by the William M. Keck Foundation Computational Fluid Dynamics Laboratory and the Center for Parallel Computing (CPC) at The University of Michigan; thank you to Dr. Hal Marshall and Mr. Andrew Caird at CPC for advice and assistance in using The UM's parallel computers and especially for their last-minute scheduling assistance. Thank you, also, to Captain Michael Aftosmis (U.S. Air Force) for many

valuable discussions and suggestions as this work has progressed, and to Mr. Warren Eaton (Gas Dynamics Laboratory, The UM) for providing dimensions and design data for his modified *U-2* design and for many valued discussions of aircraft and aviation in general. This was prepared with LaTeX and plots were produced with *Tecplot* and *Xmgf*.

References

[1] Charlton, E. F., "Analytic Aircraft Geometry Generation", unfinished, charlton@umich.edu, 1995.

[2] Gloudemans, J. R., Davis, P. C., and Gelhausen, P. A., "A Rapid Geometry Modeler for Conceptual Aircraft", In *AIAA 34th Aerospace Sciences Meeting and Exhibit.* AIAA, 1996.

[3] Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F., *Computer Graphics, Principles and Practice*, Addison-Wesley Systems Programming Series. Addison-Wesley, 2nd edition, 1991.

[4] Baehmann, P. L., *Automated Finite Element Modeling and Simulation*, PhD thesis, Rensselaer Polytechnic Institute, 1989.

[5] De Zeeuw, D. L., *A Quadtree-Based Adaptively Refined Cartesian Grid Algorithm for Solution of the Euler Equations*, PhD thesis, The University of Michigan, 1993.

[6] Coirier, W. J., *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*, PhD thesis, The University of Michigan, 1994.

[7] Bayyuk, S. A., Powell, K. G., and van Leer, B., "A Simulation Technique for 2-D Unsteady Inviscid Flows Around Arbitrarily Moving and Deforming Bodies of Arbitrary Geometry", In *AIAA 11th Computational Fluid Dynamics Proceedings.* American Institute of Aeronautics and Astronautics, 1993.

[8] Melton, J. E., Enomoto, F. Y., and Berger, M. J., "3D Automatic Cartesian Grid Generation for Euler Flows", In *AIAA 11th Computational Fluid Dynamics Proceedings*, 1993.

[9] Karman, Jr., S. L., "Unstructured Cartesian/Prismatic Grid Generation for Complex Geometries", In *Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamics (CFD) Solutions (NASA CP 3291)*, 1995.

[10] Charlton, E. F., "Computation of Cut Cells", , William M. Keck Foundation Computational Fluid Dynamics Laboratory, The University of Michigan, charlton@umich.edu, UM-CFD-96-01, 1996.

[11] Stroustrup, B., *The C++ Programming Language.* Addison-Wesley, 2nd edition, 1993.

[12] Gropp, W., Lusk, E., and Skjellum, A., *Using MPI, Portable Parallel Programming with the Message-Passing Interface.* The MIT Press, 1994.

[13] Paillére, H., Powell, K., and De Zeeuw, D., "A Wave-Model-Based Refinement Criterion for Adaptive-Grid Computation of Compressible Flows", In *AIAA 30th Aerospace Sciences Meeting and Exhibit.* AIAA, 1992.

[14] Thomas, J. L., Allmaras, S., Connell, S., Roe, P., and Venkatakrishnan, V., "Panel Discussion", In *ICASE/LaRC Workshop on Adaptive Grid Methods*, 1994.

[15] Lt. J.G. K. Quarderer, "An Aerospace Engineer's Adventure in the Navy", Aero380 Lecture, The University of Michigan, 1996.

[16] Reed, C. L., *Low Speed Preconditioning Applied to the Compressible Navier-Stokes Equations*, PhD thesis, The University of Texas at Arlington, 1995.

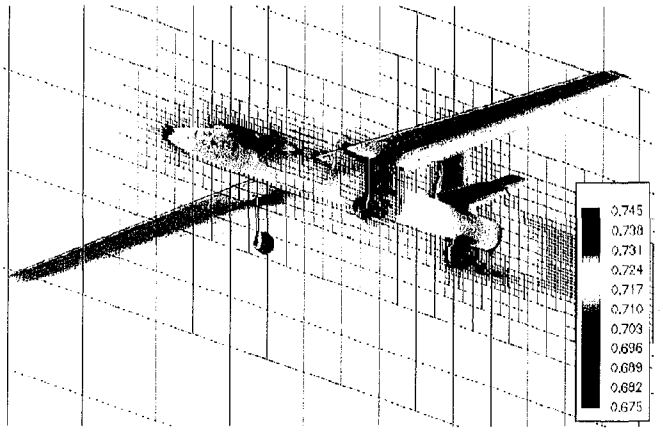

Figure 6: Reconnaissance Aircraft Grid

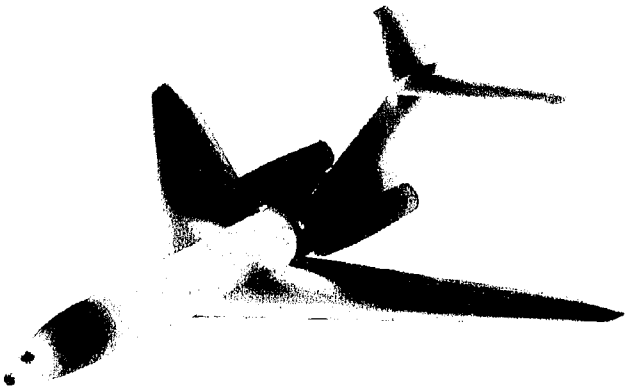

Figure 7: Supercritical business-jet configuration, pressure contours

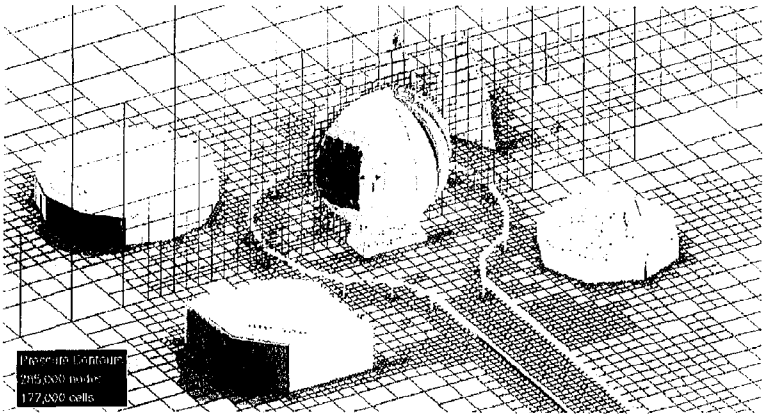

Figure 8: Flow around a simple EPCOT Center model

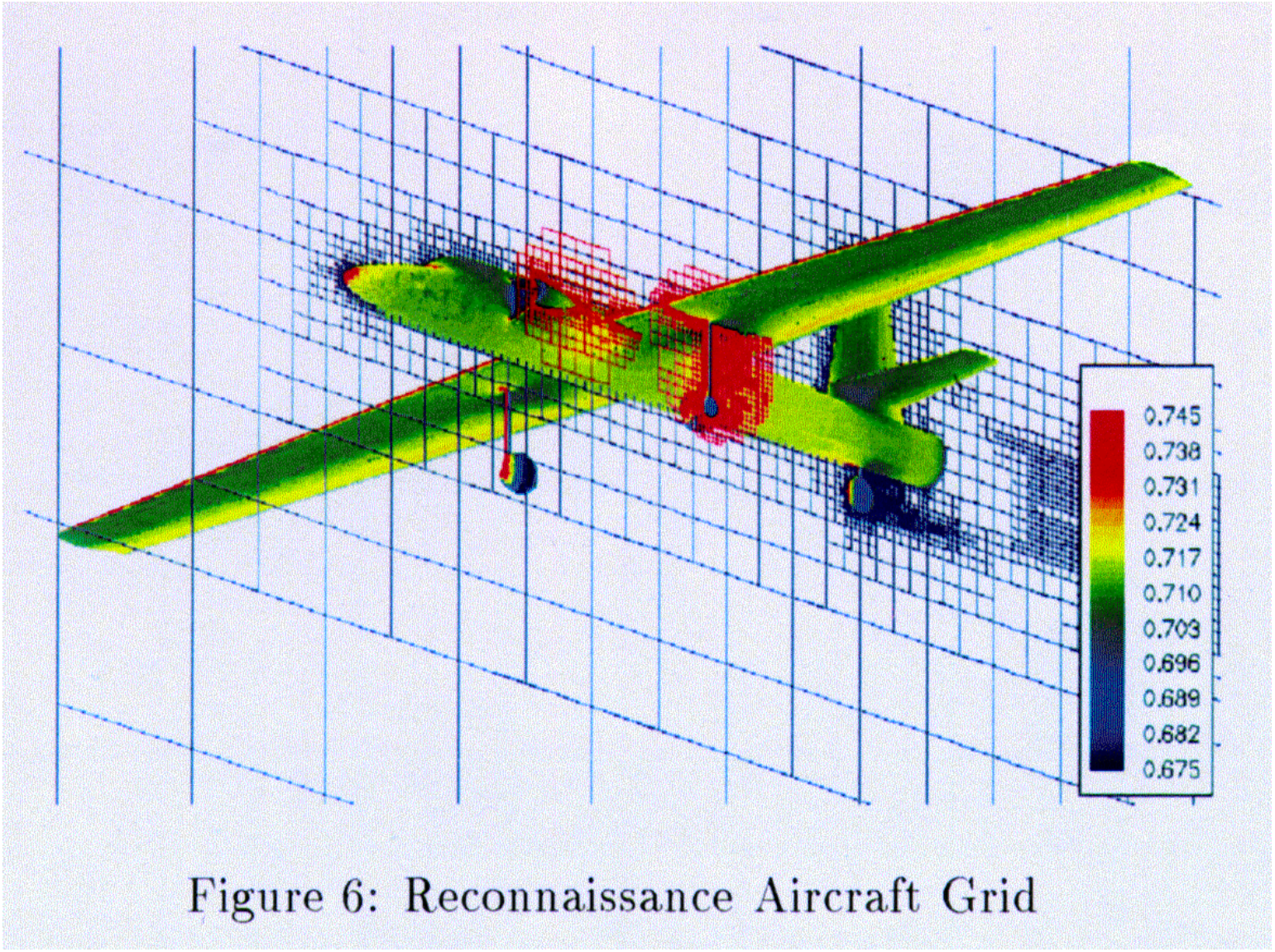

Figure 6: Reconnaissance Aircraft Grid

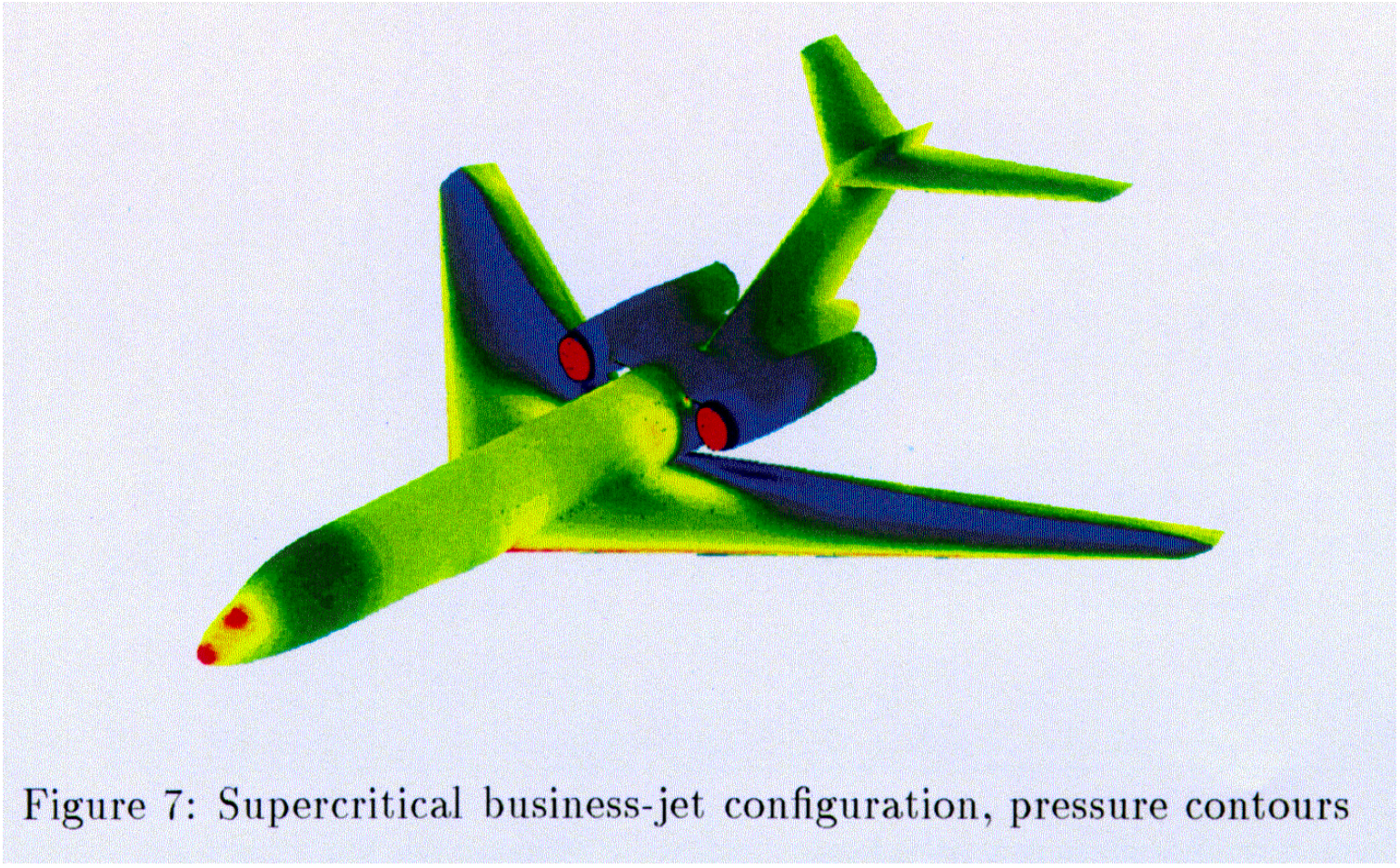

Figure 7: Supercritical business-jet configuration, pressure contours

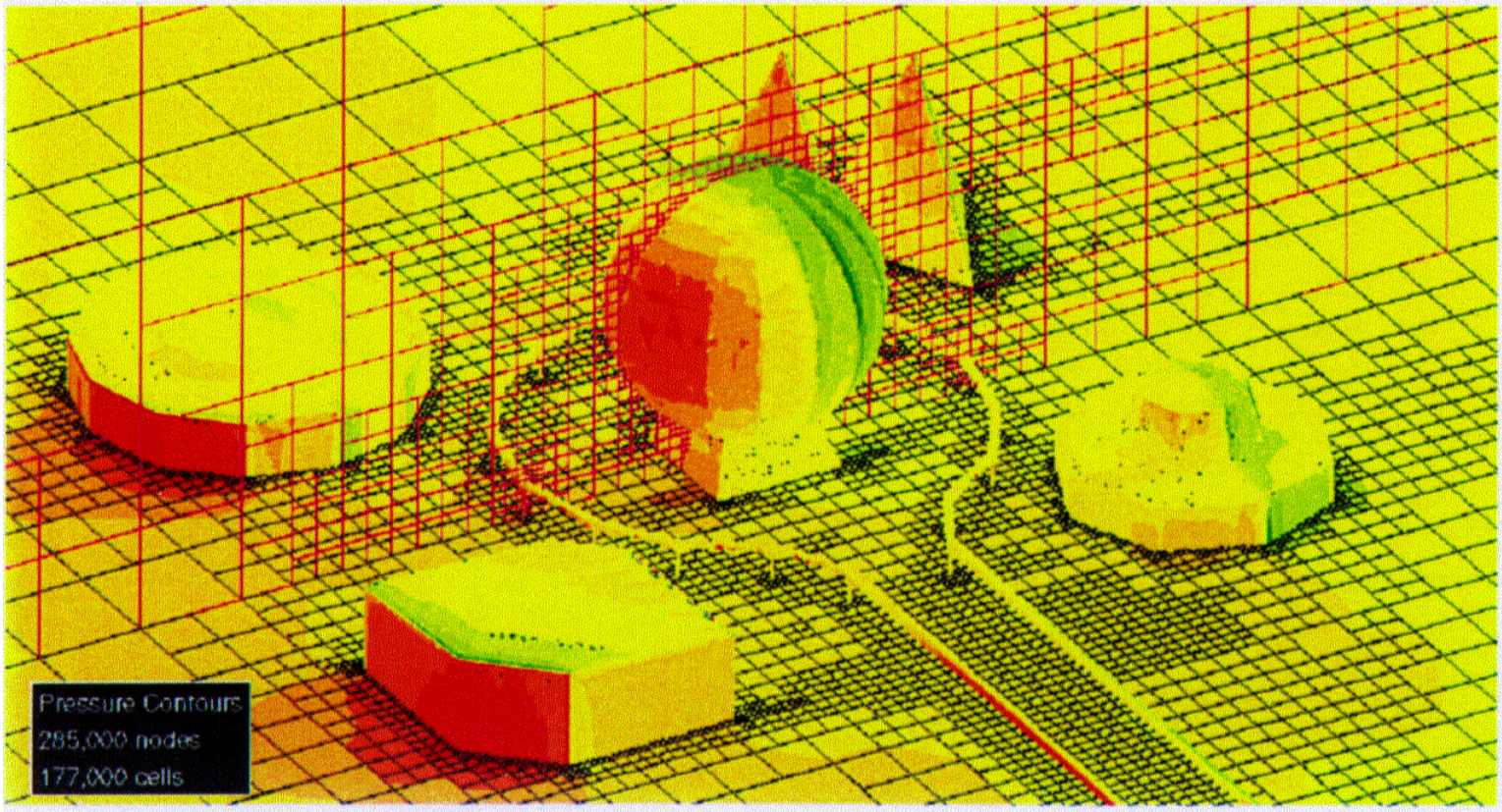

Figure 8: Flow around a simple EPCOT Center model