1995122311

DEVELOPMENT AND APPLICATION OF A 3D CARTESIAN GRID EULER METHOD

John E. Melton [*]
NASA Ames Research Center
Moffett Field, CA 94035

Marsha J. Berger [†]
Courant Institute
New York, NY 10012

Michael J. Aftosmis [‡]
USAF/NASA Ames
Wright-Patterson AFB, OH 45433

Michael D. Wong [§]
Sterling Software
Palo Alto, CA 94305

PREAMBLE

This paper documents the current state of development of our Cartesian non-body-fitted mesh algorithms for computing Euler flows around complex configurations. We include our most recent AIAA Aerospace Sciences Conference paper 95-0853 (ref. 1), which describes the geometric procedures used for the Cartesian grid generation. Two new appendices extend the description of the algorithms. Appendix I documents the flow solver used in all previous examples. Appendix II contains previously unpublished calculations for a supersonic missile (ref. 2).

ABSTRACT

This report describes recent progress in the development and application of 3D Cartesian grid generation and Euler flow solution techniques. Improvements to flow field grid generation algorithms, geometry representations, and geometry refinement criteria are presented, including details of a procedure for correctly identifying and resolving extremely thin surface features. An initial implementation of automatic flow field refinement is also presented. Results for several 3D multi-component configurations are provided and discussed.

INTRODUCTION

In this paper we discuss recent developments in our Cartesian grid approach to solving the Euler equations for flow around complex geometries. The Cartesian grid approach uses a non-body-fitted grid of rectangular cells to discretize the flow field about an object. After creating this surrounding mesh of rectangular cells, the surface geometry is simply "cut out" from the underlying cells, leaving a border of irregular cells surrounding the object. This approach has the potential to greatly simplify and automate the difficult task of grid generation around complex configurations. It replaces the difficult and case-specific problem of generating a body-fitted grid (structured, unstructured, multiblock etc.) with the more general problem of computing and characterizing the geometric intersections between the Cartesian flow field cells and the surface geometry. We present the algorithms used to automatically compute this required geometric information and describe some modifications to our original approach (ref. 3) that assist in the representation of very thin pieces of geometry, such as a spoiler, fin, or thin trailing edge section. Our algorithms have also been generalized so that the geometry of a complex configuration can be specified as a collection of distinct, repositionable, and possibly overlapping components; a single geometry description explicitly containing component intersections is no longer necessary.

[*] Aerospace Engineer, Applied Computational Aerodynamics Branch.
[†] Professor, Department of Computer Science.
[‡] Aerospace Engineer, Technical Liaison Ames Research Center.
[§] Technical Staff.

A Cartesian grid approach cannot be competitive unless it has the ability to vary the level of resolution according to the features of both the geometry and flow field. This is accomplished through the use of adaptive mesh refinement. A description of the criteria used to determine the level of refinement necessary to resolve a solid object is presented along with our approach for adaptively refining the mesh in response to features in the flow field. Finally, we demonstrate our approach by showing solutions for several three dimensional complex configurations and comparisons with experimental data.

Although various Cartesian-based approaches have been around for many years (cf. Ref. 4), renewed interest in these approaches has occurred due to continued difficulties with body-fitted grid generation. Some successful calculations in two and three dimensions, for both the potential and Euler equations, may be found in Refs. 5-12. These works differ in their approach to the data structures, flow solver, and adaptivity. Extensions to time dependent problems have been reported in Refs. 13-16.

OVERVIEW OF CARTESIAN APPROACHES

Our Cartesian grid approach to solving the Euler equations is based on the finite volume form of the conservation equations:

$$\frac{d}{dt}\iiint_{Vol} w\,dx\,dy\,dz = -\oint_{S} \mathbf{f}\cdot\mathbf{n}\,dS \qquad \{1\}$$

where $w = (\rho, \rho u, \rho v, \rho w, \rho E)$ and

$$\mathbf{f}\cdot\mathbf{n} = \begin{pmatrix} \rho\mathbf{v}\cdot\mathbf{n} \\ \rho u\mathbf{v}\cdot\mathbf{n} + pnx \\ \rho v\mathbf{v}\cdot\mathbf{n} + pny \\ \rho w\mathbf{v}\cdot\mathbf{n} + pnz \\ (\rho E + p)\mathbf{v}\cdot\mathbf{n} \end{pmatrix} \qquad \{2\}$$

For the regular (non-intersected) grid cells off of the surface, any reasonable finite volume scheme can be used. In our case, this is a Jameson-Schmitt-Turkel scheme (ref. 17) using multi-stage Runge-Kutta integration and central differencing with added second- and fourth-order artificial viscosity. The scheme must however be modified to account for the details of the geometry at the irregular cells on the body surface. As Eq. 1 shows, the volumes and face areas of the irregular cells must be provided along with the normal vector for the piece of the surface within the Cartesian cell. For second-order schemes, the centroids of the cut cell volumes and faces are also required.

In previous work, we attempted to compute the geometric information directly from the NURBS (Non-Uniform Rational B-Spline) geometrical description of the object. This can be rather time-consuming, since simultaneous nonlinear equations must be solved to compute the intersection of a cell face with the surface geometry. Furthermore, the NURBS software package we chose was incapable of robustly performing the large number of intersection calculations required to make a high-resolution Cartesian grid. These experiences strongly influenced the development of our current grid generation approach.

Our current strategy separates the grid generation process into two distinct phases. In the first phase, and prior to any flow solutions, all possible cell geometric quantities that might be needed at a later stage (for example, during subsequent mesh refinement) are computed and saved in what is called the "database" grid. This includes all the geometric information for the irregular cells at the finest possible level of refinement.

In the second phase of our grid generation process, the "actual" or "flow solution" grid is constructed solely from the geometric information stored in the database grid. This "flow solution" grid can have a variable level of resolution around the object; the appropriate level of refinement is determined by the surface curvature. The ultimate resolution of a flow solution grid is, however, limited to that of the database grid.

This separation of the grid generation process into two distinct phases and the creation of the database grid eliminates the need to repeatedly interrogate the surface geometry during later flow solution refinements. This guarantees that the flow solver will not abort midway to completion during a flow field refinement stage due to a failure in a geometric interrogation routine. However, because the database grid covers the entire surface geometry with cells of the finest resolution, our grid generation process has the disadvantages of requiring excessive storage and the calculation of possibly unnecessary geometric information.

The steps of the full 3D Cartesian grid flow simulation procedure are as follows:

Step 1. Geometry Acquisition

- Generate surface wireframe or triangulation files.

Step 2. Cartesian Grid Generation

Database Grid Generation

- Determine vertices of database grid cells.
- Compute exposed face areas of database cells.
- Sort database cells by x-location.

Initial Flow Solution Grid Generation

- Generate Cartesian flow solution grid using curvature and neighbor rule base.

Step 3. Cartesian Flow Solutions

- Cycle through flow solver and mesh adaptation steps until solution converges on grid with adequate resolution.

Step 4. Postprocessing

- Extract relevant flow field and surface data.

In the following sections we present the algorithmic details for the most important grid generation procedures, namely steps 2 and 3 above.

Geometry Acquisition

The LaWGS (Langley Wireframe Grid Standard) format for surface geometry description has been adopted as the standard input format within the current method (ref. 18). Since all of the subsequent geometrical operations take advantage of the planar nature of triangles, the LaWGS networks used to define the individual components are decomposed into collections of triangles. A provision for reading surface geometry stored in conventional triangular finite element vertex and connectivity list formats is also available.

Database Grid Generation

The following steps outline the creation of the surface-intersecting database grid cells.

1. Create coarse Cartesian background grid that encompasses the surface geometry.
2. Flag all cells that intersect the surface geometry.
3. Subdivide flagged cells, and eliminate those that are fully internal or fully external to the surface.
4. Recursively repeat Steps 2 and 3 until the maximum desired refinement level is reached.

Figure 1 depicts an extremely coarse database grid generated for a generic transport configuration. Note that the database cells can intersect several geometric components in the juncture regions between components.

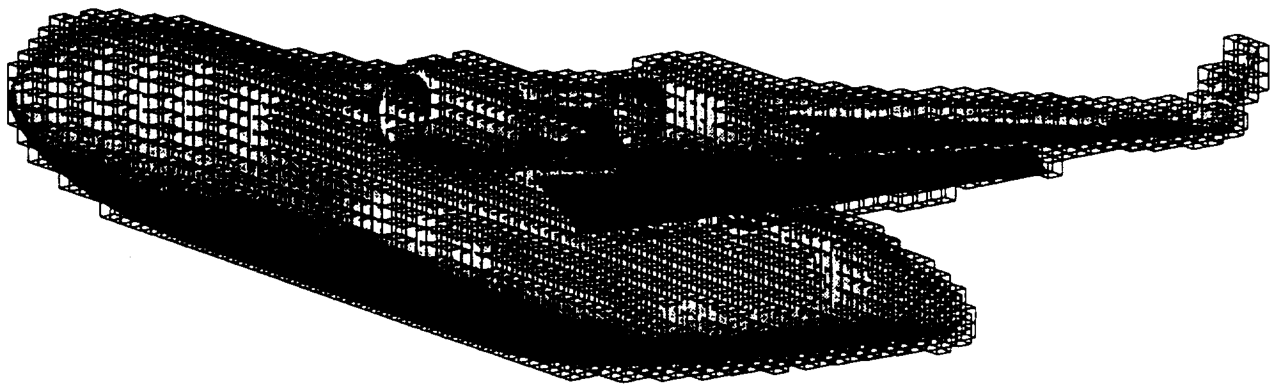

Fig. 1. Database grid enveloping a generic advanced transport configuration. Grid shown is extremely coarse for clarity. Actual database grids for practical configurations are generally divided 4-5 levels beyond that shown.

Face Intersection Calculations.—In Step 2 of the database grid generation process, each face of every cell must checked for intersection with the surface geometry components. For those components that pass an initial bounding-box test, the triangles that form the component must be checked for intersection with the face. In order for the face and triangle polygons to intersect, the intersection of one edge of one of the polygons must exist within the interior of the other. Figure 2 illustrates typical three-dimensional intersections between a triangle (polygon 1) and two non-coplanar rectangles (polygons 2 and 3).

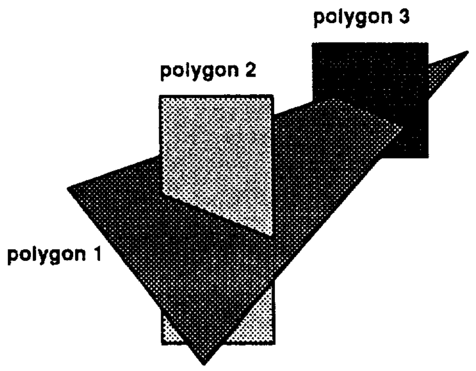

Fig. 2. Typical polygon intersections.

First, we compute the intersections between lines extending through each edge of polygon 1 and the plane that encompasses polygon 2. The polygons intersect along an edge of polygon 1 if the location of the intersection point lies both between the edge endpoints of polygon 1 and within the interior of polygon 2.

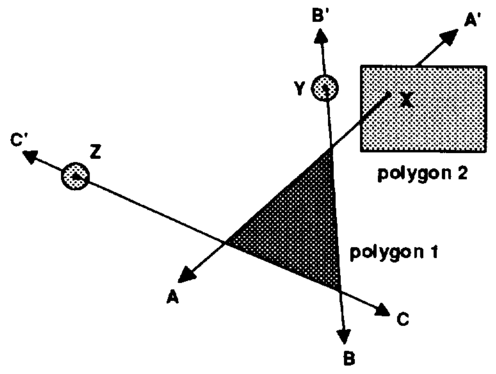

Fig. 3. Computing intersections between polygon 1 and the plane containing polygon 2.

In the case shown in Fig. 3, only the edge line *A-A'* of polygon 1 intersects the plane containing polygon 2 at point *X* within the interior of polygon 2. The *B-B'* and *C-C'* edge lines also intersect the plane of polygon 2, but both of these intersections (at *Y* and *Z*) lie outside polygon 2. Although the *A-A'* intersection lies within the interior of polygon 2, the intersection does not lie between the endpoints defining the edge of polygon 1; no intersection between polygons 1 and 2 exists.

In Fig. 4, the edge intersection line *B-B'* from polygon 1 intersects polygon 2 at point *X* within polygon 2 and between the edge endpoints; an intersection between polygon 1 and polygon 2 therefore exists.

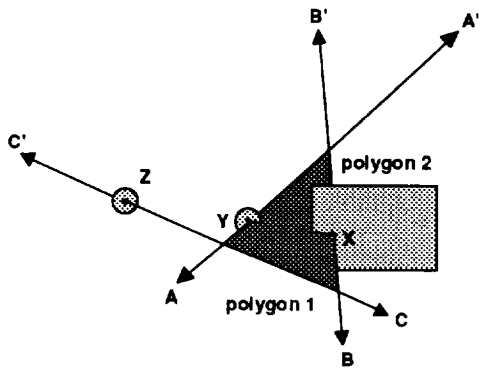

Fig. 4. An intersection between polygon 1 and polygon 2 at X.

Once the existence of intersections between the triangles that form a component and a cell face is determined, each intersection point must be characterized as occurring either interior to another component or on a flow field-exposed region of the surface. This is done using the two-step ray-casting algorithm illustrated in two dimensions in Fig. 5. First, a ray emanating from the point in question is cast in an arbitrary direction. Second, the number of intersections that occur between the ray and all other polyhedra is determined. If the ray intersects a polyhedron an even (or zero) number of times, then the point must lie outside of that polyhedron. If the number of intersections is odd, then the point lies within the polyhedron. Note that this procedure works for both concave and convex polyhedra. The polyhedra may also intersect and/or overlap, as long as each polyhedron is watertight (completely closed), with each edge of a polyhedral face shared by exactly one other facet of the same polyhedron.

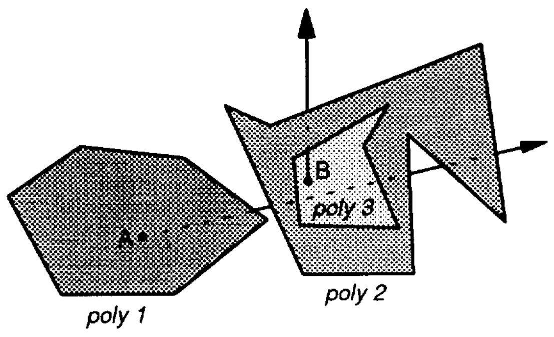

Fig. 5. Inside/outside determination.

In the above figure, the ray from point *A* makes one intersection with polygon 1, four intersections with polygon 2, and two intersections with polygon 3. Point *A* is therefore inside of polygon 1. A ray cast from point *B* makes zero intersections with polygon 1, one intersection with polygon 2, and one intersection with polygon 3. Point *B* is therefore inside of both polygon 2 and polygon 3. If no intersections with a surface component are found for a face, the ray-casting procedure is used to characterize the face as either fully internal or fully external to the surface geometry.

Face Area Calculations.—After the vertices of the database grid cells have been established, the areas of the exposed cell faces are computed. For each cell face, the cross section polygon is extracted from the surface component triangulation at the location coplanar with the cell face. The portion of this polygon that overlaps the cell face is then computed using the Sutherland-Hodgman polygon clipping algorithm (ref. 19). In Fig. 6, polygon 3 is the result of clipping the cross-section polygon 1 against the cell face polygon 2.

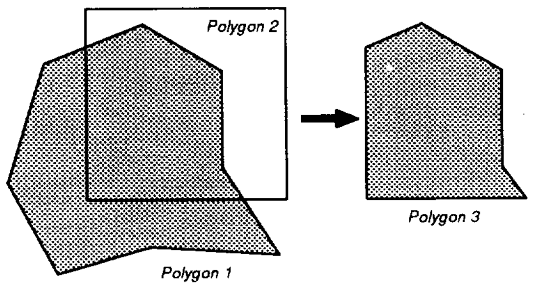

Fig. 6. Polygon clipping.

The amount of 'exposed' or 'flow-through' area for each face is then computed by subtracting the clipped cross-sectional area from the rectangular cell face area.

The procedure for computing the exposed database cell face areas becomes more complex when the surface geometry is composed of multiple (and possibly overlapping) components. Figure 7 illustrates the polygons resulting from a cross section slice at a location where the components do not intersect. The collection of polygons *A*, *B*, and *C* form the borders of polygon group *i*, and polygons *D* and *E* form the borders of polygon group *j*. Polygons *A* and *D* enclose 'solid' or 'non-flow' area, while polygons *B*, *C*, and *E* surround the 'flow-through' area within *A* and *D*.

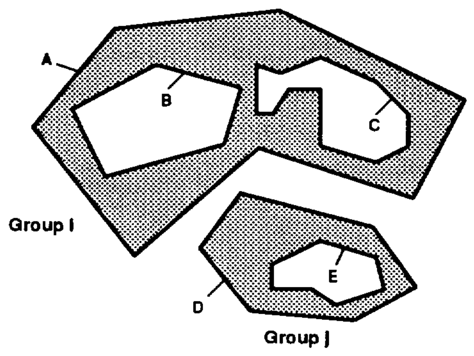

Fig. 7. Polygon groups.

For bodies composed of multiple intersecting components, the cross-section polygon groups arising from multiple components may overlap, as in Fig. 8.

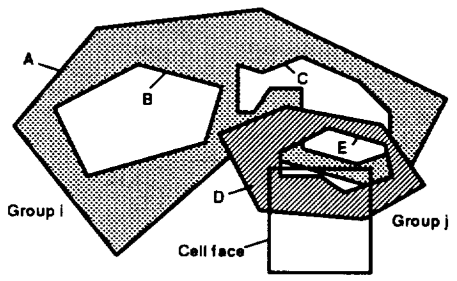

Fig. 8. Overlapping polygon groups.

To compute the net amount of each database cell face that is exposed or 'flow-through', an algorithm for computing the net amount of 'solid' area that overlaps the face rectangle is required. The equation for the solid area, A^{-}_{total}, can be expressed as

$$A^{-}_{total} = \sum_{i=1}^{n} A_i^{-} - \sum_{i=1}^{n} A_i^{+} + \sum_{i}^{n} \sum_{j}^{i \neq j} A_i^{-} \cap A_j^{+} - \frac{1}{2}\left(\sum_{i}^{n} \sum_{j}^{i \neq j} A_i^{+} \cap A_j^{+} + \sum_{i}^{n} \sum_{j}^{i \neq j} A_i^{-} \cap A_j^{-} \right) \qquad \{3\}$$

In Eq. 3, the indices *i* and *j* refer to individual component 'polygon groups', where *A* is the area of the polygon group contained within the rectangular boundaries of the face. The '+' and '-' superscripts denote the external (flow-through) and internal (to a component) areas enclosed by each group, respectively. The input to the overlap algorithm requires a listing of all the polygons associated with each group and that all polygons within every group be initially designated as enclosing either internal or external area. Finally, the Sutherland-Hodgman algorithm is used to compute the net amount of overlapping area contributed by the polygon groups. The use of the Sutherland-Hodgman algorithm is complicated by the fact that the target clipping polygon must be convex, so the Boolean intersection of two possibly non-convex polygons must be computed by first decomposing each polygon into two sets of triangles, and then keeping track of the sum of the areas resulting from clipping all triangles in each polygon against those of the other polygon.

This adoption of a component-based approach to Cartesian grid generation is somewhat analogous to the use of Chimera procedures for conventional body-fitted structured grids (ref. 20). Both techniques are intended to eliminate the need to recreate new surface definitions reflecting updated intersection information whenever individual components are translated or rotated. Since the database grid cells used in our procedure are restricted to the exterior surface of intersecting components, the intersection regions between otherwise smooth components are automatically detected. The geometric intersection information computed and saved for each cell in the database grid consists of the cell vertex coordinates and the flow field-exposed area of each cell face.

Flow Solution Grid Generation

The second phase of the grid generation process uses information extracted from the database grid to build an initial flow field mesh. Starting from a coarse background grid, the Cartesian cells are automatically refined, based only on the curvature of the geometry, until the appropriate resolution is reached. This initial mesh is then used to begin the flow solutions. Further mesh refinements, based on the flow solution itself, occur during later flow field refinement stages.

The automatic refinement of the initial mesh is based on the surface curvature, which in turn is derived from the local variation of the surface normal vectors. Although the surface normal vector within each database grid cell is not explicitly included in the database, its components can be computed by taking the difference in the exposed areas of opposing faces. This procedure is illustrated in two dimensions in Fig. 9.

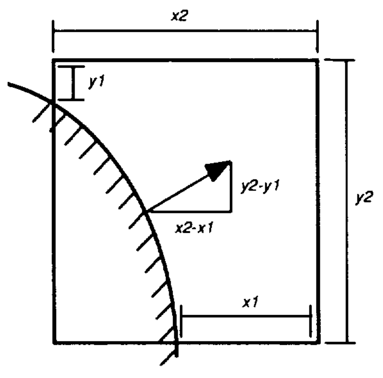

Fig. 9. Representative surface normal computed using difference in opposing exposed face areas.

For irregular flow field cells, the necessary surface normal vector and face areas are also easily obtained by summing the face areas of encompassed database cells. Figure 10 shows a flow field cell and its encompassed database cells.

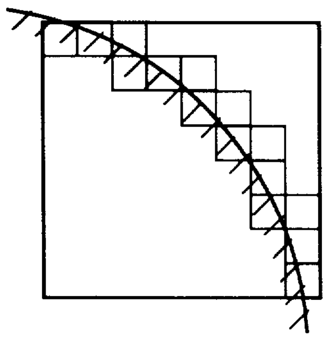

Fig. 10. Flow cell with encompassed database cells.

We use two curvature rules to refine cells in the initial flow field mesh. The first rule is based on the maximum difference in the surface normal direction between neighboring flow field cells. If the difference exceeds a user-specified tolerance, the cell is marked for additional refinement. The second rule uses the finer information in the database grid itself. If the maximum angular difference in the surface normals of the encompassed database cells exceeds a second user-specified tolerance, the flow field cell is marked for refinement, even if it passed the first test. Using these rules for the automatic refinement of the initial flow field mesh has two benefits. First, the magnitude of the surface curvature and the appropriate level of grid refinement are not inferred solely from the size of the elements used to discretize the surface. Second, the intersection regions between overlapping components are automatically detected and appropriately refined.

After the cells have been tested and marked for refinement, a buffer layer of neighboring cells is added. This creates smooth transition regions between refinement levels. Under no circumstances are neighboring cells allowed to differ by more than one level of refinement. Typical values for our grid generation control parameters are provided in Table 1.

Database angle difference	40°
Flow field angle difference	40°
Number of neighbor buffer layers	3
Total number of refinement passes	7-12

Table 1. Typical values used to create initial flow field grids.

Multiple Region Algorithms

One major disadvantage of the Cartesian approach and its reliance on a non-body-fitted grid is the difficulty in providing sufficient resolution for extremely thin components. To accurately represent the surface geometry of thin components (such as wing leading and trailing edges, fins, and engine cowlings), an excessive number of highly-refined cells are often required. Once a cell that has been split by the geometry into multiple independent regions has been identified, it should instead be replaced by multiple, non-isotropic new cells having the correct face areas and connectivity. A method of detecting cells that have been split into two or more distinct regions is therefore highly desirable, and an overview of our approach follows.

Figure 11 illustrates the split-cell situation in two dimensions:

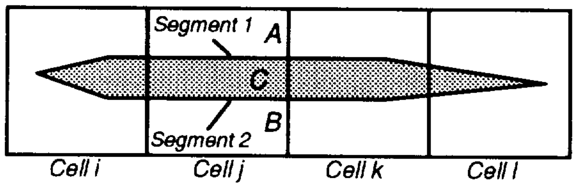

Fig. 11. Cells split into distinct regions.

Cell j is obviously split into three distinct polygonal regions (labeled A, B, and C). Polygons A and B are external to the geometry, while polygon C is completely internal. These polygons are identified and created in a procedure that examines the two segments that form the outline of the portion of the airfoil within cell j and categorizes their intersections with the cell edges (in the above figure, the segments are labeled Segment 1 and 2). Once the edges of each polygon region have been established, the area within each polygon is denoted as either fully internal or fully external to the geometry. The three-dimensional procedure for determining the number of independent regions into which a cell has been split begins by applying this two-dimensional procedure to each face and the segments resulting from the intersection of the surface geometry with that face.

Figure 12 depicts two adjoining cells (separated for clarity) intersected by the trailing edge of a wing. The portions of the wing surface within the cells are shaded.

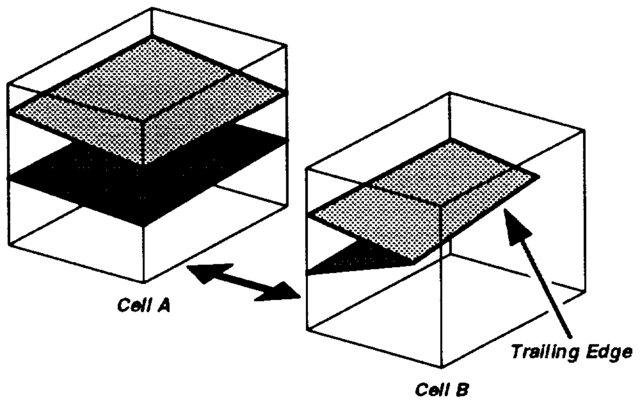

Fig. 12. Two cells near wing trailing edge.

Figure 13 shows an exploded view of the individual polygons that compose the faces of the cells.

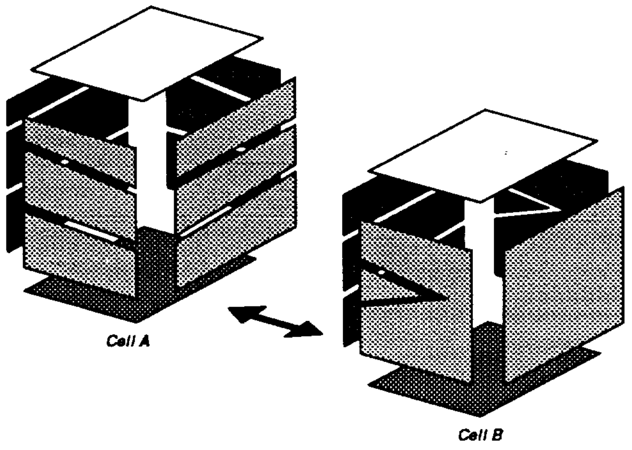

Fig. 13. Exploded view of face polygons.

After the independent polygons that compose each face are determined, matches between polygon edge segments that lie along the edges of the parent Cartesian cell are used to group adjoining polygons into the face collections that will compose the new independent regions. These links are shown with symbolic shading in Fig. 14

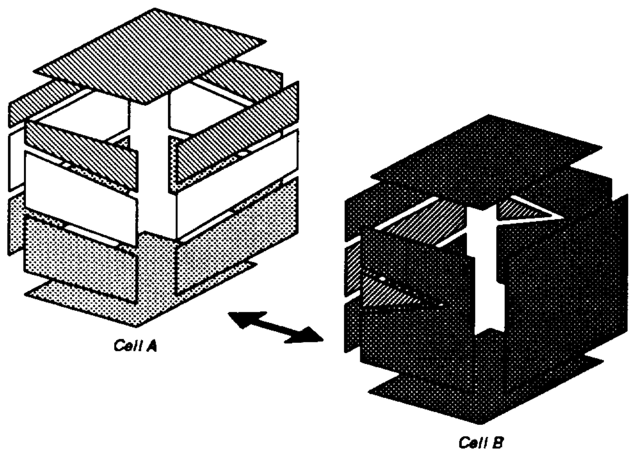

Fig. 14. Symbolic links between matching edges.

The portions of the surface that lie within the cell are then similarly matched and included into the face collections to form the complete face description of the independent regions. The exploded view of the independent regions is presented in Fig. 15, which shows that Cell *A* was split into three distinct regions, while Cell *B* was split into two.

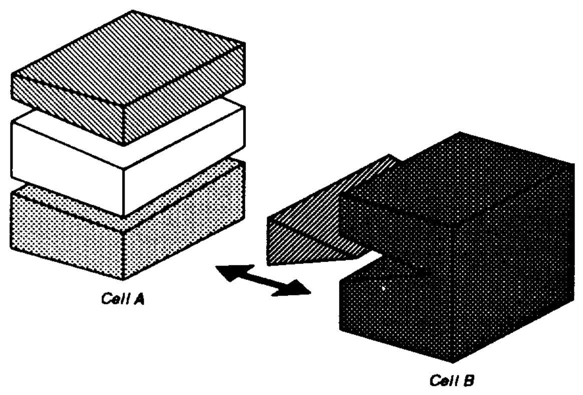

Fig. 15. Exploded view of distinct cell regions.

Using the faces that compose each polygon, the face and volume centroids of the resultant new individual regions can be computed, and the combined volumes and centroids summed and checked for consistency with the parent cell.

Volume and Centroid Calculations

The cell volumes and centroids are calculated using the Gauss divergence theorem, which transforms the required volume integrations into surface integrations over the exposed and intercepted faces of each cell. After decomposing each face polygon into triangles, a third-order three-point quadrature rule is used for the integrations.

SOLUTION ADAPTIVE CELL DIVISION

With the cell division and surface interrogation algorithms already developed for the geometric based cell refinement, the addition of solution adaptation is relatively straightforward. The approach adopted in the current work follows that presented by Aftosmis (ref. 21), appropriately modified for use within the Cartesian framework.

Approach

The underlying methodology may be characterized as a "feature detection" algorithm, since it operates directly on computed flow field differences, rather than strict discretization error estimates within the discrete solutions. The adaptation scheme uses a two pass approach in scanning the flow field for cell division candidates. The first path searches for shocks or sharp expansions, while the second tags cells containing smooth inviscid features.

This approach is based upon the observation that in a general flow field, shocks of varying strength may cause many parameters to jump by orders of magnitude (e.g. static pressure through a strong shock). Thus, there exists a danger of overlooking less prominent features with a crude detection algorithm. Failure to adapt the mesh to capture such "smooth features" may actually lead to an adaptive procedure which converges to the wrong solution (ref. 22).

Shock and Smooth Feature Detection

A normalized, undivided second difference of pressure provides a parameter which is quite sensitive to both strong and weak shocks. This quantity recognizes curvature (in computational space) in the pressure profile across a difference stencil spanning three cells. The Cartesian components of a directional refinement parameter $\mathbf{R}_s$ may be defined using a stencil that extends into the six nearest neighbors of each cell:

$$\begin{aligned}\mathbf{R}_S &= R_{S_x}\mathbf{i} + R_{S_y}\mathbf{j} + R_{S_z}\mathbf{k} \\ &= \frac{\delta_x^2 p}{\bar{p}_{(x)}}\mathbf{i} + \frac{\delta_y^2 p}{\bar{p}_{(y)}}\mathbf{j} + \frac{\delta_z^2 p}{\bar{p}_{(z)}}\mathbf{k}\end{aligned} \qquad \{4\}$$

where $\delta_x^2 p = p_{i-1,j,k} - 2p_{i,j,k} + p_{i+1,j,k}$

and $\bar{p}_{(x)} = p_{i-1,j,k} + 2p_{i,j,k} + p_{i+1,j,k}$

Cells are tagged for division when the vector magnitude of $\mathbf{R}_s$ exceeds a preset absolute threshold T_s (usually set to 0.05).

$$|\mathbf{R}_S| > T_S \qquad \{5\}$$

With the cells spanning strong nonlinearities identified, only the remaining cells are re-scanned for smoother features. Since the extrema have been removed, the remaining discrete data may be considered smoothly varying, and the refinement proceeds on a statistical basis as described in Ref. 23.

The parameter chosen to identify smooth, inviscid features in this preliminary work uses undivided first differences of density, and is computed directionally as:

$$\mathbf{R}_l = \frac{\delta_x \rho}{\bar{\rho}_{(x)}}\mathbf{i} + \frac{\delta_y \rho}{\bar{\rho}_{(y)}}\mathbf{j} + \frac{\delta_z \rho}{\bar{\rho}_{(z)}}\mathbf{k} \qquad (6)$$

Cells are tagged for division if the magnitude of $\mathbf{R}_l$ exceeds the flow field mean by a pre-specified fraction, T_l, of the standard deviation, σ.

$$\textit{if}\ \left|\mathbf{R}_l\right| > \overline{\left|\mathbf{R}_l\right|} + \sigma T_l \ \ \textit{then tag cell for division}$$

T_l was set to 0.1 for the adaptation example in the next section. Ellipsoidal and hexahedral selection tools permit the user to restrict adaptation to certain regions of the flow field. This ability further focuses computational resources on specific regions of an evolving discrete solution.

CARTESIAN APPLICATIONS

Selected results for three different configurations are presented in the following sections. All of the applications were performed on the CRAY C-90 at the Numerical Aerodynamic Simulation (NAS) facility at NASA Ames Research Center.

Advanced High Wing Transport

Aircraft Figure 16 displays a front view of an advanced, high wing, transport configuration. The model is a four engine aircraft and includes the fuselage, wing, engine pylons, nacelles and mass-flow plugs within each nacelle. Symmetry permitted use of a half model in the simulations. The surface description of this model consisted of twelve LaWGS networks, each of which was triangulated to form a closed, watertight polyhedron. Intersection regions between overlapping polyhedra were successfully identified and automatically discretized within the database grid generation process. The aircraft was simulated in transonic cruise at a low angle of attack.

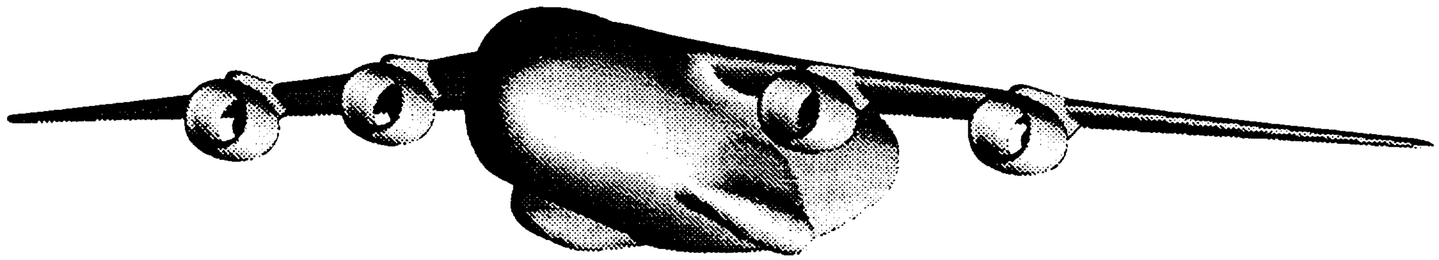

Fig. 16. Front view of an advanced, high-wing transport configuration with fuselage, wing, engine pylons, engine nacelles, and mass flow plugs defined by 12 intersecting and overlapping surface triangulations.

Starting from an initial 32x32x16 background grid, the solution process began with the generation of a five-level, geometry-adapted mesh. This coarse initial grid contained approximately 70,000 cells, and is shown in Fig. 17.

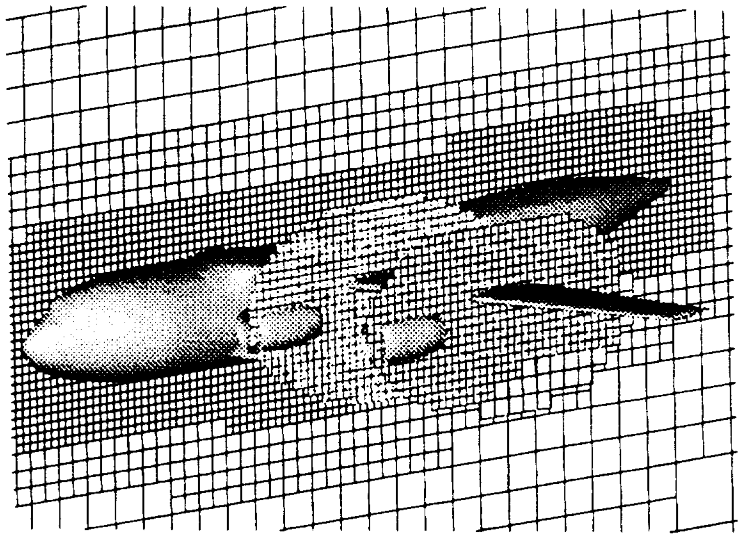

Fig. 17. Geometry-adapted starting mesh.

Note that since this mesh was created solely through the geometric cell refinement procedure described in section II.C, it actually constitutes the "starting mesh" for the adaptive flow solution. Using this grid as a starting point, the methodology generated a final solution by cycling the discrete solution through five successive flow solver - adaptive mesh refinement stages. The final mesh contained approximately 2.9 million nodes, ten levels of refinement, and extended 20 fuselage lengths fore and aft, above and below, and 10 fuselage lengths spanwise from the geometry. Figure 18 provides a view of the final, solution adapted mesh. The multiple region technology was not used in the calculations.

Figure 19 displays two views of the isobars in the discrete solution. The figure shows the presence of the main wing shock as it extends up and over the fuselage. The solution is mapped onto cutting planes at two wing stations aligned with the center of each engine nacelle. The resolution of the shock can be seen both on the main wing and inside the flow-through nacelles. Figure 20 shows a close-up of the wing shock and the mesh adaptation pattern between the inboard pylon and the fuselage. This figure also shows some irregularities in the mesh adaptation which will be removed through improvements in the rule base governing the adaptation. Surface isobars and selected spanwise C_P cuts for this case were also computed and compared well with experimental data.

High Speed Civil Transport

Figure 21 shows the surface pressure distribution for an advanced supersonic transport configuration computed with the Cartesian method. Two grids consisting of 260,000 and 490,000 cells were also generated for the configuration without nacelles and diverters. Extensive calculations and comparisons with experimental data for the wing-body configuration were made at Mach numbers from 1.65 through 2.4. In Fig. 22, the predictions at Mach 1.8 of the lift, drag, and pitching moment coefficients computed using the two grids are compared with wind tunnel data. Experimental comparisons of lift, drag and pitching moment coefficient versus angle of attack were in similar agreement for all other Mach numbers. The multiple region technology was not used in the calculations.

Wing C

An initial demonstration of the multiple-region cell technology was performed using the Wing C configuration (ref. 24). This wing has a leading edge sweep of 45 degrees, an aspect ratio of 2.6, and a thin, non-planar trailing edge due to its use of twisted, supercritical airfoil sections. Two grids of approximately 203,000 and 610,000 cells were used to compute transonic flow solutions. The 610,000 cell grid was created by performing one additional refinement of the coarse grid. Both coarse grid and fine grids were generated in a total of 30 C-90 CPU minutes. The dimension of the smallest cells in the finest grid is approximately 2.8 times larger than the wing trailing edge thickness, and 0.0063 of the mean aerodynamic chord. After computing solutions at a Mach number of 0.85 and an angle of attack of 5 degrees for both the coarse and fine grids, the coarse grid was examined for cells that could be split into multiple distinct regions. Using the multiple-region algorithms, new cells were then added, yielding a corrected version of the coarse grid with approximately 204,000 cells. The flow solver executed at a rate of 6.5 μ-seconds/cell/iteration, and all solutions were converged at least 4 orders of magnitude in the density residual. Solutions for all three grids are compared to wind tunnel data at selected span stations in Fig. 23. Most of the disagreement between the computed and experimental distributions at the outboard station can be attributed to viscous effects and the separation cell observed in the wind tunnel tests, but these differences were not quantified. The advantages of the multiple-region technology are, however, clearly evident. Without the multiple-region technology, the distinction between upper and lower surface cells at the trailing edge is lost, and the effective local chord for the coarse uncorrected grid is shortened by approximately ten percent. Although the fine grid results show that this effect can be reduced by doubling the cell resolution, the application of the multiple-region algorithms is clearly a significantly more efficient means of capturing extremely thin geometric details. Figure 24 shows the upper surface of the Wing C configuration colored by the computed surface pressure distributions for the coarse grids. The effect of the uncorrected multiple-regions cells along the trailing edge is clearly evident in the left half of the figure. The right half of the figure indicates the benefits of the application of the multi-region technology and the associated reduction in the surface pressure errors.

CONCLUSIONS

The current grid generation approach has been used to develop and successfully demonstrate four important features necessary for future Cartesian grid approaches. The first is the capability to identify and refine flow field cells in regions of large surface curvature before a single flow solution is performed. The second feature is a "component-based" capability for handling surface geometry. This feature allows simple translations and rotations to be applied to individual component definitions to create new configuration variations, eliminating the need to return to the CAD system to create new surface discretizations that explicitly reflect the intersections between components. Third, a means for correctly distinguishing the upper and lower surface of extremely thin (less than a cell dimension) geometrical features improves the fidelity of the simulations without excessive refinement. The fourth feature is the full incorporation of automatic and adaptive flow field mesh refinement. Current work focuses on improving the accuracy of the surface boundary conditions and restructuring the grid generation procedures for increased efficiency.

This research has targeted the development and integration of many of the important technologies required for a future automated Euler Cartesian CFD procedure. Demonstrations of their application to arbitrary, three-dimensional configurations indicate the method's ability to dramatically reduce the time and effort required to produce useful inviscid CFD simulations for complex vehicles.

ACKNOWLEDGMENTS

M. Berger was supported in part by DOE grant DE-FG02-88ER25053, and by AFOSR grant F49620-94-1-0132. NASA's Numerical Aerodynamic Simulation Facility provided much of the computer time used in this study. This support is gratefully acknowledged.

APPENDIX I

This appendix contains additional details about the flow solver used to compute the examples in this paper. The flow solver is currently undergoing vigorous development, with special attention focused on the procedures at the solid wall boundary. We include here a description of our current algorithm for documentation purposes. The flow solver uses a multistage Jameson-Schmitt-Turkel (ref. 17) Runge-Kutta time integration procedure with local time-stepping to reach steady state (a multigrid acceleration scheme has not yet been implemented). For all interior points in the domain (e.g., those whose stencil does not contain a cut cell next to the solid wall), the usual central difference scheme augmented with pressure-switched second- and fourth-order dissipation terms is applied. Typical values of the user-specified second- and fourth-order dissipation coefficients are 1.2 and 0.5, respectively. At the edge of the computational domain, these discretizations are applied on cells that are no longer regular. Currently, the location of the volume centroid and the area centroids of the exposed portions of cut faces are not used when computing the fluxes. Along the boundaries, the order of accuracy of the basic central difference scheme therefore drops to first order. For cells with cut faces or faces completely internal to the geometry, the second and fourth derivative dissipation terms are contaminated by first and third difference contributions. Numerical experiments have shown that the errors introduced by these dissipation terms appear, however, to be much smaller than those associated with our current simple flux calculation strategy and our first-order solid wall boundary condition, in which the pressure at the wall is extrapolated from the interior using piecewise constant extrapolation. These cut cell discretizations have been improved in research versions of the code, but are not yet incorporated into the production version. In careful numerical studies of these methods (which will be reported elsewhere), improved strategies for the flux calculations at intersected faces and along the solid wall boundaries lead to a considerable reduction in the boundary error.

APPENDIX II

Static aerodynamic coefficients for a supersonic missile configuration (Ref. 2) were computed at Mach numbers between 2.30 and 4.63. The missile has an ogive nose cone and four cruciform fins mounted at the aft end of a cylindrical fuselage (side and front views are provided in Fig. 25, which is taken from Ref. 2). Three grids of 290,000, 470,000, and 624,000 cells were used. The lift, drag, and pitching moment coefficients for all three grids are compared with experimental data for Mach 2.30 in Figure 26. Additional comparisons with the 290,000 cell grid for Mach numbers 2.96, 3.95, and 4.63 are presented in Figures 27, 28, and 29. The trends with increasing Mach number are seen to be accurately predicted, and the effects of changes in angle of attack are in good agreement up to approximately 15 degrees. No corrections for skin friction were made to the computed drag values.

REFERENCES

1. Melton, J.E., Berger, M.J., Aftosmis, M.A., and Wong, M.J., "3D Applications of a Cartesian Grid Euler Method," AIAA Paper 95-0853, January 1995.
2. Fuller, D.E., Richardson, C.S., and Spearman, M.L., "Aerodynamic Characteristics of a Wingless Missile with Aft Cruciform Controls at Mach Numbers from 2.30 to 4.63," NASA TM-X 2488, 1972.
3. Melton, J.E., Enomoto, F.Y., and Berger, M.J., "3D Automatic Cartesian Grid Generation for Euler Flows," AIAA Paper 93-3386-CP, July 1993.
4. Varga, R., *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, N.J., 1962.

5. Purvis, J.W., and Burkhalter, J.E., "Prediction of Critical Mach Number for Store Configurations," *AIAA Journal*, Vol. 17, No. 11, November 1979, pp. 1170-1177.

6. Samant, S.S., Bussoletti, J.E., Johnson, F.T., Burkhart, R.H., Everson, B.L., Melvin, R.G., Young, D.P., Erickson, L.L., and Madson, M.D., "TRANAIR: A Computer Code for Transonic Analyses of Arbitrary Configurations," AIAA Paper 87-0034, January 1987.

7. Gaffney, R.L., Hassan, H.A., and Salas, M.D., "Euler Calculations for Wings Using Cartesian Grids," AIAA Paper 87-0356, January 1987.

8. De Zeeuw, D., and Powell, K.G., "An Adaptively-Refined Cartesian Mesh Solver for the Euler Equations," AIAA Paper 91-1542-CP, April 1991.

9. Tidd, D.M., Strash, D.J., Epstein, B., Luntz, A., Nachson, A., and Rubin, T., "Application of an Efficient 3-D Multigrid Euler Method (MGAERO) to Complete Aircraft Configurations," AIAA Paper 91-3236, September 1991.

10. Morinishi, K., "A Finite Difference Solution of the Euler Equations on Non-body-fitted Cartesian Grids," *Computers Fluids*, Vol. 21, No. 3, 1992, pp. 331-344.

11. Coirier, W.J., "An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations," NASA TM 106754, October 1994.

12. Karman, S.L., "Splitflow: A 3D Unstructured Cartesian/Prismatic CFD Code for Complex Geometries," AIAA Paper 95-0343, January 1995.

13. Berger, M.J., and LeVeque, R.J., "An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries," AIAA Paper 89-1930-CP, June 1989.

14. Berger, M.J., and LeVeque, R.J., "Stable Boundary Conditions for Cartesian Grid Calculations," ICASE Report No. 90-37, May 1990.

15. Quirk, J.J., "An Alternative to Unstructured Grids for Computing Gas Dynamic Flows Around Arbitrarily Complex Two-Dimensional Bodies," ICASE Report 92-7, February 1992.

16. Pember, R.B., Bell, J.B., Colella, P., Crutchfield, W.Y., and Welcome, M.L., "Adaptive Cartesian Grid Methods for Representing Geometry in Inviscid Compressible Flow," AIAA Paper 93-3385-CP, July 1993.

17. Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods using Runge-Kutta Time-Stepping Schemes," AIAA-81-1259, June 1981.

18. Craigdon, C. B., "A Description of the Langley Wireframe Geometry Standard (LaWGS) Format," NASA TM 85767, February 1985.

19. Foley, J.D., van Dam, A., Feiner, S.K., and Hughes, J.F., *Computer Graphics, Principles and Practice*, Addison-Wesley, 1990.

20. Benek, J.A., Buning, P., and Steger, J.L., "A 3-D Chimera Grid Embedding Technique," AIAA Paper 85-1523, July 1985.

21. Aftosmis, M.J., "Upwind Method for Simulation of Viscous Flow on Adaptively Refined Meshes," *AIAA Journal*, Vol. 32, No. 2, February 1994, pp. 268-277.

22. Warren, G.P., Anderson, W.K., Thomas, J.L., and Krist, S., "Grid Convergence for Adaptive Methods," AIAA Paper 91-1592, June 1991.

23. Kallinderis, Y.G., "Adaptation Methods for Viscous Flows," Doctoral Dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1988.

24. Keener, E.R., "Pressure-Distribution Measurements on a Transonic Low-Aspect Ratio Wing," NASA TM 86683, September 1985.

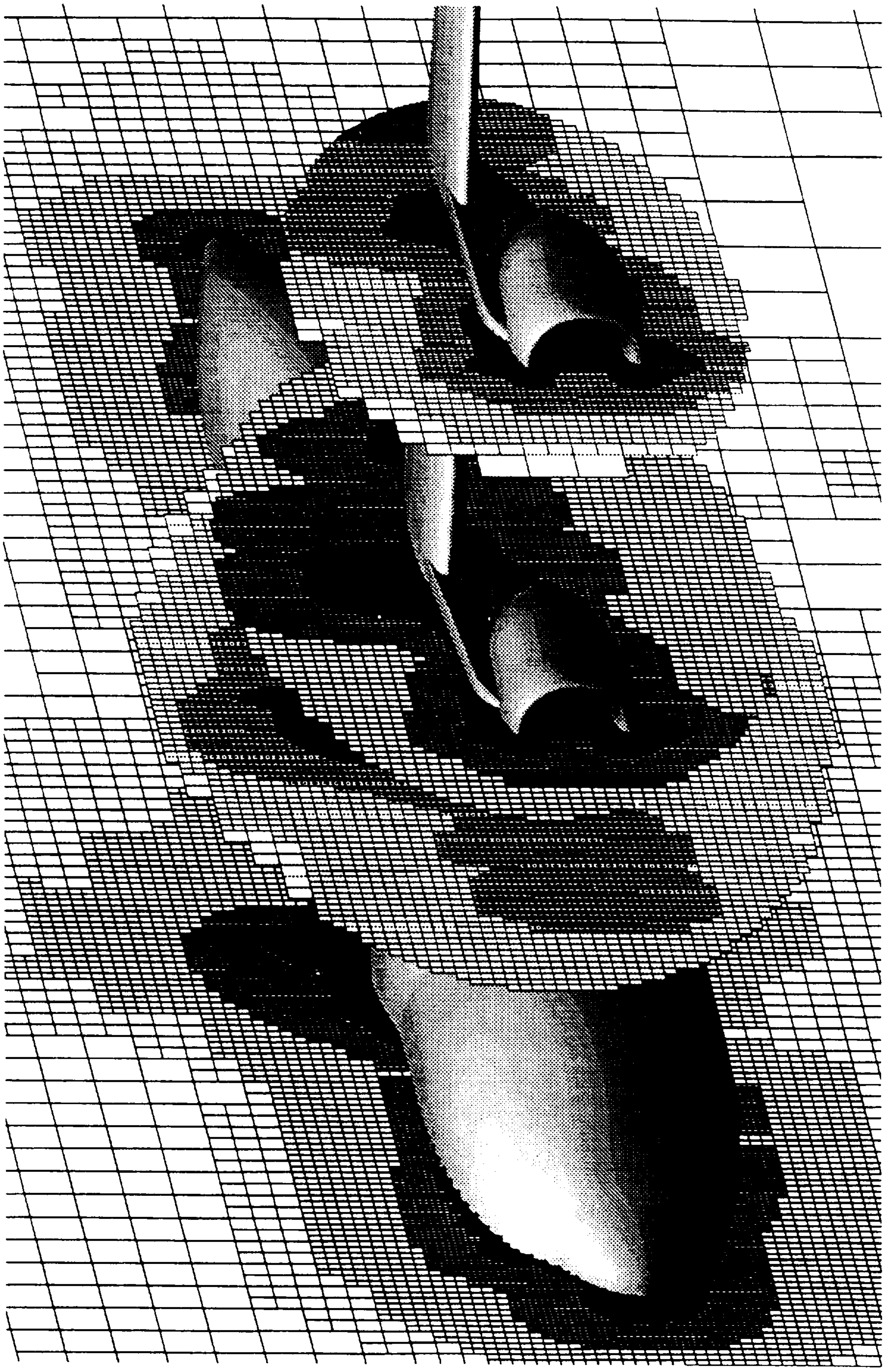

Figure 18. Adapted computational mesh for the advanced high-wing transport. The mesh contains over 2.9 million nodes. The aircraft surface is defined by twelve separate surface triangulations.

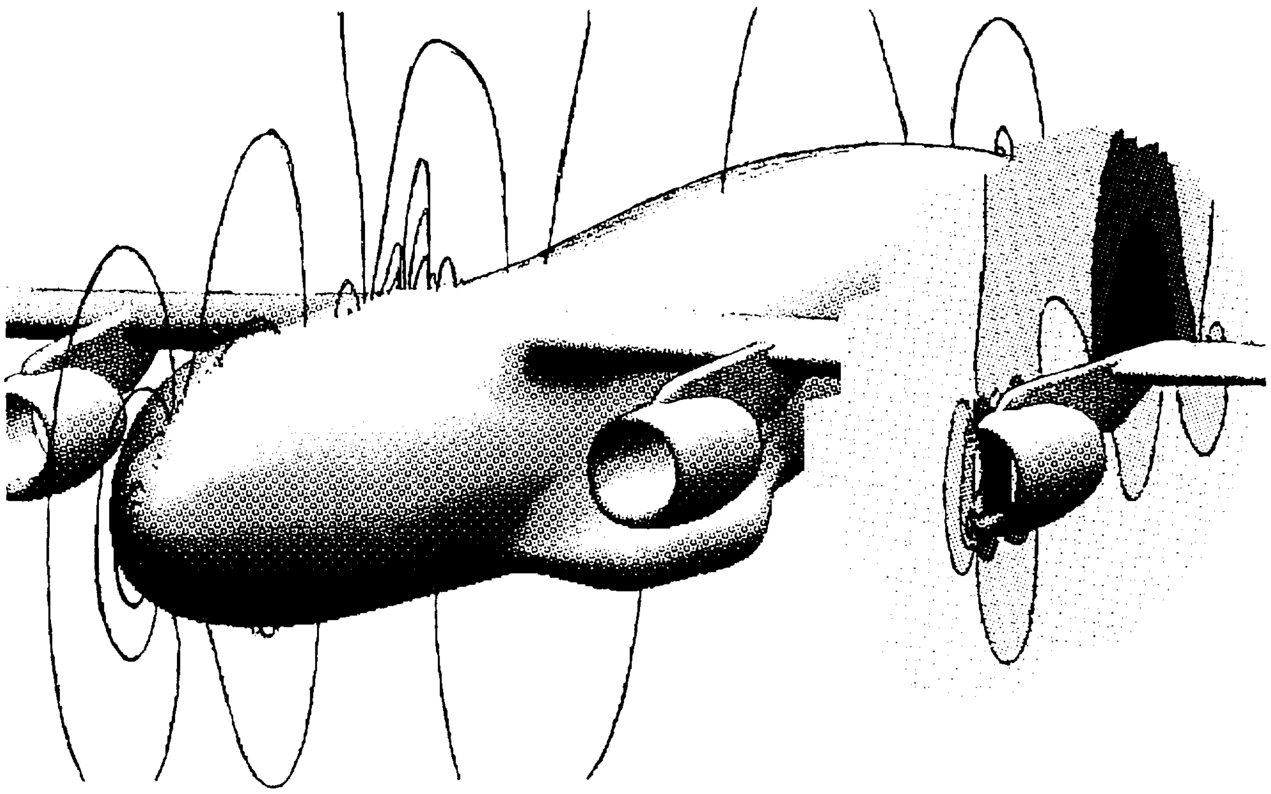
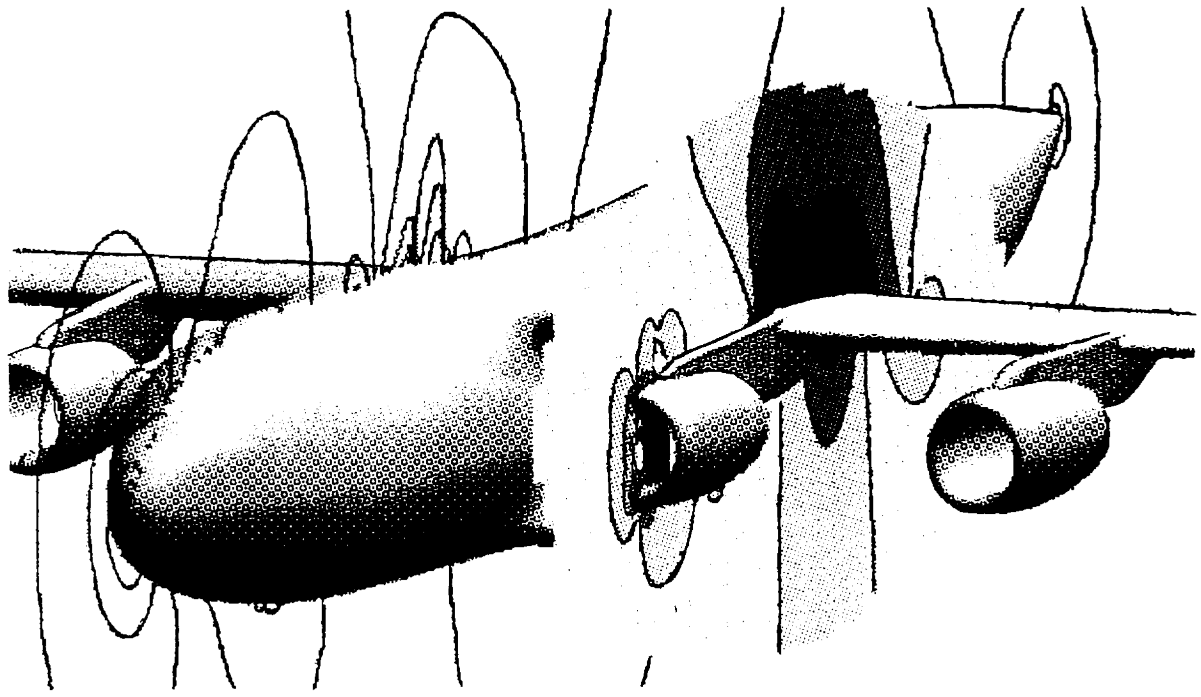
Figure 19. Pressure contours for the advanced high-wing transport. Contours are displayed in the symmetry plane and on cutting planes aligned with the inboard and outboard nacelles.

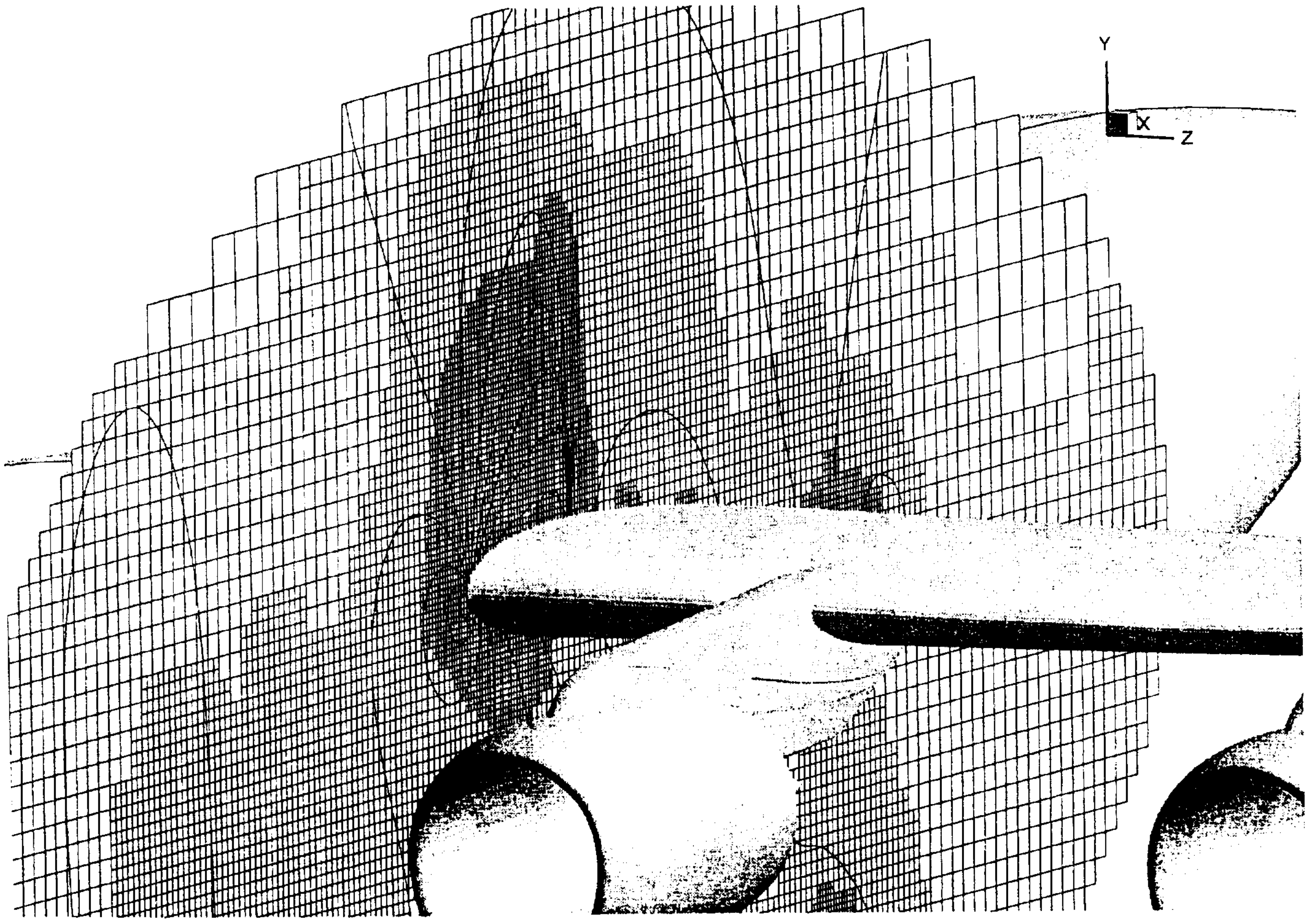

Figure 20. Close-up of wing station near inboard nacelle showing final adapted mesh and flow field pressure contours for advanced high-wing transport.

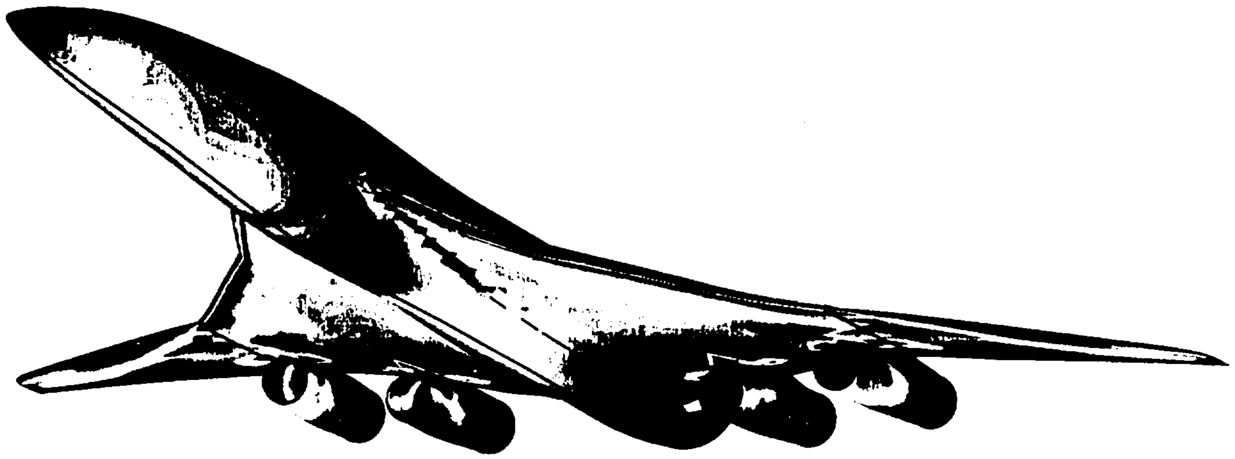

Figure 21. Surface pressure distribution for advanced supersonic transport, Mach 2.4.

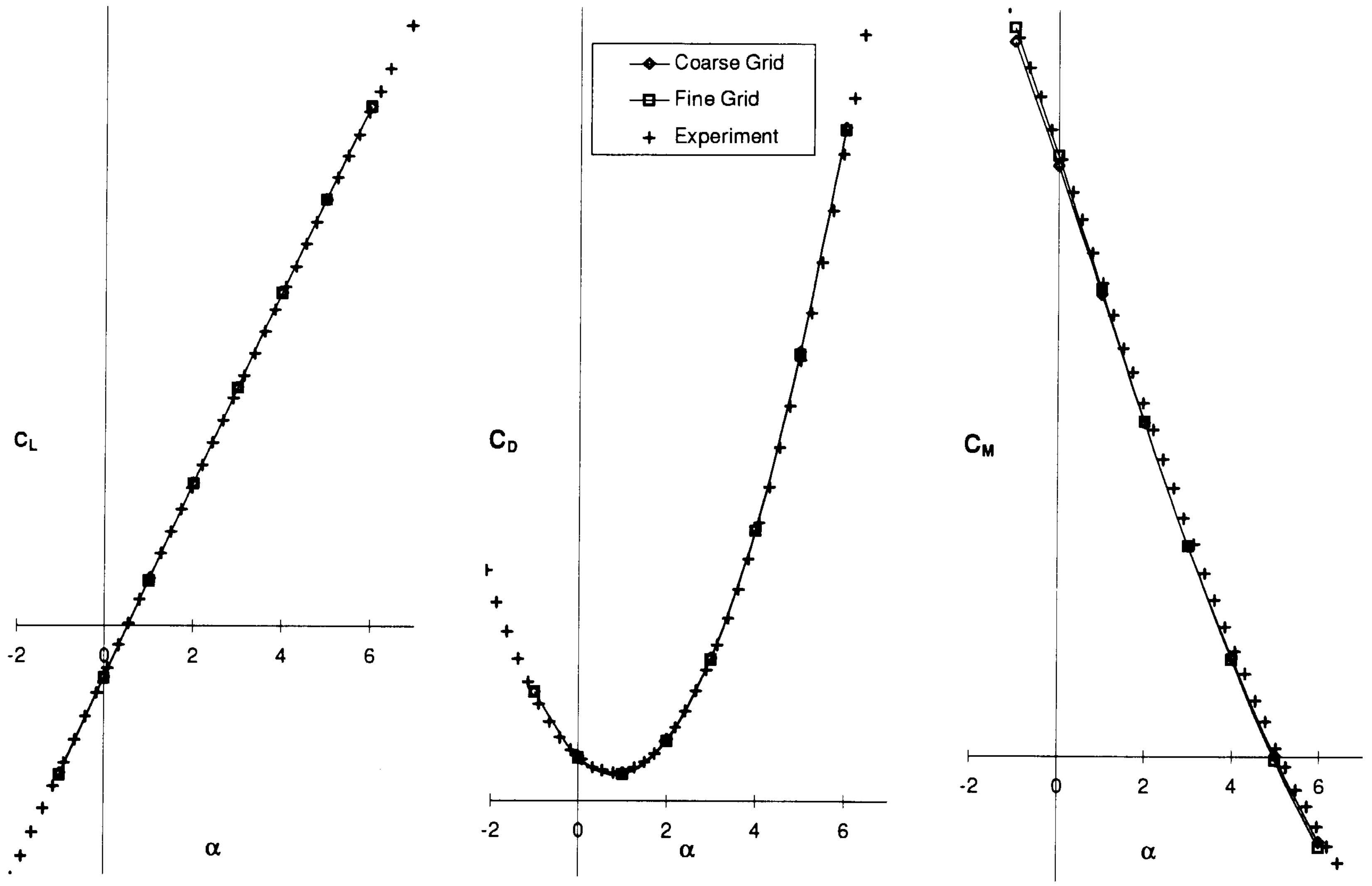

Figure 22. Lift, drag, and pitching moment for supersonic transport wing-body configuration, Mach 1.8. (CFD data corrected for skin friction drag increment)

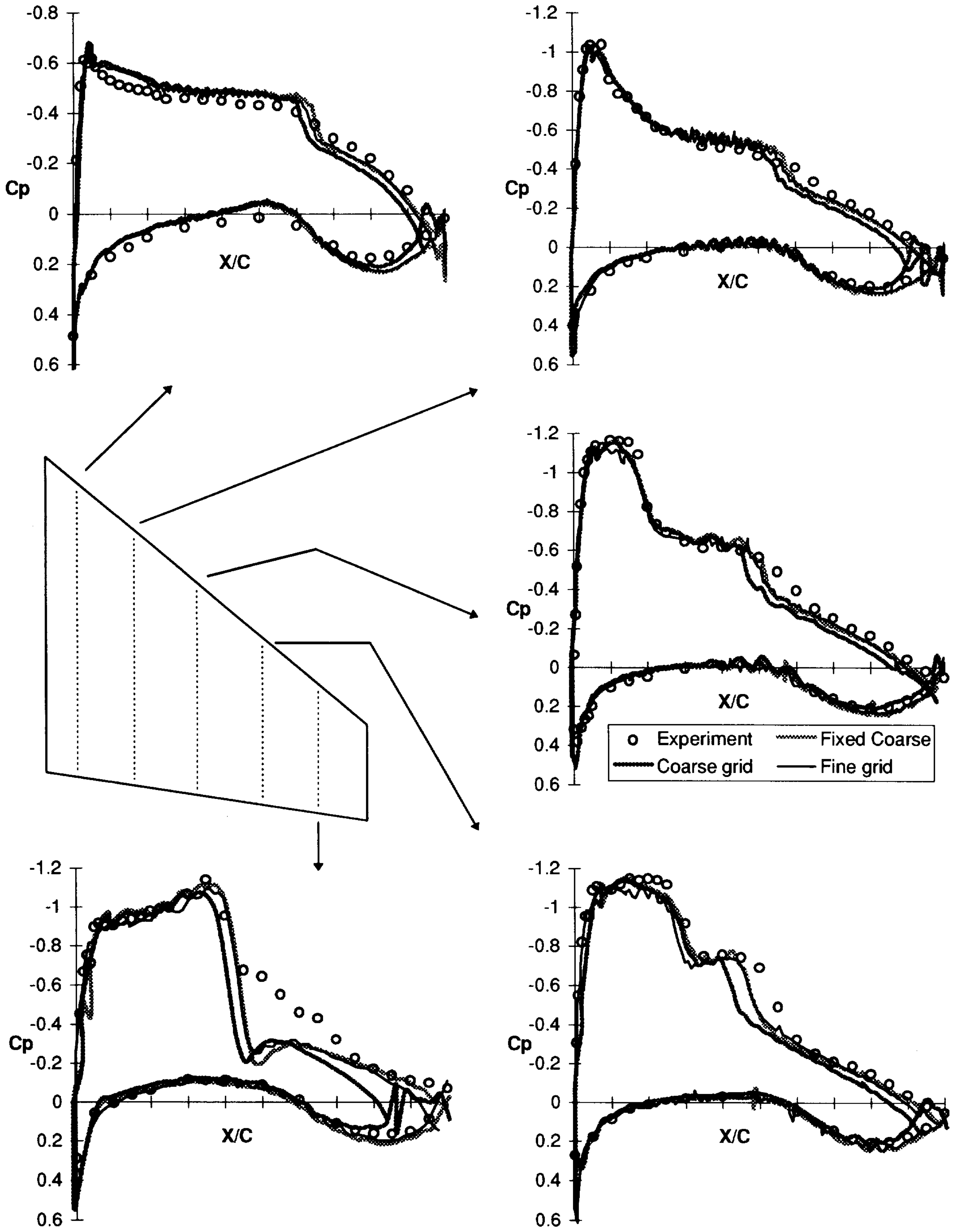

Figure 23. C_p distributions for Wing C, Mach 0.85, $\alpha = 5°$.

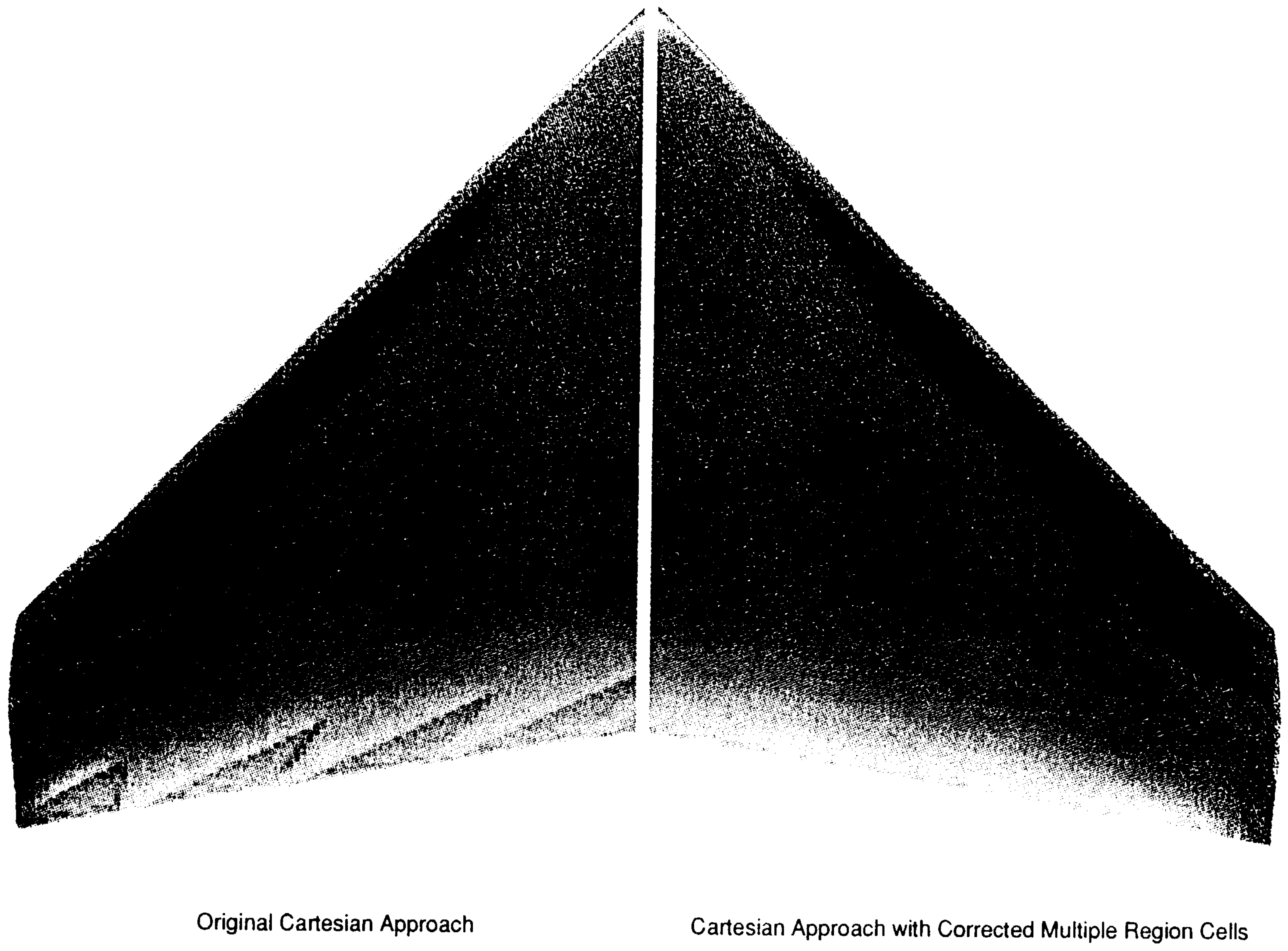

Figure 24. Effect of correcting multiple-region cells on upper surface C_p distributions for Wing C, Mach 0.85, $\alpha = 5°$.

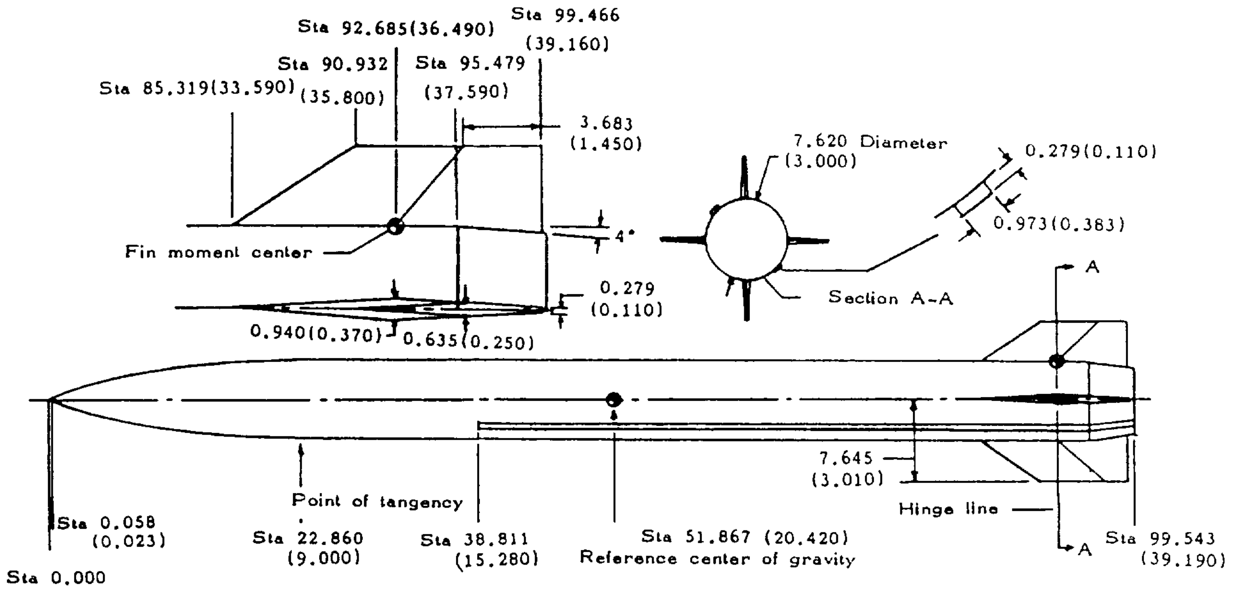

Figure 25. Wingless missile configuration (from Ref. 2). Dimensions are in centimeters (inches).

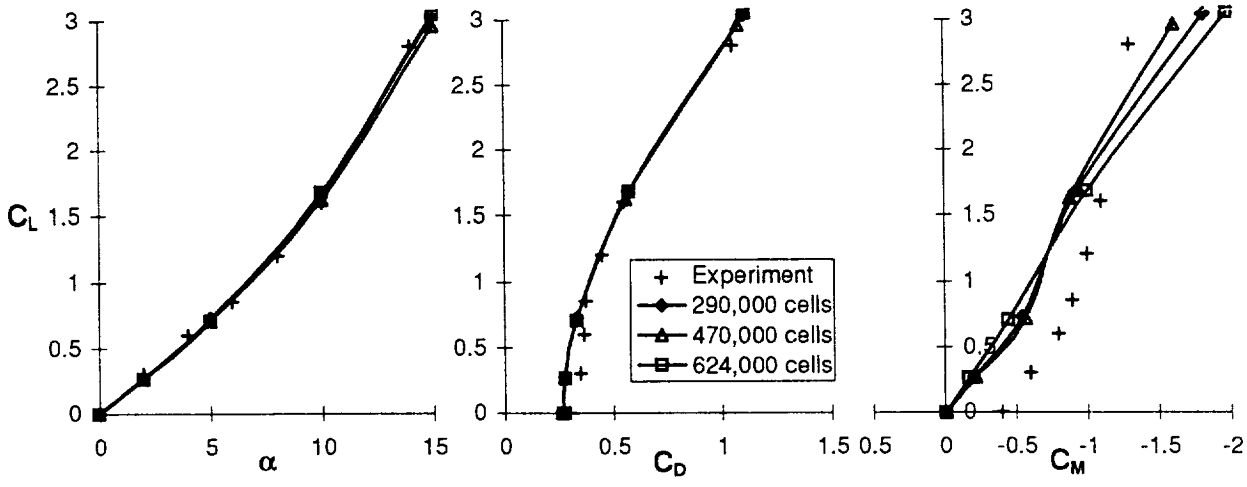

Figure 26. Lift, drag, and pitching moment coefficients for supersonic missile, Mach 2.3.

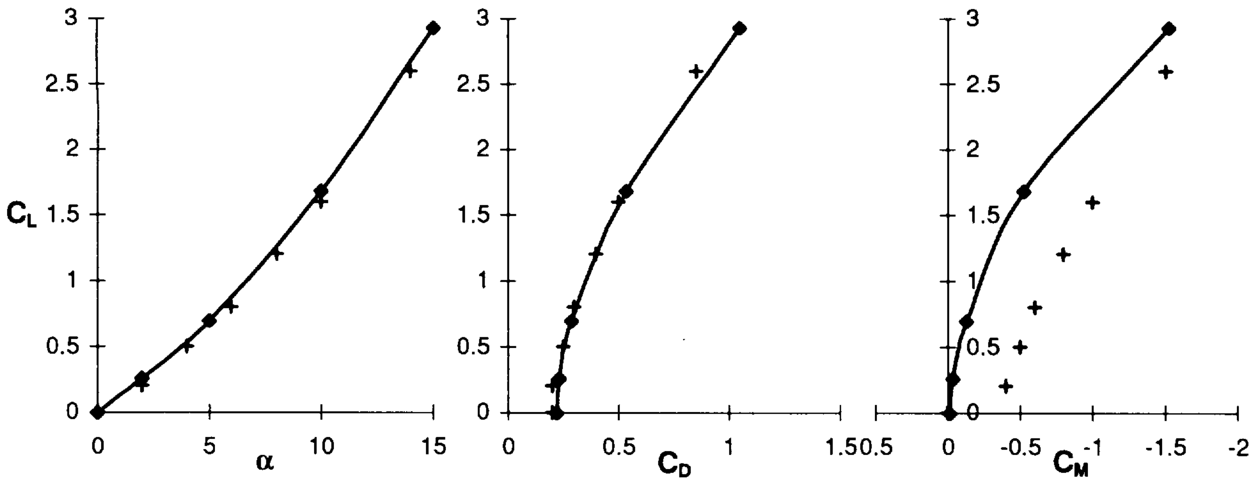

Figure 27. Lift, drag, and pitching moment coefficients for supersonic missile, Mach 2.96.

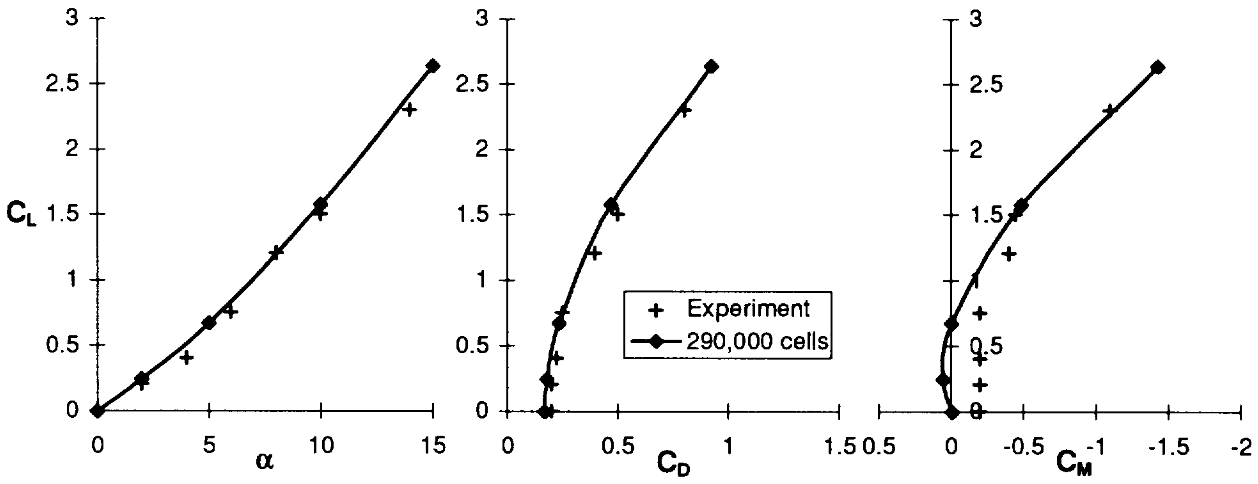

Figure 28. Lift, drag, and pitching moment coefficients for supersonic missile, Mach 3.95.

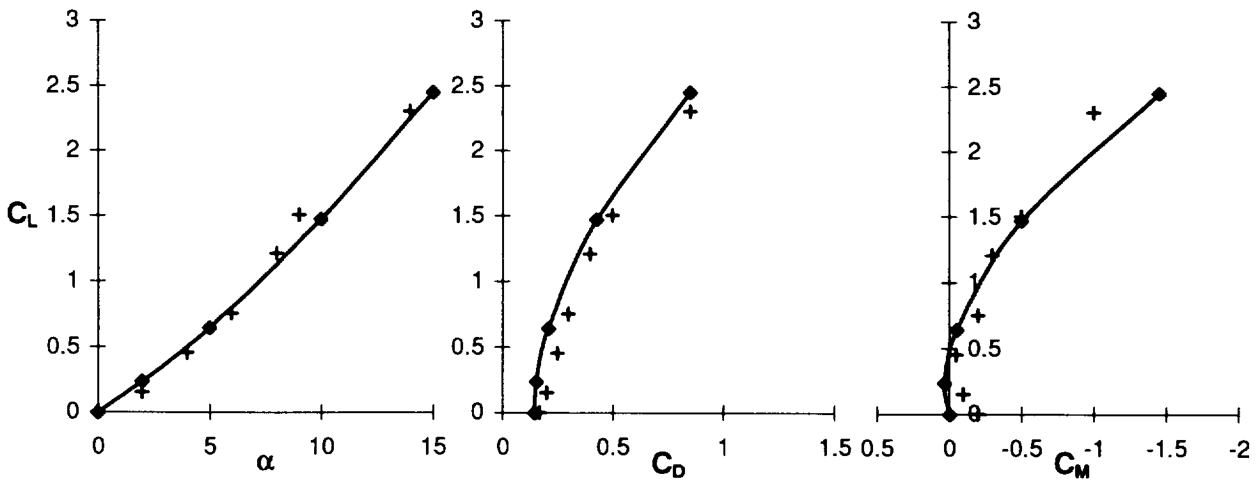

Figure 29. Lift, drag, and pitching moment coefficients for supersonic missile, Mach 4.63.