UNSTRUCTURED CARTESIAN/PRISMATIC GRID GENERATION FOR COMPLEX GEOMETRIES

Steve L. Karman Jr.*
Lockheed Fort Worth Company
Fort Worth, Texas 76101

SUMMARY

The generation of a hybrid grid system for discretizing complex three dimensional (3D) geometries is described. The primary grid system is an unstructured Cartesian grid automatically generated using recursive cell subdivision. This grid system is sufficient for computing Euler solutions about extremely complex 3D geometries. A secondary grid system, using triangular-prismatic elements, may be added for resolving the boundary layer region of viscous flows near surfaces of solid bodies. This paper describes the grid generation processes used to generate each grid type. Several example grids are shown, demonstrating the ability of the method to discretize complex geometries, with very little pre-processing required by the user.

INTRODUCTION

The CFD analysis of complex geometries is becoming more prevalent in the early phases of advanced tactical aircraft development. Unstructured grid methods are gaining popularity with the design engineers because of the reduced amount of pre-processing effort required by the CFD practitioner[1,2,3]. Cartesian unstructured methods have recently been developed in which essentially all the grid is automatically generated[4,5,7]. The flowfield around three-dimensional configurations as complex as a fully loaded fighter aircraft have been analyzed using these new Cartesian unstructured methods[10,11]. These analyses generally assume inviscid flow and capture extremely detailed features in the flowfield by using solution adaptive grid refinement. Viscous analyses, on the other hand, are prohibitive due to the large number of Cartesian cells required to resolve viscous regions, such as boundary layers.

Prismatic grid generation methods can generate meshes clustered near body surfaces[6]. Typically, these grids are generated using a method which marches the grid away from the surface in the normal direction. The resulting prismatic grid interfaces with another type of grid used to discretize the global computational domain or is converted to an unstructured grid format and combined with the external unstructured grid[8].

This paper will describe the hybrid grid approach of combining the automatic grid generation versatility of a Cartesian mesh with the efficient clustering capability of a prismatic grid. The combination enables the flow solver to compute solutions about complex geometries without the limitation of assuming inviscid flow. The methods used to generate each part of the hybrid mesh will be described in this paper. The interfacing strategy, used by the flow solver to connect the inner and outer CFD solutions, will also be discussed. Example grids will be shown which demonstrate the use of the techniques on realistic fighter aircraft configurations. No flowfield solutions are presented in this paper, although flowfield features are discernible in the final adapted grids.

---

*Engineering Group Specialist

SYMBOLS

A, a, b, c	grid adaption parameters
A_l, A_u	adaption function lower and upper thresholds
a_x, a_y, a_z	cell face areas in x-, y-, and z-directions
$\hat{n}$	unit normal at grid node
$\hat{n}_f$	unit normal at facet
Vol	cell volume
x, y, z	Cartesian coordinates
Δx, Δz, Δz	grid spacings in x-, y-, and z-directions
α	included angle
θ	angle between node normal and facet normal
ϕ	normal smoothing parameter
ψ	normal influence coefficient
ω	relaxation parameter
Subscripts	
1	minimum x-, y-, or z-coordinate of Cartesian cell
2	maximum x-, y-, or z-coordinate of Cartesian cell
c	cell centroid index
f	facet index
i	iteration counter
j	neighboring node index

SURFACE REPRESENTATION

Surface geometry is input to the Cartesian grid method and the prismatic grid method as a triangulated surface mesh. The surface mesh is typically provided by the engineering computer aided design (CAD) package used to define the configuration. By interfacing with the CAD package used by the designers directly, the time required to convert between the designers drawings and CFD surface modeling is virtually eliminated.

The surface in the CAD file is defined as a list of X, Y and Z coordinates and a connectivity in the form of three node numbers corresponding to the global indices of the forming points of each triangle. Geometry facets are oriented such that the surface normals point into the computational domain. Subsets of the facets are grouped together and are associated with a common boundary condition type, such as symmetry, characteristic far-field, surface tangent flow, etc.

The surface triangulation is assumed to be an accurate representation of the surface shape, as opposed to representing the surface using splines or surface polynomials. Therefore, smaller triangles are required to resolve high curvature regions of the geometry. The surface triangles also control the size of the local Cartesian cells in the initial grid generation process.

CARTESIAN GRID GENERATION

The Cartesian grid is generated using recursive cell subdivision. Cell subdivision is initially based on the user supplied geometry. As the CFD solution evolves, cell subdivision is based on gradients of selected adaption functions. The major elements of the Cartesian grid generation process are described in this section.

Octree Data Structure

An octree data structure is used to store information for each Cartesian cell during the recursive grid generation process. A subdivided cell produces eight new offspring cells, as shown in Figure 1. The parent cell is retained in the grid after the subdivision. The information stored for each cell consists of the global index of the parent cell, the global indices of the eight children that may exist and the grid level of the cell. The position of each offspring cell, in relation to the parent, is predetermined in the subdivision process. With this information, the neighboring cell indices can quickly be determined. In addition, many of the search procedures are made efficient using the octree data structure. For instance, the cell cutting process, described later, makes extensive use of the parent-child information to quickly traverse the data tree and determine which cells each geometry facet cuts.

Initial Grid Refinement Based on Geometry

The initial Cartesian grid is generated based on the resolution of the surface triangulation. The process begins with the generation of a cube-shaped root cell that encompasses the entire computational domain. The root cell is at grid level 1 and is subdivided in the X, Y and Z directions, resulting in eight offspring cells at grid level 2, as shown in Figure 1. The process continues with each offspring cell being recursively subdivided according to a length scale criterion.

In the length scale criterion, if the Cartesian cell length scale is larger than the length scale of any facet contained within the cell or touched by the cell, the Cartesian cell is subdivided. A Cartesian cell's length scale is defined as the length of one side of the cell. A geometry facet's length scale can be defined in a number of ways. The most common method is the average length of the three sides of the facet. Other possible definitions include the minimum length of the sides of the facet, the average length of the medians of the facet or the minimum length of the medians of the facet. The user can supply a scale factor associated with each body of the geometry that is used to multiply the facet length scale.

The initial cell subdivision process continues down each branch of the octree data structure until all cells without offspring satisfy the length scale criterion. Recursively, the new offspring cells are tested using the same criterion. The resulting grid contains cells near the boundaries that are proportional in size to the geometry facets near the cell, see Figure 2.

Grid Smoothing

During the subdivision process several grid quality constraints are enforced. The first constraint limits the number of neighbors a cell can have to no more than four neighbors on any side. This corresponds to limiting the difference in the grid levels between neighbors to one. This constraint is enforced so the octree data structure can be used to rapidly determine the neighbor information of the cells on all grid levels and to simplify the flux calculation performed by the flow solver at each face. Any refinement resulting from this constraint quickly propagates through the grid. The resulting grid varies smoothly from fine resolution cells near the bodies or high gradient regions of the flowfield to coarse resolution cells in the far field.

The second constraint improves the quality of the grid, and ultimately the numerical solution on the grid, by prohibiting rapid changes in grid levels in the mesh. Any cell with finer mesh on opposite sides in any direction is refined, as shown in Figure 3. This constraint eliminates the tendency to

generate coarse-fine-coarse grid regions in the mesh. Numerical accuracy in the flow solver is directly related to the local grid size and is adversely affected by this type of rapid change in mesh size.

Boundary Cell Cutting

Boundary conditions in the flow solver are imposed on the actual surface shape, as opposed to the stair-stepped surface resulting from the collection of Cartesian cells touching the geometry. Cells at the boundary must therefore be clipped to conform to the surface shape. This cutting process is a critical part of the Cartesian grid generation process. It must be capable of handling totally arbitrary geometries. The cutting process must also be fast, since it is performed after each grid refinement and derefinement process.

The cutting process consists of generating a list of triangular boundary facets resulting from the intersection of the geometry facets with the Cartesian cells. A brute-force approach to generating this boundary facet list would be to test each Cartesian cell for possible intersections with each geometry facet. This approach is extremely slow. A faster approach is to use the octree data structure to test each geometry facet for possible intersections with Cartesian cells. Only cells down the branches of the octree data structure in the vicinity of the geometry facet need be tested. The search procedure consists of testing the limits of a bounding box surrounding a geometry facet for possible intersections with each child of the root cell. If the bounding box intersects with or is contained by the child cell, the offspring of the child cell are then tested for possible intersections with the bounding box. The process continues, recursively, down the branch of the tree to the finest level. Cells at the finest level are then checked for true intersections with the geometry facet.

Once it has been determined that a Cartesian cell without offspring intersects a geometry facet, a list of intersection points is generated. Vertices of the geometry facet contained within the cell are also stored. The intersection points are ordered such that traversing the list of points generates a right-hand rule normal pointing into the computational domain. A centroid of the intersection points is determine by simple averaging of the collected points. The centroid will lie on the geometry facet because the facet is planar and only one facet at a time is processed in this manner. Smaller triangular boundary facets are then generated from the list of ordered intersection points and the computed centroid, see Figure 4. These boundary facets comprise a sub-region of the original geometry facet.

Each Cartesian cell may intersect an arbitrary number of geometry facets. Therefore. the resulting list of boundary facets is generally much larger than the list of geometry facets. A typical ratio of boundary facets to geometry facets for a complex configuration is on the order of 20 or greater.

The actual boundary conditions are applied on these boundary facets. Each boundary facet is associated with only one geometry facet and only one Cartesian cell. Since each geometry facet is a particular type of boundary condition, each Cartesian cell may also contain an arbitrary number of boundary condition types.

Grid Validity

The Cartesian grid generation process may result in cells that are divided into multiple distinct volumes, such as the cell near the sharp trailing edge region shown in Figure 5. A distinct volume exists above the surface and another exists below the surface. Both of the volumes reside in one Cartesian cell. Cells divided by the geometry in this way are invalid cells, since storage for only one set of conservative variables exists for each cell. Recursive subdivision of invalid cells can correct the situation; however, detecting invalid cells can be difficult and computationally expensive.

Several methods have been devised for detecting invalid cells. The most general method, and the most expensive method, involves sorting the distinct segments of the boundary facets that exist on the six faces of a boundary cell. If the segments occurring on the faces of the Cartesian cell can be

sorted into one curve, then the cell is a valid cell. If the sorting produces more the one curve, then the cell is an invalid cell. This method involves generating a list of segments for each cell, deleting duplicate segments that occur from common edges of boundary facets, and attempting to sort the remaining segments into one distinct curve. Throughout the process, an accurate grid tolerance must be used when comparing endpoints of the segments. A tolerance too small or too large could easily result in multiple closed curves and an incorrect determination of an invalid cell. Another drawback is that the method cannot distinguish between the valid case and the invalid case shown in Figure 6. This may result in excessive grid refinement. However, it may be argued from a numerical accuracy standpoint, that each of these cases should be refined. Each boundary cell can have an arbitrary number of boundary facets. But at what point is the number of boundary facets excessive? It may be wise to refine the ambiguous cases and improve the quality of the mesh in the process.

A much simpler method to detect invalid cells is to sum the X, Y and Z area components of the boundary facets in each cell. If any of the area components sum to zero when the maximum magnitude of the area components in the same direction is non-zero, then the cell may be an invalid cell. This is less precise than the first method and also cannot distinguish between the valid and invalid cases shown in Figure 6. This method will also not detect an invalid cell when the geometry cuts the cell at an angle to the Cartesian grid.

A third cell validity checking approach is a modification of the previous method. Sum the negative and positive contributions to the area components in each direction. If both negative and positive summations of significant magnitude occur in any of the three directions, then the cell may be an invalid cell. Significant magnitude may be defined as a percentage of the face area of an uncut cell. This approach will not distinguish between the valid and invalid cases in Figure 6, but it will detect invalid cells cause by the geometry cutting the cell at an angle to the Cartesian grid, as in Figure 7. This method will also force grid refinement in regions where there is a change in the orientation of surface normal for smooth geometries, such as the wing leading edge shown in Figure 7.

Solution Adaptive Refinement & Derefinement

Periodically, the grid resolution may be enhanced to capture pertinent flowfield features and to improve the solution accuracy. Adaption of the grid is based on gradients of user selected functions. There are currently ten adaption functions available to users. A few of the more commonly used adaption functions are velocity magnitude, Mach number, pressure, and helicity. Directional adaption parameters are computed for each cell for each selected function, *f*, as

$$a = \frac{\partial f}{\partial x}(\Delta x)^{1+g},\ b = \frac{\partial f}{\partial y}(\Delta y)^{1+g},\ c = \frac{\partial f}{\partial z}(\Delta z)^{1+g}$$

$$A = \sqrt{a^2 + b^2 + c^2}$$

The variable in the exponent of the length scale multiplier is used to decrease the effect of discontinuities in the flowfield on the grid adaption[9]. Generally, *g* is set to a value of 1.0. A value of 0.0 corresponds to computing the gradients of the function in computational space. Thus, a shock would produce a constant valued adaption function as the grid was refined and could inhibit refinement from taking place in smoother regions of the flow where large cells exist. Values of g greater than 1.0 would lessen the effect of discontinuities even further and have been used successfully in fully supersonic flowfields to force more rapid grid refinement of the entire flowfield.

The magnitude, *A*, of the adaption parameters is statistically averaged. Then, during the derefinement process, a lower threshold value of the adaption function is computed as

$$A_l = m - d\sigma ,$$

where m is the mean, σ is the standard deviation and d is a user defined constant, typically set to 1.0. Any cell, without children, with all three directional adaption parameters less than A_l is marked for deletion. If all eight children of a parent cell are marked for deletion, then the children are deleted from the cell list. The parent cell would reclaim the collective volume of the computational domain previously occupied by the children.

During the refinement process, an upper threshold value of the adaption function is computed as

$$A_u = m + e\sigma .$$

The variable e is another user defined constant typically set to 1.0. Any cell, without children, with any one of the directional adaption parameters greater than A_u is marked for refinement.

Additional Considerations

The refinement process could, conceivably, continue to add cells to the grid indefinitely. Therefore, some limits must be imposed. The first limit is on the minimum and maximum cell size allowed in the grid. No cell can be refined with a length scale less than the minimum cell size specified by the user and no cell can exist without children with a length scale greater than the maximum cell size specified by the user. These cell size limits are usually unnecessary because of the next limiting process imposed on the refinement process.

The user specifies a "target" number of cells desired in the final grid. This target must be slightly less than the maximum dimension of the code in order to allow the code to enforce the smoothness criteria mentioned earlier. The user can also specify a maximum number of new cells added per adaption function during each grid refinement process. Limiting the number of cells added per refinement allows the solution to evolve slowly between refinements. This prohibits the grid from reaching the maximum number of cells prematurely, before the flowfield can adequately develop. Additionally, the user may specify a refinement box which delineates a region in the computational domain where adaptive grid refinement takes place.

During the refinement process the cells are ranked from highest to lowest, according to adaption parameter A. The marking of the cells for refinement proceeds from the top of the list down until the estimated total number of cells exceeds the target maximum number of cells or the estimated number of new cells added exceeds the specified maximum number added per adaption function. Thus, if insufficient deletion took place to reduce the current number of cells below the target maximum, no refinement would take place. In this case, the multiplier on the standard deviation in the lower threshold value could be reduced or the target number of cells could be increased. The effect of these limiting processes is to produce a near optimum grid for the target number of cells and the selected adaption functions.

Cell Face Areas, Volumes and Centroids

The flow solver for the CFD code that uses this Cartesian mesh requires accurate computation of the face areas and volumes of all cells. The higher order extrapolation procedure also requires the location of the centroid of each face and the centroid of the cell. For uncut cells the face areas, volumes and centroids are easily computed.

$$a_x = \Delta y \Delta z, a_y = \Delta x \Delta z, a_z = \Delta x \Delta y$$

$$x_c = \frac{(x_1 + x_2)}{2}, y_c = \frac{(y_1 + y_2)}{2}, z_c = \frac{(z_1 + z_2)}{2},$$

$$Vol = \Delta x \Delta y \Delta z$$

For cube shaped cells, the spacing in each direction, Δx, Δy, Δz, are equal. The centroids of the faces are the appropriate components of the cell centroid, (x_c, y_c, z_c).

The boundary facets contain the necessary information for computing the face areas, centroids and volumes of the cut cells. Edges of boundary facets that exist on each face of a cut cell are used to compute the exposed area and centroid of the face. The face areas and the boundary facets are then used to compute the cell volume and cell centroid.

An example of the exposed area computation of a face of a cut Cartesian cell is shown in Figure 8. The shaded area in the top figure represents the portion of the face contained inside the geometry. The unshaded region is the exposed area or the portion of the face existing in the computational domain. An elaborate procedure has been developed to compute the exposed area of the face. The procedure involves summing areas swept out by the edges of the boundary facets that exist on the face.

A summation point is selected, such as the lower left corner of the face shown. The areas swept out by the two vertices of each edge and the summation point are summed. The ordering of the edge points determines whether the computed area is positive or negative. The positive area that results from this process is shown as the initial area of the face.

The two vertices of each edge are then projected to the top and right boundaries of the face. The "anti-areas" swept out by the projected vertices and the summation point are summed. The ordering of the projected points determines whether the computed area is positive or negative. The negative area that results from this process is shown as the anti-area of the face.

The area and anti-area of the face are combined to produce the actual exposed area of the face. In this case the combination produces the negative area shown as the combined area. Adding this negative area to the area of the uncut face produces the area of the exposed region shown as the resulting area. The same procedure is applied on each of the faces of cut cells that are intersected by boundary facets.

The centroids of the faces are computed by summing the area weighted centroids of the swept out regions during the previously described steps and dividing the result by the actual face area. The individual swept out regions are triangles, so the centroids are simply the average of the three forming nodes of the triangle.

Volumes of cut cells are then computed by summing volume of sub-components of the cell. A cell summation point is selected, such as the centroid of the uncut Cartesian cell. Tetrahedra are formed from the summation point and the forming nodes of the boundary facets. The volume of each tetrahedra contributes to the total volume of the cell. The orientation of the facet forming nodes in relation to the summation point determines whether the computed tetrahedral volume is positive or negative. Prismatoids are formed from the summation point and the area of each face of the cell. The volume of each prismatoid is positive and contributes to the total volume of the cell. An example of the sub-components of a cut cell are shown in Figure 9.

Cell centroids are computed from a volume weighted average of the centroids of the sub-components of the cell just described. The centroids of the tetrahedra are simply the average of the four forming nodes. The centroid of the prismatoid is assumed to be the point one fourth the distance from the face centroid to the cell summation point.

PRISMATIC GRID GENERATION

The prismatic grid is generated by marching a triangulated surface mesh outward. The major elements of the process are the surface triangulation, the computation of the surface normals at each grid point, the marching step size, and the prevention of grid crossing in concave and convex regions of the surface.

Surface Triangulation

The surface triangulation produced by the CAD program described earlier is used in the development of the prismatic grid. In this case, however, the resolution of the surface mesh is more than a description of the geometry. It is the starting layer of the prismatic grid. Therefore, facet resolution in critical regions other than high curvature regions is important. The prismatic grid is currently not refined by the CFD code during the solution process, so adequate resolution of the surface in critical regions must be supplied by the surface triangulation.

Surface Normals

The triangular surface facets are marched outward along carefully computed normal vectors from each node point, see Figure 10. The normal vector at each node must be constructed such that it is visible to each facet containing that node[6]. This ensures that the developing grid layers in convex regions do not cross and produce negative prismatic cell volumes.

The initial normal vector at each node is computed as the weighted average of the normals of the common faces

$$\hat{n}^0 = \frac{\sum_{f}^{mf} \hat{n}_f \alpha_f}{\sum_{f}^{mf} \alpha_f},$$

where $\hat{n}_f$ is the unit normal of common faces, and α_f is the included angle of the face as shown in Figure 10. The variable *mf* is the number of faces surrounding the node.

An iterative procedure is then used to improve the normal vector at each node using a linear combination of a weighted average of the normal vectors of the common faces and a weighted average of the position vectors of the neighboring nodes projected to the next layer.

$$\hat{n}^{i+1} = \hat{n}^i + \omega \left((1-\phi) \left(\frac{\sum_{f}^{mf} \alpha_f \hat{n}_f \psi_f^i}{\sum_{f}^{mf} \alpha_f \psi_f^i} \right) + \phi \left(\frac{\sum_{j}^{nn} \dot{r}_j d_j}{\sum_{j}^{nn} d_j} \right) \right)$$

$$\psi_f^i = 1 + \theta.$$

The weighted average of the face normals is once again based on the included angle of each common face. The common face normal vector, $\hat{n}_f$, is multiplied by ψ, which is a function of the angle between the current node normal vector and the face normal vector, θ. The position vector of the neighboring nodes projected to the next level is given by $\dot{r}_j$ and is weighted by the distance to the node, d_j. The variable ϕ varies from 0.0 to 0.5 and controls the amount of smoothing of the normal vector. With no smoothing, the scheme tends to minimize the angle between the resulting node normal and common

face normals. With smoothing, the grid expands as it grows outward, approaching a mesh where the tangential spacing between the nodes is equidistant. A typical value for the smoothing parameter is 0.25. An under-relaxation factor, ω, is used to update the new normal vectors. The iterative procedure continues until the changes in the normal vectors are negligible or a maximum iteration limit is reached.

The use of the included angle in the averaging procedure eliminates the tendency to get skewed vectors for cases where a node has many small included angle facets on one side. This situation may occur at sharp trailing edges. This normal calculation procedure has been tested on many complex shapes and has produced valid surface vectors.

Additional control is added in extreme convex and concave regions of a geometry. The smoothing portion of the equation is turned off for the nodes where the angle between the face normals common to any given facet edge exceeds a user specified amount, such as 60 degrees. The normals at such nodes are then improved using only the face normal portion of the iterative procedure.

Marching Step Size

The grid is advanced to the next layer by a specified spacing increment. The spacing increment can be equally spaced, generated from a prescribed normal distribution or based on the minimum radius of an inscribed circle for the facets on each layer. The marching step size at extremely concave nodes is slightly increased. The marching step size at extremely convex nodes is slightly decreased. These adjustments help to improve the quality of the grid at the next layer. The effect of the adjustments is to reduce the maximum angle between face normals on each subsequent grid layer, i.e. smoothing out the waviness.

The total thickness of the prismatic grid in the normal direction is usually just large enough to resolve the estimated boundary layer thickness for viscous analyses or large enough to improve the invalid cell situation at sharp edges for inviscid analyses. At each new layer the normals at each node are recomputed using the above scheme and the process continues until the desired number of prismatic layers is completed. Any portion of the boundary layer in viscous analyses not resolved by the prismatic grid will be resolved by the refined Cartesian grid using the grid adaption scheme.

Prevention of Grid Crossing

Grid crossing in convex regions of the surface is prevented by carefully computing the surface normals, as described earlier. Grid crossing in concave regions of the surface is controlled by reducing the marching step size. As each layer is marched outward, the current grid layer is checked for grid crossing. When grid crossing is detected, the local marching step size is reduced. The marching step sizes across the current layer are then smoothed using a Laplacian type smoothing. The nodes are projected outward once again and rechecked for grid crossing. This process continues until the crossing is eliminated.

CARTESIAN/PRISMATIC GRID INTERFACE

The Cartesian grid treats the outer layer of the prismatic grid as another boundary. The Cartesian grid generation, cell cutting and so forth are performed in the usual manner, since the outer layer of the prismatic grid is made up of triangular elements like all other defined geometries. During the solution process, the prismatic grid and the Cartesian grid interact through the flux calculations at the interface boundary. Fluxes are computed at the interface boundary facets using data from the interior of the prismatic grid and the local Cartesian grid cells. Fluxes are computed at the interface boundary using the same upwind or central differencing schemes as used with the Cartesian grid faces and the interior prismatic grid faces. The computed fluxes are distributed to each side of the interface in a fully conservative manner.

EXAMPLE GRIDS

Several example cases are included to demonstrate the capability of the grid generation procedures to discretize complex geometries and complex flowfields. Some of the cases use only the Cartesian grid, while one case uses the hybrid grid system. The final case includes only the prismatic grid for a complex store configuration. Flowfield solutions were obtained for all of the Cartesian grid-only examples and the single hybrid grid example. Grid refinement of the Cartesian grid was used to resolve pertinent flowfield features in these cases. Since this paper pertains to the grid generation aspect of the problems, no flowfield solution data will be shown.

F16 Forebody

An inviscid supersonic flowfield for a F16 forebody/inlet geometry was computed using the Cartesian grid system. The surface model included extensive geometric detail, including the inlet duct back to the compressor face, the diverter section between the inlet and the underside of the fuselage, and the CD band antenna on the underside of the nose. Grid refinement was based on two adaption functions, Mach number and total pressure. The purpose of the analysis was to gain an understanding of the shock structure and total pressure field entering the inlet. Qualitative results from this analysis helped guide engineers in grid generation for a structured grid Navier-Stokes solution. A symmetry plane cut through the mesh is shown in Figure 11. Adaption to Mach number is evident in the increased grid resolution at the shocks, while the adaption to total pressure is responsible for the refinement aft of the CD band antenna. The final grid contained 601,513 Cartesian cells.

F16 With Various Store Loadings

Several solutions were computed for a full F16 configuration with various types of weapons loadings. The F16 aircraft geometry included accurate modelling of the inlet duct back to the compressor face, the nozzle duct forward to the turbine face, the ventral fin, the wing with tip missile rail, and the horizontal and vertical tails. The Cartesian grids for three different weapons loadings are shown in Figure 12, Figure 13, and Figure 14. An inviscid transonic flow was computed in each case and the adaption functions were velocity magnitude and static pressure.

The weapons shown in Figure 12 include a 600 gallon fuel tank and 3 CBU58s mounted on a triple ejector rack. The constant-X cutting plane located aft of the wing trailing edge reveals the improved resolution of the three-dimensional shock structure on the wing upper surface. The final grid contained 953,385 Cartesian cells.

Figure 13 shows a AIM 9 missile along with the fuel tank and the CBU58s. The constant-X cutting plane is located at the mid-point of the wing leading edge and shows the increased resolution of the region about the CBU58s. A total of 865,993 Cartesian cell are contained in the final grid.

The additional components shown in Figure 14 include the AIM 9, a 370 gallon fuel tank, an ALQ119 ecm pod and a MK84 with pylon. Increased grid resolution about the MK84 store is evident in the figure. This analysis was part of a quasi-steady trajectory analysis of the MK84 store separation. Steady-state solutions were computed for several instances along the trajectory path. The computed forces and moments were used, in conjunction with a six degree of freedom package, to compute the new store locations. The process is analogous to the wind tunnel testing technique known as Captive Trajectory Simulation.

Wing/Pylon/Store

A hybrid grid was used to discretize the computational domain surrounding a wing/pylon/store configuration. The geometry was a clipped 45 degree swept delta wing with a pylon located at mid-span and a generic store[10,11,12]. This analysis used a prismatic grid about the store combined with a

Cartesian grid for the wing, pylon and the remainder of the domain. The extent of the prismatic grid was limited because of the close proximity of the store to the base of the pylon. The prismatic grid consisted of 8,838 cells per layer in 5 cell layers for a total of 44,190 cells. The final Cartesian grid consisted of 307,361 cells in 17 grid layers. Periodic grid refinement on the Cartesian grid was performed based on the gradients of Mach number and pressure. A side view of the Cartesian grid is shown in Figure 15. A refinement box that extended just above the wing geometry was used to limit the Cartesian grid refinement to the region in the vicinity of the store. The prismatic grid about the store is shown in Figure 16. Less than half of the gap between the store and the pylon is discretized by the prismatic grid. The remaining gap was discretized with the Cartesian grid. The use of the prismatic grid about the store resulted in approximately half as many Cartesian cells as required for a Cartesian-only solution.

MK84 Store

A prismatic grid was generated for an isolated MK84 store geometry, see Figure 17. The MK84 geometry is very similar to the generic store shown in the previous case, e.g. a cylindrical body of revolution with four tail fins. A five layer prismatic grid was generated for demonstration purposes only. No solution has been obtained for this geometry. The thickness of the layers was restricted to approximately 0.5 inches, due to the 90 degree corners at the fin/body juncture. The grid was equally spaced in the direction normal to the surface. Grid smoothing is necessary to ensure the prismatic elements march out of the corner regions properly. The marching step size for each layer also has to be limited to prevent grid crossing in the corners. A thicker prismatic grid is possible with additional layers, but would make viewing the grid more difficult than it currently is. Had this been an actual prismatic grid for a viscous analysis, the number of layers would be larger, and the normal grid spacing would be clustered toward the surface, as can be seen in the 21 layer grid shown in Figure 18 and Figure 19.

CONCLUSIONS

The methods for generating unstructured Cartesian meshes and triangular-element prismatic meshes have been described. Example grids for various configurations have been shown for Cartesian-only grids, hybrid grids, and prismatic-only grids. Grid adaption for the Cartesian grid has been demonstrated in several of the examples where a flowfield solution was generated using the CFD solver developed by the author. Highly three-dimensional flowfield features are apparent in the meshes, as the refinement scheme detected gradients in the selected adaption functions. The flexibility of the prismatic grid was demonstrated in an actual hybrid grid solution for a wing/pylon/store geometry where a prismatic grid was employed around the store. The resulting inviscid solution made more efficient use of fewer Cartesian cells to resolve the remainder of the domain. A prismatic grid suitable for viscous solutions was shown for a complex store geometry. The ability to control the thickness and normal distribution of the prismatic layers was demonstrated.

REFERENCES

1. Frink, N. T., Parikh, P., and Pirzadeh, S., "A Fast Upwind Solver for the Euler Equations on Three-Dimensional Unstructured Meshes," AIAA-91-0102.

2. Potsdam, M. A., Intemann, G. A., Frink, N. T., Pirzadeh, S., "Wing/Pylon Fillet Design Using Unstructured Mesh Euler Solvers," AIAA-93-3500.

3. Pirzadeh, S., "Viscous Unstructured Three-Dimensional Grids by the Advancing-Layers Method," AIAA-94-0417.

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

5. Melton, J. E., Enomoto, F. Y., Berger, M. J., "3D Automatic Cartesian Grid Generation for Euler Flows," AIAA-93-3386-CP.

6. Kallinderis, Y., Ward, S., "Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations," AIAA-92-2721-CP.

7. Melton, J. E., Pandya, S. A., Steger, J. L., "3D Euler Flow Solutions using Unstructured Cartesian and Prismatic Grids," AIAA-93-0331.

8. Ward, S., Kallinderis, Y., "Hybrid Prismatic/Tetrahedral Grid Generation for Complex 3-D Geometries," AIAA-93-0669.

9. Warren, G. P., Anderson, W. K., Thomas, J. L., Krist, S. L., "Grid Convergence for Adaptive Methods," AIAA-91-1592.

10. Karman, S. L. Jr., "SPLITFLOW: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries," AIAA-95-0343.

11. Welterlen, T. J., Karman, S. L. Jr., "Rapid Assessment of F-16 Store Trajectories Using Unstructured CFD," AIAA-95-0354.

12. Parikh, P., Pirzadeh, S., Frink, N. T., "Unstructured Grid Solutions to a Wing/Pylon/Store Configuration Using VGRID3D/USM3D", AIAA-92-4572.

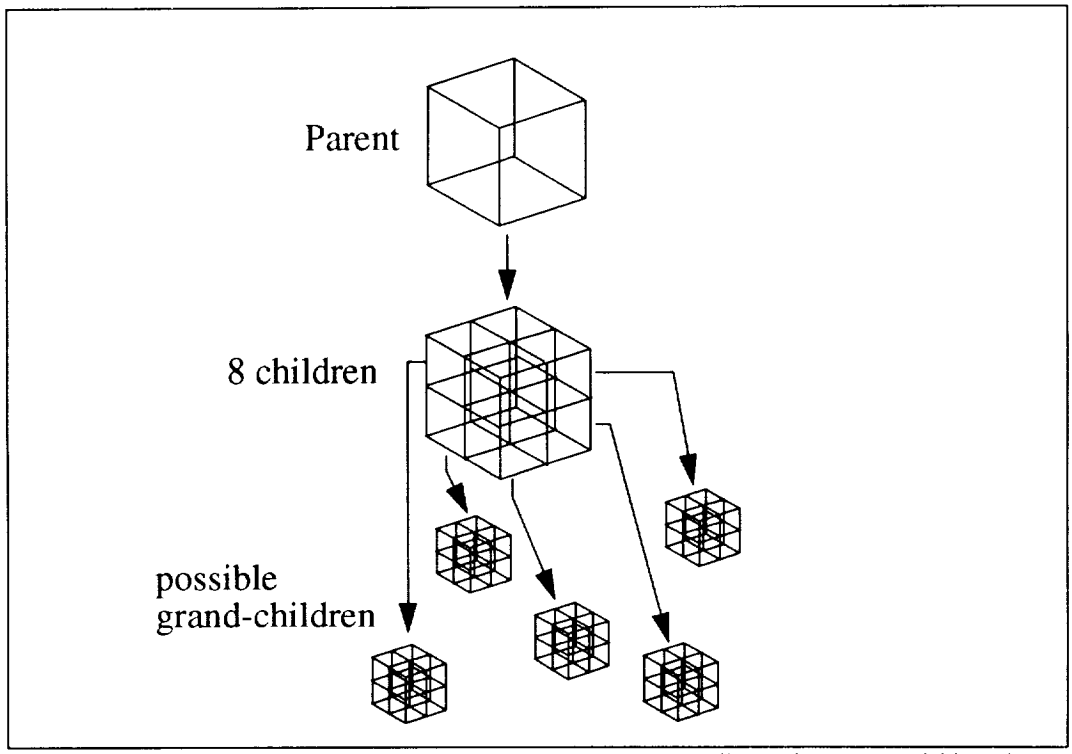

Figure 1. Each cell subdivision results in eight new cells at the next grid level using an octree data structure.

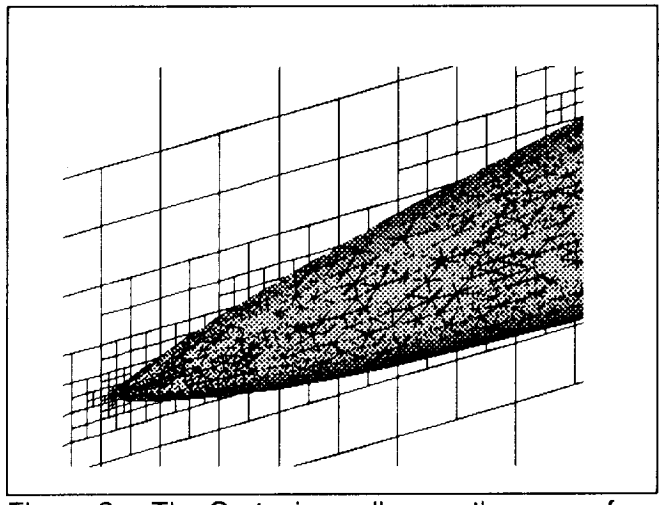

Figure 2. The Cartesian cells near the nose of a fighter configuration are proportional in size to the local geometry facets.

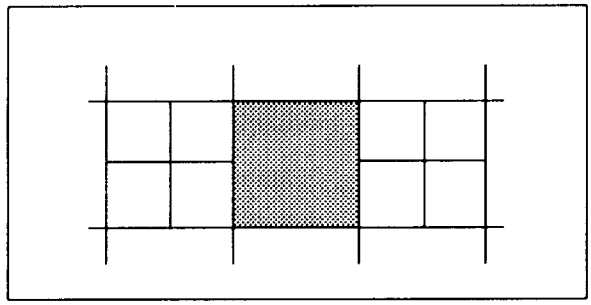

Figure 3. Cells with finer mesh on opposite sides are refined.

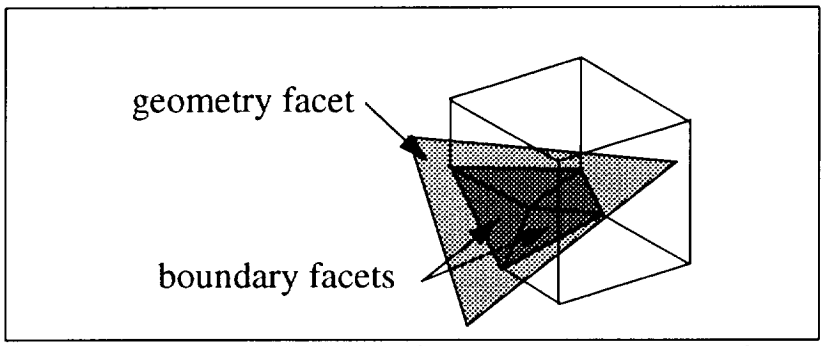

Figure 4. The cutting process for any given "geometry" facet may produce multiple "boundary" facets.

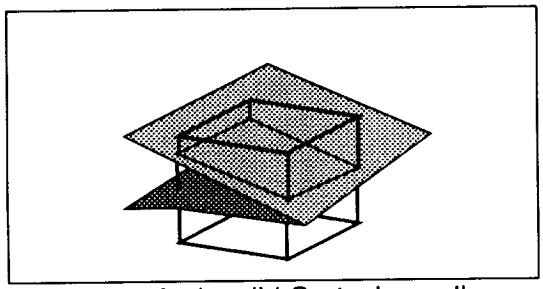

Figure 5. An invalid Cartesian cell caused by the cutting of a sharp trailing edge.

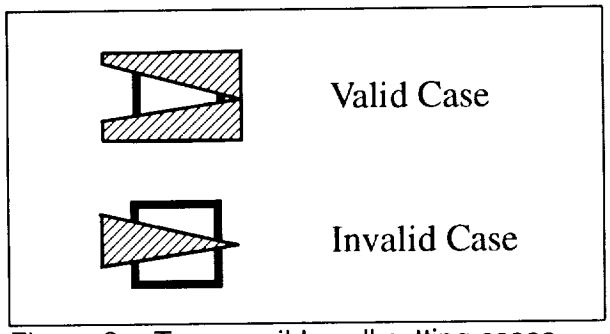

Figure 6. Two possible cell cutting cases.

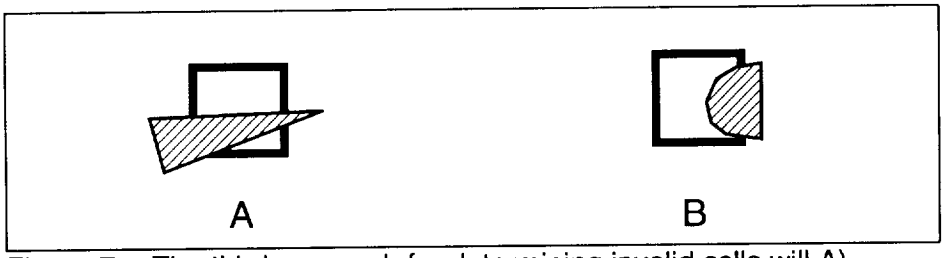

Figure 7. The third approach for determining invalid cells will A) correctly detect the cutting of cells at an angle to the Cartesian grid and B) detect changes in surface orientation.

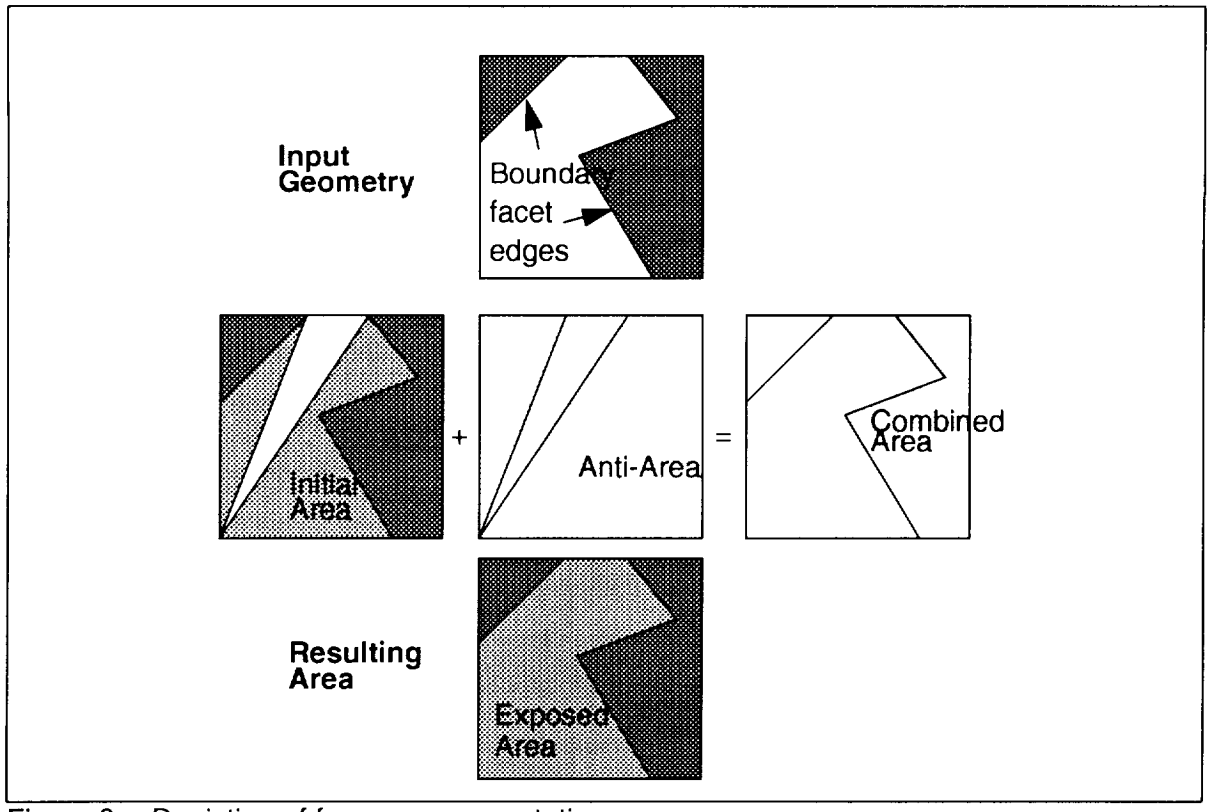

Figure 8. Depiction of face area computation process.

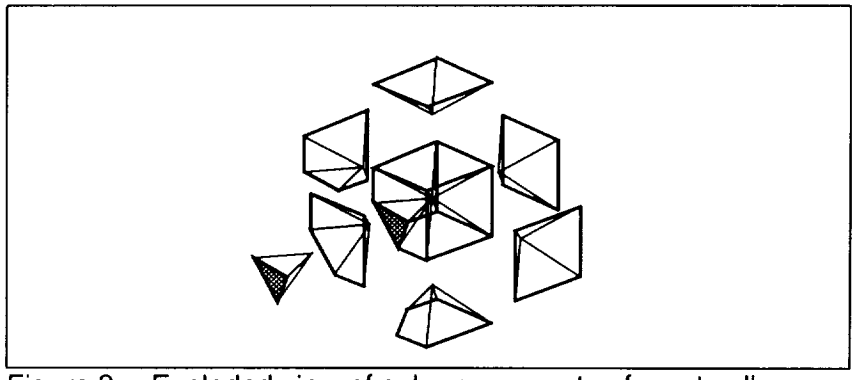

Figure 9. Exploded view of sub-components of a cut cell containing a single boundary facet.

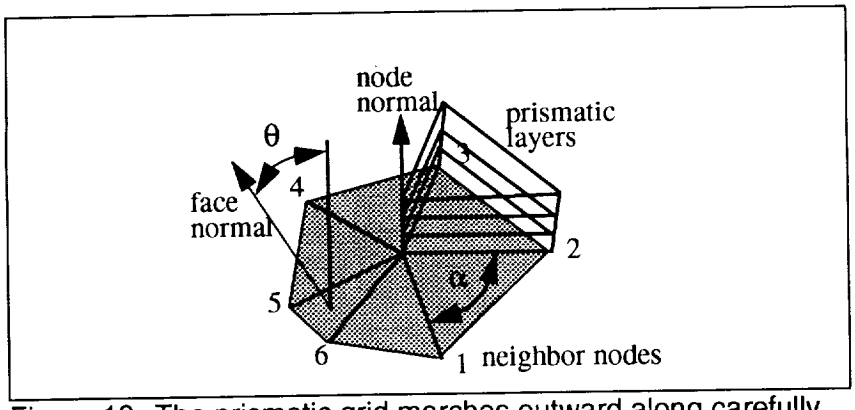

Figure 10. The prismatic grid marches outward along carefully computed normal vectors.

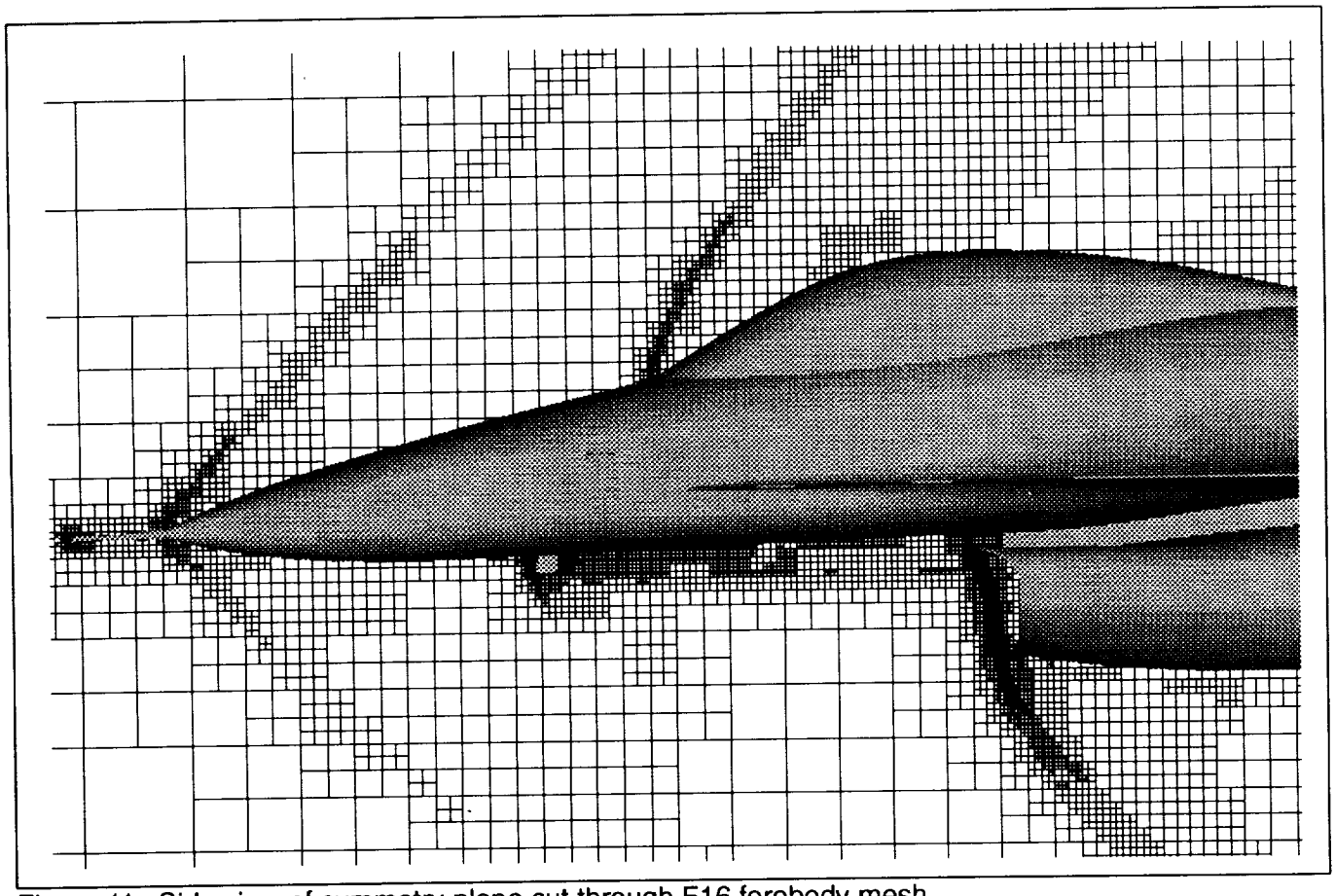
Figure 11. Side view of symmetry plane cut through F16 forebody mesh.

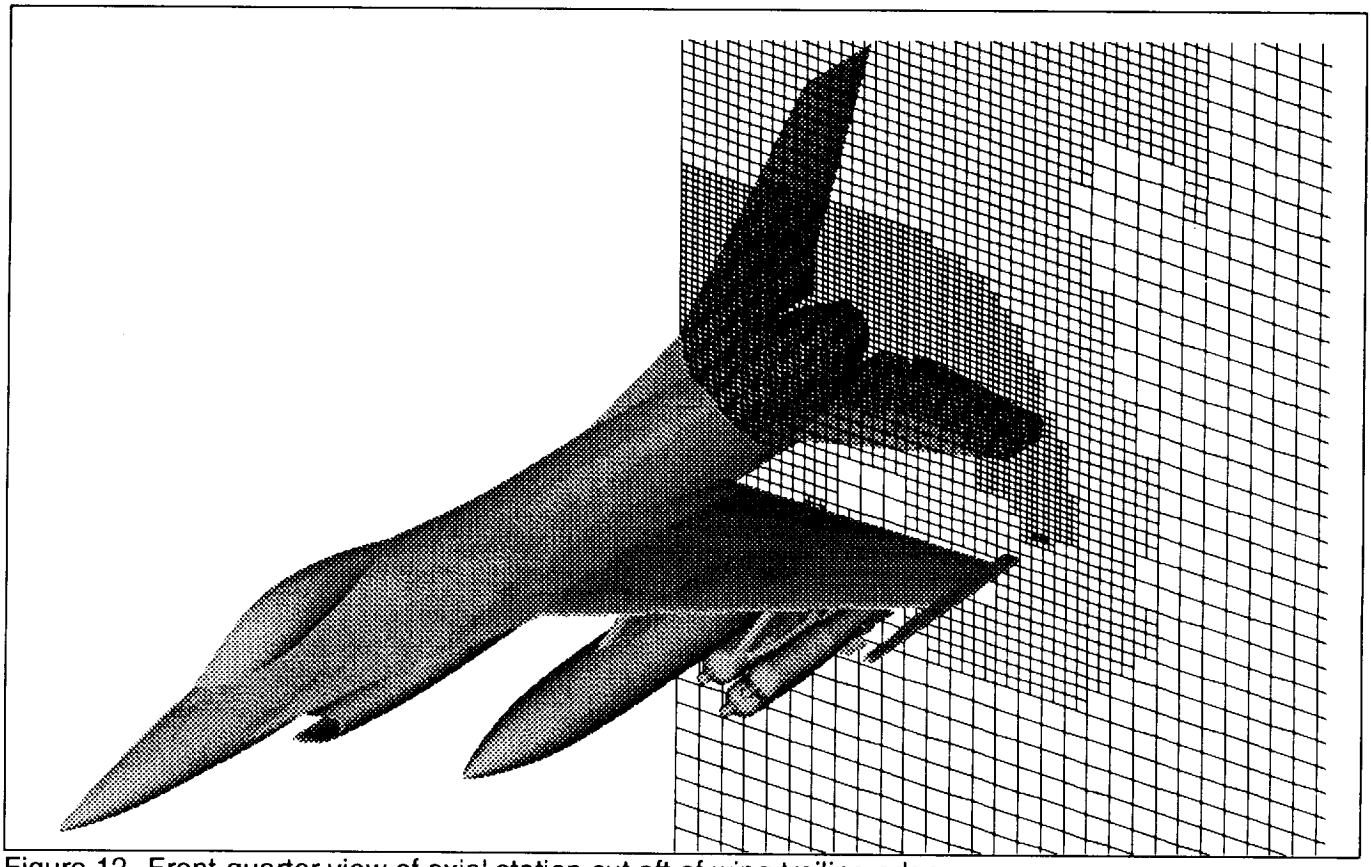

Figure 12. Front-quarter view of axial station cut aft of wing trailing edge.

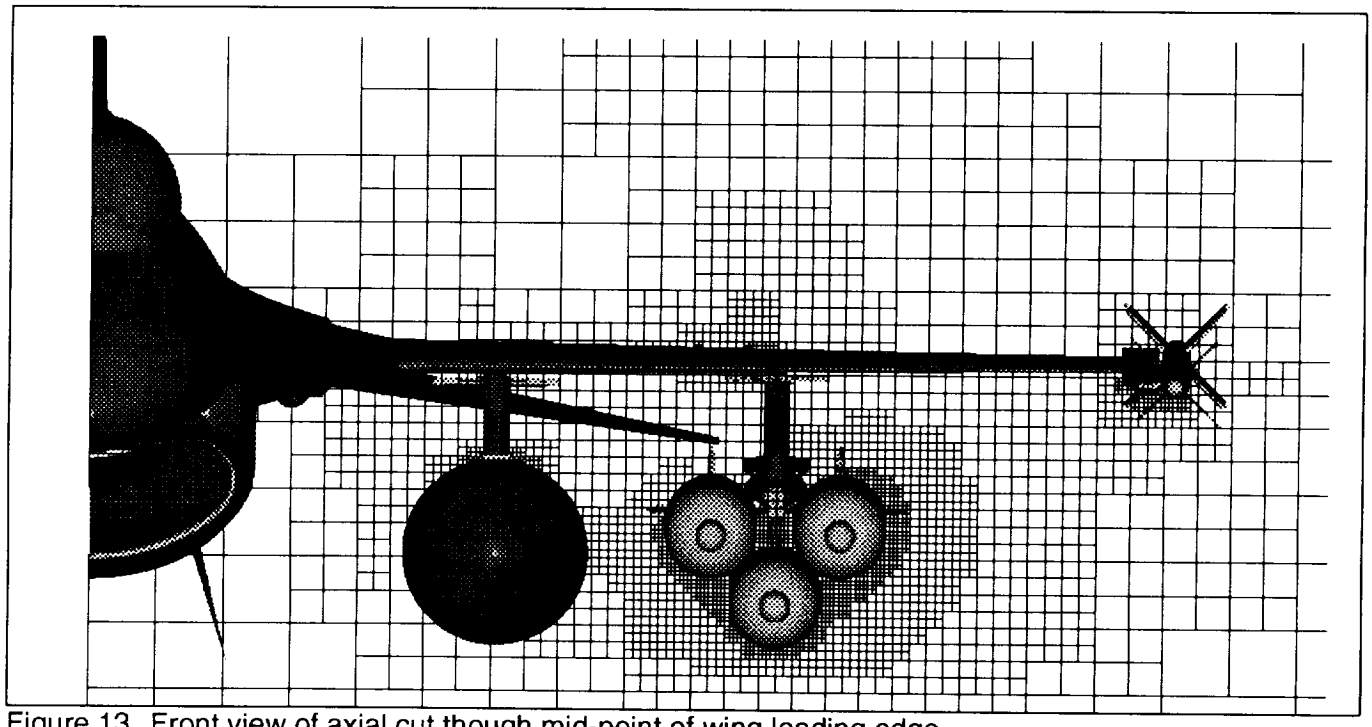

Figure 13. Front view of axial cut though mid-point of wing leading edge.

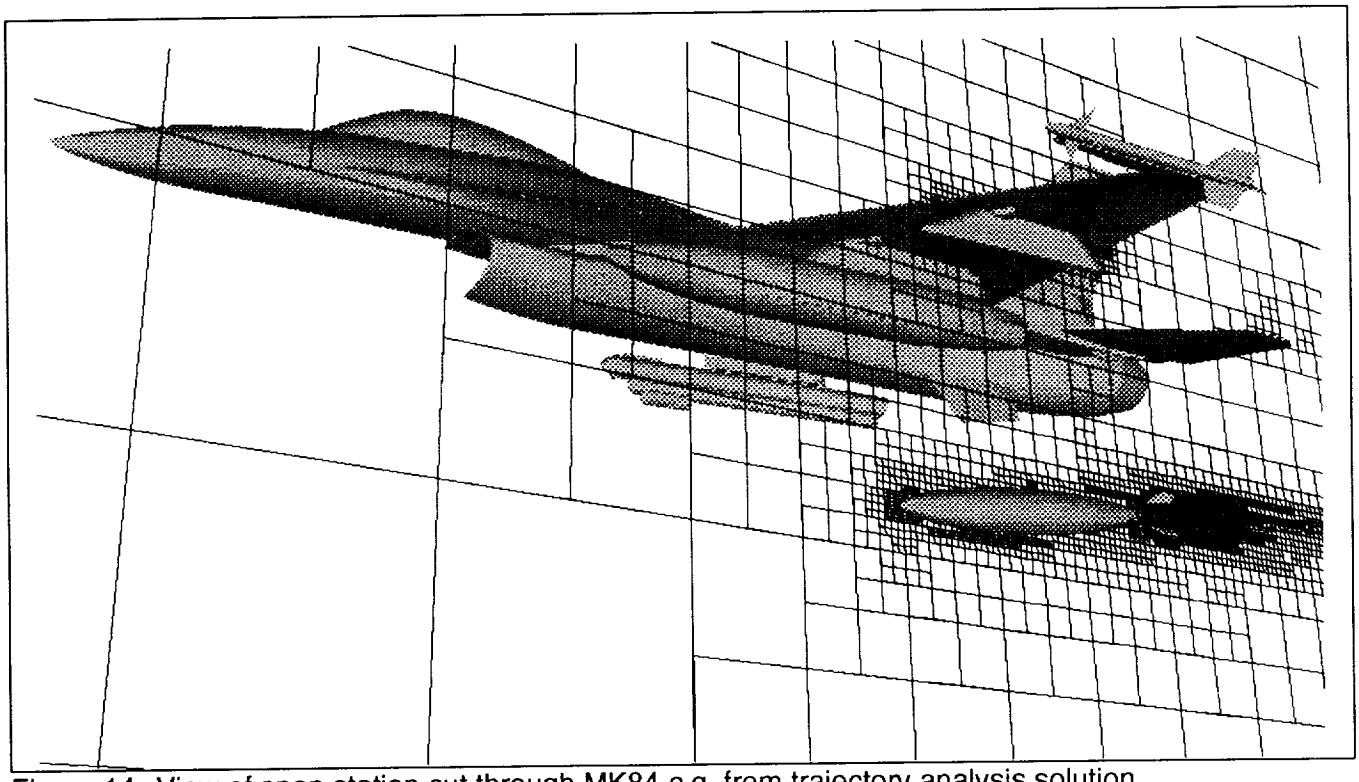

Figure 14. View of span station cut through MK84 c.g. from trajectory analysis solution.

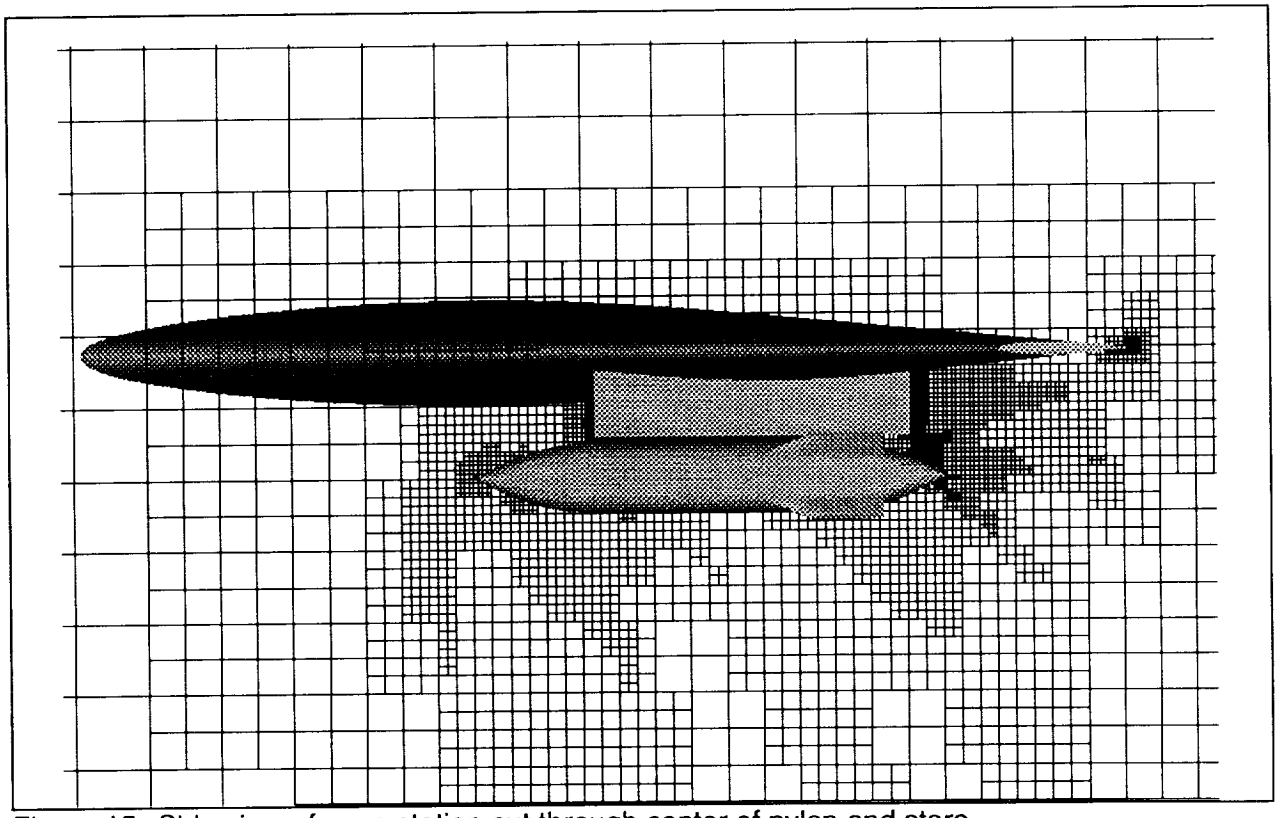

Figure 15. Side view of span station cut through center of pylon and store.

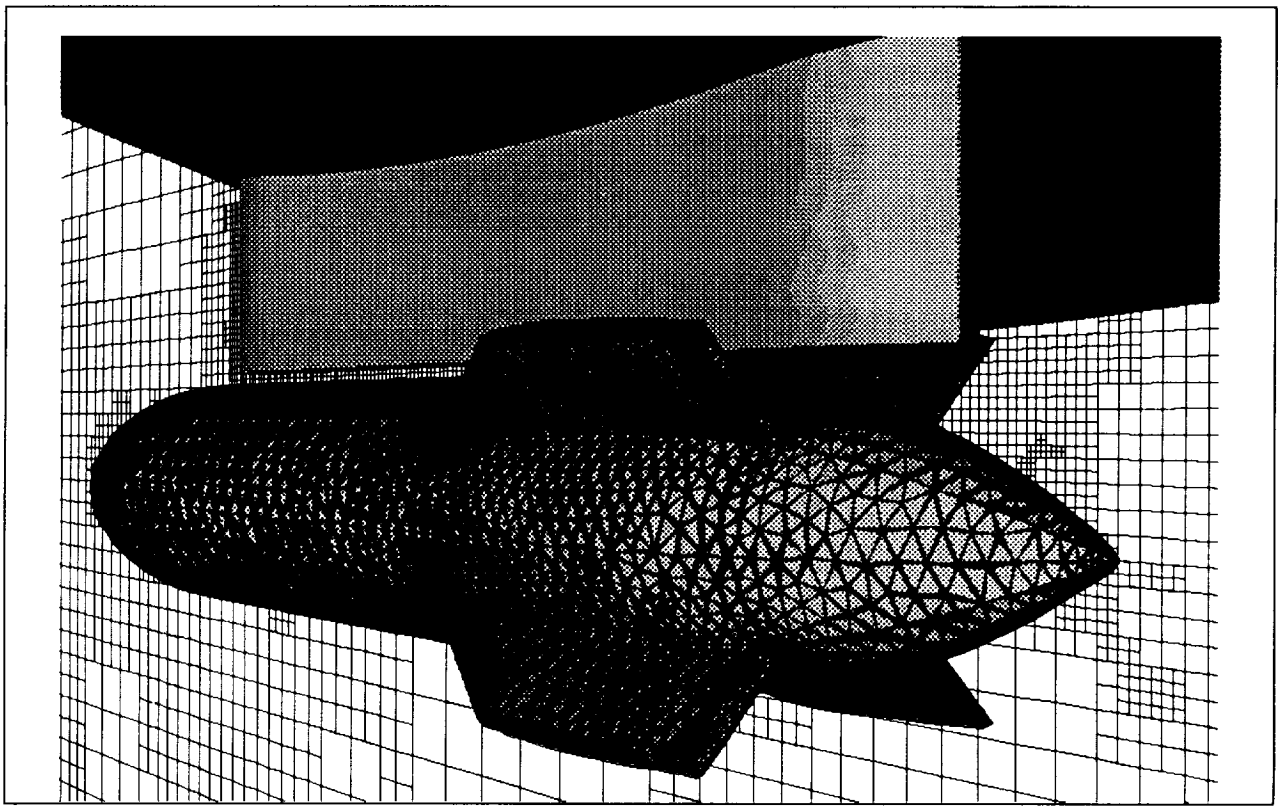

Figure 16. Prismatic grid about store displayed with pylon span station cutting plane through Cartesian grid.

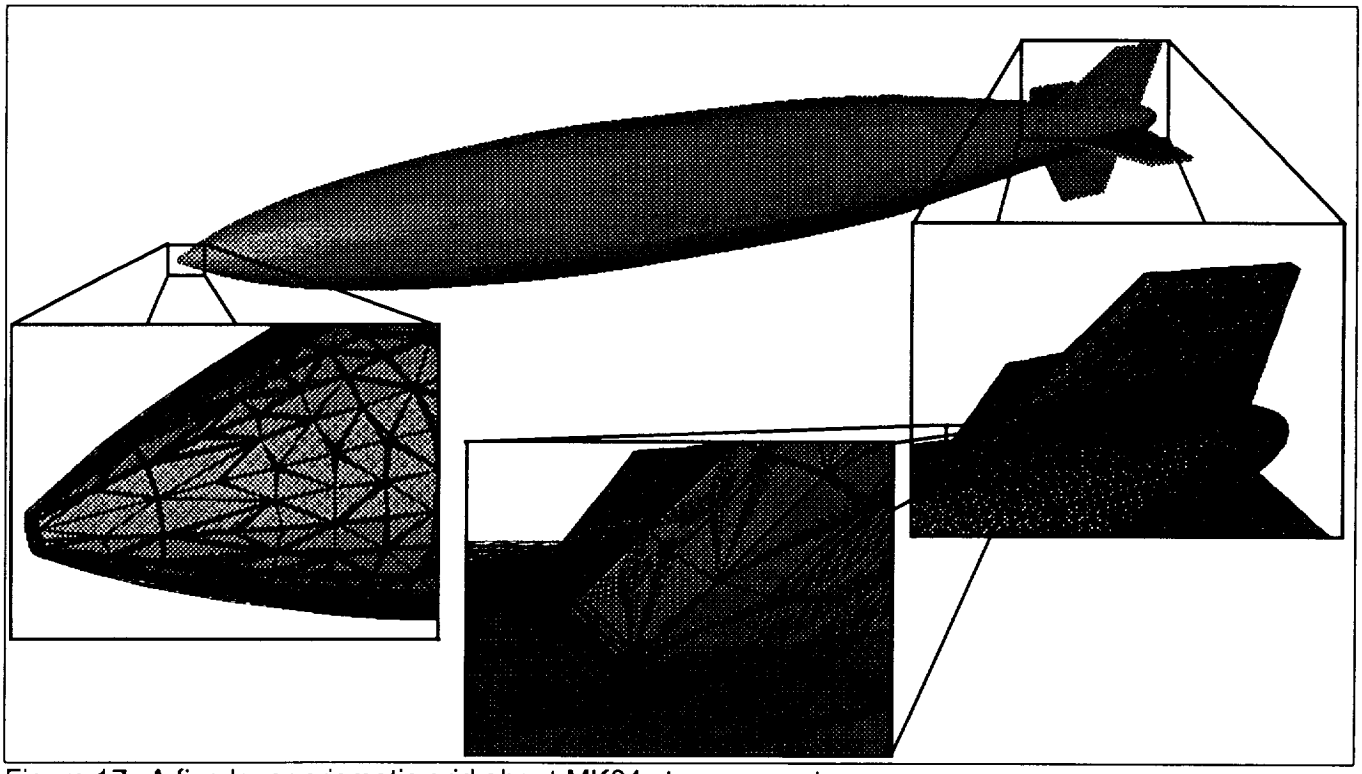

Figure 17. A five layer prismatic grid about MK84 store geometry.

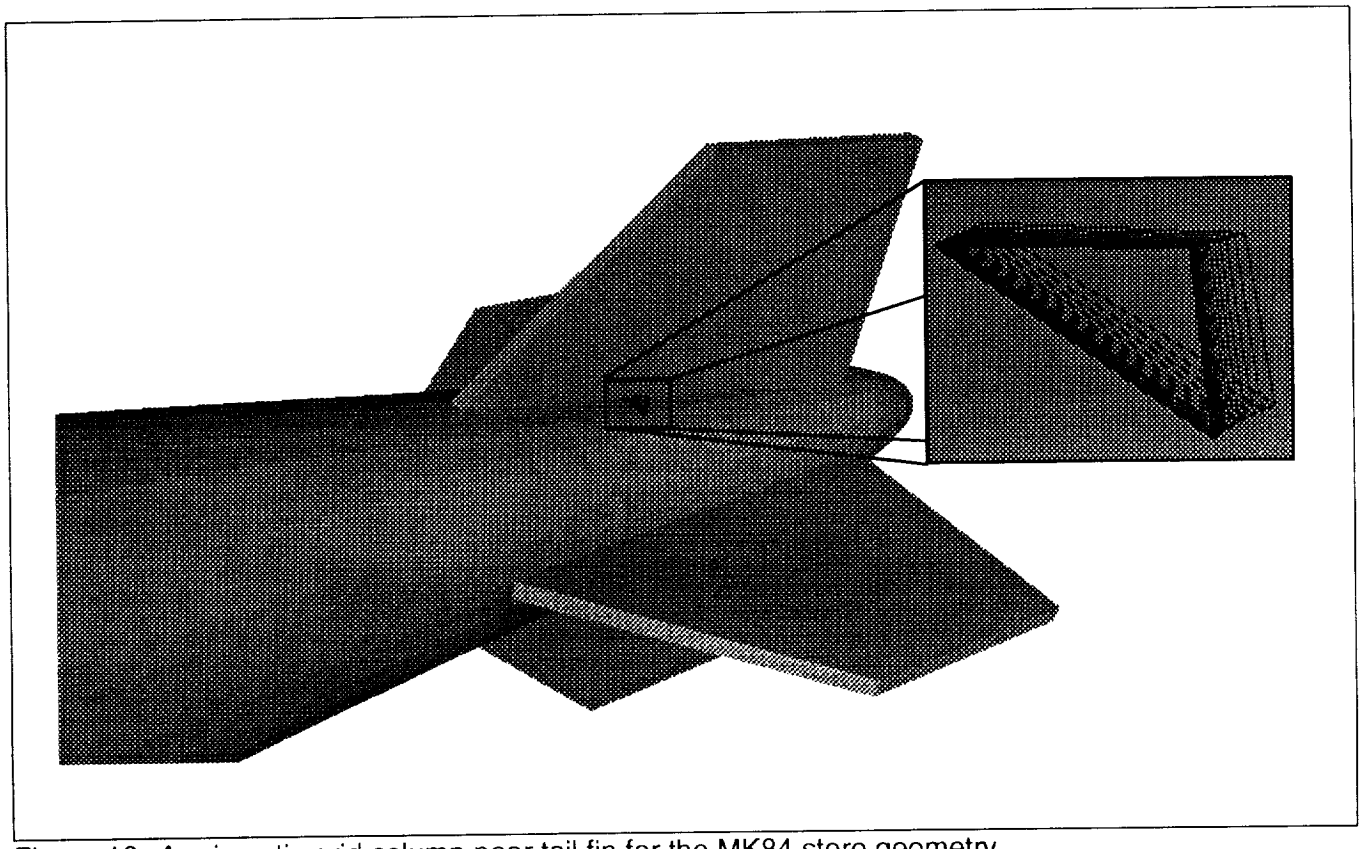

Figure 18. A prismatic grid column near tail fin for the MK84 store geometry.

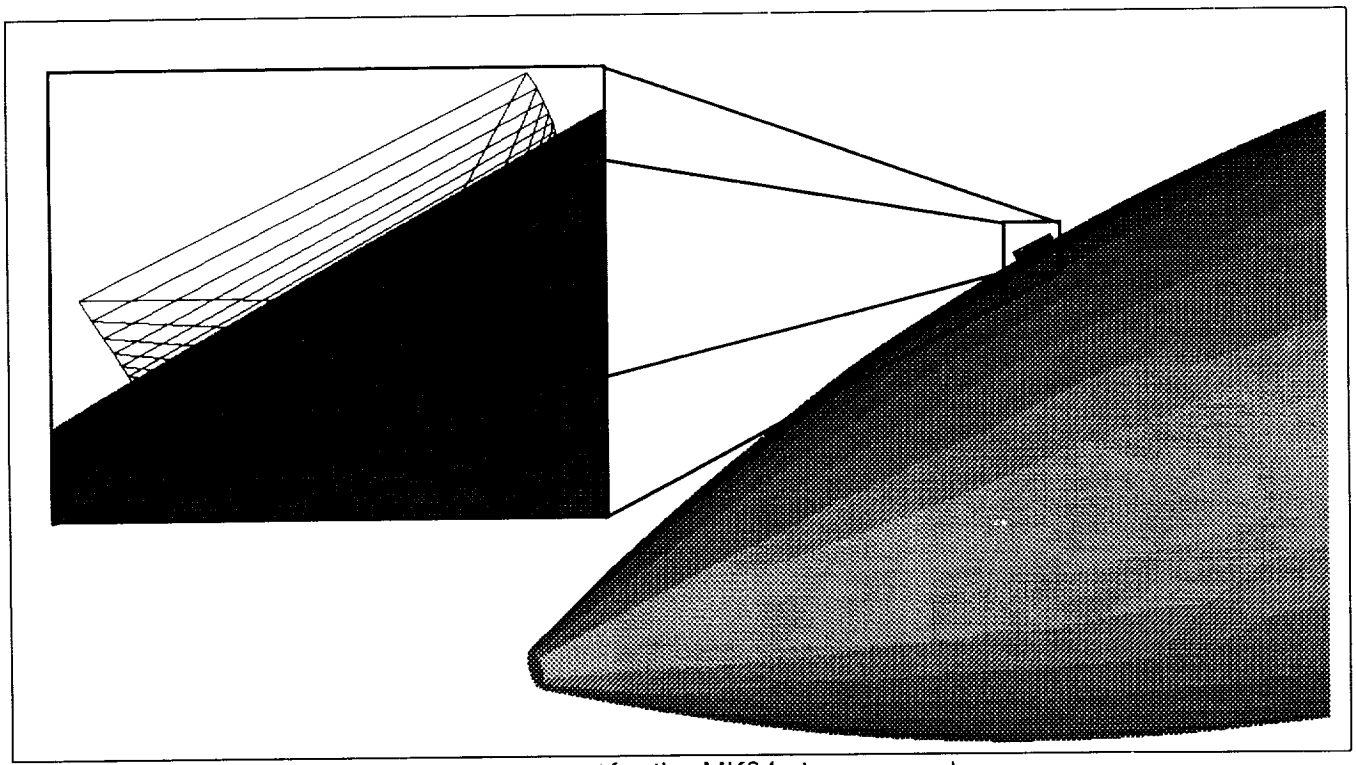

Figure 19. A prismatic grid column near nose for the MK84 store geometry.

UNSTRUCTURED GRID TECHNOLOGY