

# A $2^N$ TREE BASED AUTOMATED VISCOUS CARTESIAN GRID METHODOLOGY FOR FEATURE CAPTURING

Z.J. Wang (zjw@cfdrc.com)\*, R.F. CPhen\*, N. Hariharan\*, A.J. Przekwas<sup>†</sup>  
CFD Research Corporation, Huntsville, AL 35805

and

Darren Grove\*  
NAWC/AD, Patuxent River, MD 20670

## ABSTRACT

A  $2^n$  tree (capable of supporting binary, Quadtree and Octree) based viscous Cartesian grid generation method has been successfully developed for complex geometries. Compared with an Octree data structure, the  $2^n$  tree data structure supports anisotropic grid adaptations in any of the coordinate directions in an arbitrary manner. This capability enables high resolution of flow features such as shocks, shear layers and wakes with high aspect ratio cells. In order to properly resolve viscous boundary layers, a viscous layer grid is “inserted” between the Cartesian grid and the body surface through a projection technique. The thickness of the viscous layer grid can be determined based on the expected boundary layer thickness. Algorithms to automatically detect and resolve narrow gaps, and geometrically critical features have been developed. The method completely avoids cell-cutting, and produces overall good quality computational grids. Several demonstration cases are included to showcase the capability of the method.

## INTRODUCTION

It is generally recognized that unstructured grid-based CFD algorithms offer the best promise for automation in fluid flow simulations. The last decade has seen tremendous progress in unstructured grid methods. Types of unstructured grids include classical triangular or tetrahedral grids<sup>1-5</sup>, quadrilateral or hexahedral grids<sup>6</sup>, prismatic grids<sup>7</sup> or mixed grids<sup>8-9</sup>. The most appealing properties of unstructured grids are the geometric flexibility and the ease in which the grid can be adapted according to flow features. Tetrahedral grids are the easiest to generate. Many well-known grid

generation algorithms, such as the advancing front<sup>10</sup> and the Delauney triangulation method<sup>11</sup> have been developed to generate tetrahedral grids for complex geometries. However, experiences have indicated that tetrahedral grids are not as efficient and/or accurate as hexahedral or prismatic grids for viscous boundary layers. On the other hand, prismatic grids and hexahedral grids can resolve boundary layers more efficiently but they are more difficult to generate than tetrahedral grids. Many CFD researchers have come to the conclusion that mixed grids (or hybrid grids) are the way to go.

Recently, there has been a renewed interest in using Cartesian grids for complex geometries<sup>12-17</sup>. Coupled with a tree-based data structure and grid adaptation, with respect to both the geometry and the flow field, these methods have been demonstrated to be very viable tools for inviscid flows, with very complex geometry. The main advantages of the Cartesian grid methods are the followings: (1) automatic grid generation, (2) automatic grid adaptation and (3) simplified data structure. One of the most appealing properties of a Cartesian grid is its efficiency in filling space with a minimum number of cells and faces given a certain grid resolution. With the adaptive Cartesian grid approach, grid-independent solutions can be obtained with automated solution-based grid adaptations<sup>13</sup>. To achieve the same solutions with a non-adaptive mesh would be very expensive. One obvious drawback of the adaptive Cartesian grid method is its inability to support directional grid adaptation required in viscous boundary layer type flow problems. Isotropic grid adaptations in a boundary layer are not only too expensive but inefficient as well<sup>14</sup>. Furthermore, the irregular cut cells near the solid wall boundaries have been shown to produce severely non-positive numerical stencils for the Navier-Stokes equations and may cause convergence/stability problems. More recently, several approaches were developed to extend the Cartesian grid method to viscous flows. One was developed by Karman<sup>18</sup> and another by Wang<sup>19</sup>. Karman employed an adaptive

---

\* AIAA member

<sup>†</sup> AIAA Associate Fellow

Cartesian/fixed prism grid to tackle viscous flows. A disadvantage of Karman's approach is the use of the fixed prism grid, which partly negates the effectiveness of the adaptive Cartesian grid. To eliminate this drawback, Wang developed the adaptive Cartesian/adaptive prism grid approach for viscous flows. The computational grid is adaptive in both the Cartesian and the prism regions. In a successful 2D demonstration<sup>20</sup>, grid independent viscous solutions were achieved automatically. A common drawback of Cartesian/prism grid approaches, however, is that a complex cell-cutting procedure needs to be performed near the Cartesian/prism interface, resulting in very irregular, non-convex cut cells.

Another very promising method to extend the adaptive Cartesian grid approach to viscous flows is to "project" layers of viscous layer grids from the Cartesian grid to the geometry<sup>6, 21-23</sup>. This approach is named the **viscous Cartesian** approach in this paper. One appealing feature of the viscous Cartesian approach is that nearly all computational cells generated are convex, and that cell-cutting is avoided. In this approach, the geometry is used only for Cartesian grid "blocking", and projection. In theory, a valid surface discretization of the geometry is not necessary to generate a valid volume grid. The surface grid is automatically generated when the Cartesian grid "front" is projected to the geometry<sup>23</sup>. **As a consequence, the viscous Cartesian grid method does NOT require the surface geometry to be "water-tight"**. Therefore, it can handle "dirty" geometries with overlaps, and small cracks (small relative to the near body Cartesian grid size). This feature is potentially a considerable advantage over conventional grid generators, which require a "clean water tight" geometry to be available before grid generation can start. In this paper, the viscous Cartesian approach is further extended to 2<sup>n</sup> tree data structures to support anisotropic grid adaptations. Furthermore, a narrow gaps detection algorithm has been developed to automatically generate viscous layers grids inside narrow gaps. The method is tested on a variety of complex geometries, including the F/A 18C aircraft. Sample flow simulations are also carried out with solution based grid adaptations to demonstrate the capability of the 2<sup>n</sup> tree data structure in efficiently resolving flow features.

## **ADAPTIVE CARTESIAN GRID GENERATION**

### Geometry Entity

As explained in the introduction section, the viscous Cartesian grid method is capable of supporting arbitrary

geometric entities as long as they can be used to block Cartesian cells, and project Cartesian grid points. In terms of operations, the geometry needs to support the following three operations:

- 1) If\_Intersect\_Cartesian\_Cell(Cartesian\_Cell);
- 2) Is\_Point\_Inside(Point); and
- 3) Project\_To\_Geometry(Point);

Operation 1) return TRUE if the geometry intersects a Cartesian cell. Otherwise, it returns FALSE. Operation 2) is TRUE if a given point is inside the geometry, otherwise, FALSE. Operation 3) is the projection operator, which returns the projection of a given point to the geometry. In the current study, a triangulated surface geometric entity has been implemented, and all the operations can be defined easily.

### Data Structures and Cartesian Grid Generation

One of the popular data structures for adaptive Cartesian grid is the Octree. The drawback of Octree is that only isotropic grid refinement is supported. In this study, a 2<sup>n</sup> tree data structure has been used. The 2<sup>n</sup> tree supports binary, Quadtree and Octree type of subdivisions, and therefore allows the adaptive Cartesian grid to be refined in a non-isotropic manner. 2<sup>n</sup> tree is a hierarchical data structure in which each non-leaf tree node can have either 2, 4 or 8 child nodes. All possible Cartesian cell subdivisions supported by the 2<sup>n</sup> tree are illustrated in Figure 1. Note that a Cartesian cell can be sub-divided in an arbitrary manner. To enable reverse tree traversal, each node also keeps the address of its parent node. Therefore, two-way tree-traversal can be performed very efficiently.

Usually an adaptive Cartesian grid is generated by recursively subdividing a single root cell. In this study, the adaptive Cartesian grid is started from a forest of root cells in order to add more flexibility in controlling the size of the computational domain and aspect ratios of Cartesian cells. Since the root forest must cover the entire computational domain, the surface geometry is contained in the root Cartesian cells. The size of the Cartesian cells intersecting the geometry is controlled by two parameters, disT and disN. Parameter disN controls the Cartesian cell size in the geometry normal direction, whereas disT specifies the Cartesian cell size in the geometry tangential direction. The ratio disT/disN determines the maximum aspect ratio in the Cartesian grid. The recursive sub-division process stops when all the Cartesian cells intersecting the geometries satisfy the length scale requirements. For the sake of solution accuracy, it is very important to ensure that the

Cartesian grid is smooth. In the present study, the sizes of any two neighboring cells in any coordinate direction cannot differ by a factor exceeding 2. The use of the  $2^n$  tree data structure makes high aspect ratio Cartesian cells possible. This property can translate into considerable efficiency gains when anisotropic grid adaptations are used to resolve flow features. As an example, an Octree grid is compared with a  $2^n$  tree grid generated for the F-16 aircraft. Both grids used the same parameter  $disN$ , while the  $2^n$  tree used a  $disT$  5 times the size of  $disN$ , which means the  $2^n$  tree Cartesian grid can generate Cartesian cells with a maximum aspect ratio of 5. The Octree Cartesian grid consists of 559,938 cells, whereas the  $2^n$  tree Cartesian grid has only 87,046 cells. The saving in the number of cells with the  $2^n$  tree compared to Octree is over a factor of 6. The side view of the Octree and  $2^n$  tree Cartesian grids are shown in Figure 2. Note that at places where the geometry surface is approximately aligned with any of the coordinate directions, non-isotropic, high aspect ratio cells are automatically generated with the  $2^n$  tree. Even if the geometry surface is not exactly aligned with a coordinate direction, non-isotropic cells can still be used to reduce the total number of cells, as shown in this figure.

#### Cartesian Grid Front Generation and Smoothing

It is easy to notice that if there are narrow gaps in the geometry smaller than the near body Cartesian grid cell size, the narrow gaps may be completely invisible to the grid generator, and may be meshed over. In order to resolve geometrically important narrow gaps, small Cartesian cells must be generated inside the narrow gaps. Therefore, narrow gaps in the geometry must be detected first. An efficient ray-tracing algorithm has been developed to detect narrow gaps in the geometry.

In order to “insert” a viscous layer grid between the Cartesian grid and the body surfaces, Cartesian cells intersected by the geometry must be removed. Cartesian grid cells which are intersected by the body surface are called Cut Cells (CC), and cells which are located inside the body surfaces are named Hole Cells (HC) and the rest of the cells are called Normal Cells (NC). All the CC and HC are “blocked” (turned off) from the Cartesian grid, leaving an empty space between the Cartesian grid and the body surface. Both the  $2^n$  tree (for the Cartesian grid) and the ADT data structures (for the surface triangles) are extensively utilized for efficient search operations.

After this step, all the exposed Cartesian faces are gathered together and form the so-called Cartesian grid front. The exposed Cartesian faces are then smoothed using a Laplacian smoother to form a smoother front, which is to be projected to the body surface. An example Cartesian front generated for a missile geometry is shown in Figure 3. The front before and after smoothing is also displayed in Figure 3.

### VISCOUS LAYER GRID GENERATION

#### Projection of the Cartesian Front to the Body Surface

After the smoothed front in the Cartesian grid is obtained, each node in the front needs to be connected to the body surface to form the viscous layer grids. The “foot prints” of the layer grids on the body surface have the same topology (or connectivity) as the Cartesian front. With this assumption, the viscous layer grids are naturally “blended” with the adaptive Cartesian grid, eliminating the need of cell-cutting currently adopted by many Cartesian grid generators. Another major advantage of the approach over cell cutting is that nearly all computational cells generated are convex, boosting the stability and convergence properties of flow solvers. The projection from the Cartesian grid front to the body surface is performed according to the minimum distance rule. To efficiently identify the triangles which are close to a given point, an Alternating Digital Tree (ADT) data structure<sup>24</sup> is used to store the bounding boxes of these triangular faces. The ADT data structure is a binary tree data structure, in which each tree node has two child nodes. The search complexity using ADT tree is close to  $\log_2(M)$  rather than  $M$  ( $M$  being the total number of triangles).

#### Geometric Feature Preservation

The front projection based on the minimum distance criterion may smear critical geometric features, especially near non-convex corners. This is shown in Figure 4. The geometrically “critical” concave corner never connects with a front node, and is therefore not represented in the projected surface grid. In order to eliminate this problem, a geometric feature recovery algorithm was developed. The algorithm first detects all the critical features in the geometry automatically. Then all the critical features are preserved through a feature recovery technique, in which front nodes are reconnected to the critical features.

**Critical Feature Detection:** Each edge of a surface triangulation is examined to see whether it is a “critical edge”. If the angle between the normals of the two triangles sharing an edge is larger than a threshold (e.g.

30 degrees), the edge is classified as a “critical edge”. All Critical edges are what must be preserved in Cartesian front projections.

**Critical Feature Preservation:** In order to preserve the geometric features, Cartesian front nodes must be connected to these features directly. Since the topology of the projected body grids is the same as that of the Cartesian front, any critical edge must be matched to a group of connected edges on the Cartesian front. The projected surface grids of the missile geometry with and without feature recovery are shown in Figure 5a and 5b. Note that the critical intersection curves between the fins and the missile body are properly resolved with feature preservation.

Surface Grid Smoothing and Layer Grid Generation

The projected surface grid from the Cartesian grid front is usually not smooth because the projections are based on the minimum distance to the triangulated surfaces. The surface grid is also dependent on the local surface curvatures, and the initial triangulation. After the feature recovery step is carried out, the surface grid becomes even more unsmooth. A Laplacian smoothing algorithm is therefore applied to improve the grid quality.

By connecting each point on the Cartesian grid front to the corresponding projected point on the surface, only a single layer of viscous grids is produced. To perform a meaningful viscous flow simulation, many more layers of viscous grids are necessary. Therefore this single layer of viscous grids is further divided into (user specified) arbitrary number of layers with arbitrary grid point distributions.

**FLOW SOLVER AND SOLUTION BASED GRID ADAPTATION**

A cell-centered finite volume flow solver supporting arbitrary cells was used to perform flow simulation<sup>20</sup>. The flow solver is second-order accurate with a cell-wise least-squares linear reconstruction technique. Roe’s approximate Riemann solver<sup>25</sup> was used to compute the inviscid flux. The viscous flux was computed with an efficient and compact approach described in Reference 20, which is also linearity-preserving. An improved LU-SGS scheme with fast convergence characteristics<sup>26</sup> was employed for time-integration.

To achieve automation in flow simulation, solution-based grid adaptation is essential. To take full advantage

of the anisotropic grid adaptation capability offered by the 2<sup>n</sup> tree, the three coordinate directions of each Cartesian cell are examined independently for possible adaptation. Since the viscous layer grid is generated by projecting the Cartesian front to the geometry, it cannot be independently adapted. However, the number of viscous layers, and the grid clustering factor can be changed based on local flow features. Both first and second derivative based grid adaptation criteria have been developed. The following cell-wise parameters are used as the adaptation indicators in x-direction

$$\tau_{ix} = \left| \frac{\partial q_i}{\partial x} \right| \Delta x_i^{1.5} \tag{1}$$

$$\pi_{ix} = \left| \frac{\partial^2 q_i}{\partial x^2} \right| \Delta x_i^3 \tag{2}$$

where q is a flow variable,  $\tau_{ix}$  is a first-derivative based indicator, while  $\pi_{ix}$  is a second-derivative based indicator. The adaptation indicators in other directions can be computed similarly. The standard deviation of the parameter is computed as:

$$\tau = \left( \frac{\sum_{i=1}^N \tau_{ix}^2 + \sum_{i=1}^N \tau_{iy}^2 + \sum_{i=1}^N \tau_{iz}^2}{3N} \right)^{\frac{1}{2}} \tag{3}$$

where N is the total number of Cartesian cells. The following conditions are used for grid adaptation:

- 1) if  $\tau_{ix} > c*\tau$ , cell i is to be refined in x direction;
- 2) if  $\tau_{iy} > c*\tau$ , cell i is to be refined in y direction;
- 3) if  $\tau_{iz} > c*\tau$ , cell i is to be refined in z direction;

where c is a control parameter determining the total number of cells to be refined. In this study, c is chosen to be 1.

**TEST CASES**

Transonic Flow over ONERA M6 Wing

This first test case is the inviscid transonic flow over an ONERA M6 wing configuration. The M6 wing has a leading-edge sweep angle of 30 degrees, an aspect of 3.8, and a taper ratio of 0.562. The airfoil section of the wing is the ONERA “D” airfoil, which is a 10% maximum thickness-to-chord ratio conventional section. The flow was computed at a Mach number of 0.84, and an angle of attack of 3.06. To resolve the flow field near

the wing with reasonable accuracy, a box source with corner coordinates  $(-0.25, -0.25, 0)$  and  $(1.25, 0.25, 1.25)$  was added to control Cartesian grid distributions. Cartesian grid cells inside the box are required to satisfy  $\Delta x = 0.02$ ,  $\Delta y = 0.02$ ,  $\Delta z = 0.04$ . The far field was placed 20 chords away from the wing. The initial viscous Cartesian mesh is shown in Figure 6, which consists of 83,997 cells, and 262,129 faces. The grid has two viscous layers between the Cartesian grid and the body. The computed pressure contours on the wing surface and the plane of symmetry are displayed in Figure 7. It is clear that the leading edge of the wing does not have enough grid resolution, and the lambda-type shock structure is quite heavily smeared. Three levels of solution-based grid adaptations were then performed after the solutions fully converged on the coarser meshes. The adaptation criteria were pressure and total velocity gradients. The level 3 viscous Cartesian grid and the computed pressure contours are shown in Figure 8. The Level 3 grid has 395,117 cells and 1,278,336 faces. Note that the computational mesh was not only adapted near the shocks, and shear layers after the trailing edge, but also adapted in smooth flow regions, which is important to achieve high global solution accuracy. The lambda-shock structure was sharply captured on the level 3 grid. The computed pressure coefficient distributions are also compared with experimental wind tunnel data<sup>27</sup> at six spanwise stations in Figure 9. We can observe that the solution on the level 2 grid is nearly identical to that on the level 3 grid, indicating that the solution on the level 2 grid is nearly grid-independent. Generally speaking, the computed pressure coefficient profiles agree quite well with experimental data except at the root station where viscous effects are important.

#### Transonic Flow Over a Wing/Pylon/Store Configuration

The second test case is performed for a wing/pylon/store configuration reported in Reference 28. The configuration consists of a clipped delta wing with a 45 degree sweep composed of a constant NACA64010 symmetric airfoil section. The wing has a root chord of 16 inches, a semispan of 13 inches, and a taper ratio of 0.134. The pylon is located at the midspan station and has a cross section characterized by a flat plate closed at the leading and trailing edges by a symmetrical ogive shape. The width of the pylon is 0.294 inches. The four fins on the store are defined by a constant NACA0008 airfoil section with a leading-edge sweep of 45 degrees, and a truncated tip. The store is in the so-called carriage position. The narrow gap detection algorithm was applied to automatically detect the gap between the pylon and the store, and fine grid cells were then

generated within the gap automatically. A cross section grid through the narrow gap is displayed in Figure 10, indicating that the gap is sufficiently resolved. The initial viscous Cartesian grid and the level 3 solution-adaptive grids are shown in Figure 11. Pressure gradients were used as the adaptation criterion. The initial grid consists of 128,659 cells, and 435,195 faces, and the level 3 grid has 268,137 cells and 900,929 faces. The flow solutions are presented at a Mach number of 0.95, and an angle of attack of 0. The computed pressure contours on the wing lower surface and store for the initial and the level 3 grids are displayed in Figure 12. The shock waves were sharply captured on the level 3 grid. The computed pressure coefficients on the store are also compared with experimental data at two circumferential stations in Figure 13. The agreement with experimental is quite good considering that no viscous effects were included.

#### Viscous Cartesian Grid for Complete F/A-18 Aircraft

The final test geometry is the F/A 18 fighter aircraft including loaded missiles. The original input geometry was in PLOT3D patches. CFDR's geometric modeler and grid generator CFD-GEOM was used to clean the geometry, and produce a surface triangulation with 60,263 faces. The feature detection algorithm was employed to automatically track all the critical edges, and the narrow gap detection algorithm was employed to identify small gaps in the geometry. The surface grid generated is shown in Figure 14, which consists of 41,890 polygons. The grid has a total of 375,285 cells, 1,182,100 faces, and it has eight viscous layers with an exponential clustering factor of 1.3. This grid took about half an hour to generate on a SGI O2 (R10000) workstation. The viscous Cartesian grid on the plane of symmetry is displayed in Figure 15a, and a cutting plane of the grid near the main inlet is shown in Figure 15b. Note that small cells were generated inside the small gap near the inlet. Due to the coarseness of the initial surface triangulation, some geometric features are still smeared. Improvements are being made to produce a better grid for a flow simulation.

### CONCLUSIONS

The 2<sup>n</sup> tree adaptive viscous Cartesian grid methodology developed in this study is capable of automatically generating viscous computational grids for complex geometries. The 2<sup>n</sup> tree data structure allows the Cartesian grid to be adapted in an arbitrary manner to capture features efficiently. In addition, the use of state-of-the-art data structures, such as the 2<sup>n</sup> tree and ADT data structures, considerably reduces costs

associated with various search operations in the method, resulting in a very efficient overall methodology. Furthermore nearly all grid cells are convex. A geometric feature detection and preservation algorithm, and a narrow gap detection algorithm were shown to be critical to the success of the method. Another unique feature of the viscous Cartesian grid method is that valid computational grids can be generated for “dirty geometries” with overlaps or cracks. Solution based grid adaptations were shown to be very effective in capturing key flow features. Extensive demonstrations with viscous turbulent flows will be carried out in the near future.

### ACKNOWLEDGMENT

The authors gratefully acknowledge valuable discussions with many colleagues in CFDRC, in particular, William Coirier, Sami Habchi, and Sami Bayyuk. The technical guidance and advice of Ashok Singhal of CFDRC were very valuable. Thanks are also due to H. Luo of SAIC for providing the experimental data of the M6 wing case, and a surface triangulation of the M6 wing, to Stacy Rock of CFDRC for providing the wing-pylon-store geometry, to Les Hall for providing the experimental data for the wing-pylon-store case and for cleaning the F/A 18 geometry, and to John Whitmire for generating the F/A 18 surface triangulation. We thank Richard Sun and Kumar Srinivasan of DaimlerChrysler for testing the viscous Cartesian grid generator. The study has been funded under a Navy SBIR project from NAWC/AD, the U.S. Navy.

### REFERENCES

- Barth, T.J., and Frederickson, P.O., “Higher-Order Solution of the Euler Equations on Unstructured Grids Using Quadratic Reconstruction,” AIAA Paper 90-0013, 1990.
- Batina, J.T., “Unsteady Euler Algorithm with Unstructured Dynamic Mesh for Complex Aircraft Aerodynamic Analysis,” AIAA Journal, vol. 29, pp. 327-333, 1991.
- Venkatakrishnan, V., “Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters,” J. Comput. Phys., vol. 118, 1995, pp. 120-130.
- Anderson, W.K. and Bonhaus, D.L., “An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids,” Computers Fluids, 23 (1994), pp. 1-21.
- Connell, S.D. and Holmes, D.G., “A 3D Unstructured Adaptive Multigrid Scheme for the Euler Equations,” AIAA paper 93-3339-CP, 1993.
- Schneiders, R., “Automatic Generation of Hexahedral Finite Element Meshes,” in Proceedings of 4th International Meshing Roundtable '95, October 1995, Albuquerque, NM.
- Nakahashi, K., “Adaptive-Prismatic-Grid Method for External Viscous Flow Computations,” in Proceedings of 11th AIAA Computational Fluid Dynamics Conference, July 1993, Orlando, FL. AIAA-93-3314-CP.
- Kallinderis, Y., Khawaja, A. and McMorris, H., “Hybrid Prismatic/Tetrahedral Grid Generation for Complex Geometries,” AIAA Paper No. 95-0210, 1995.
- Coirier, W.J. and Jorgenson, P.C.E., “A Mixed Volume Grid Approach for the Euler and Navier-Stokes Equations,” AIAA Paper No. 96-0762, 1996.
- Lohner, R. and Parikh, P., “Generation of Three-Dimensional Unstructured Grids by Advancing Front Method,” International J. Numerical Method in Fluids, vol. 8, pp. 1135-1144, 1988.
- Baker, T.J., “Three-Dimensional Mesh Generation by Triangulation of Arbitrary PointSets,” AIAA Paper No. 87-1124, 1987.
- Berger, M.J. and LeVeque, R.J., “An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries,” AIAA-89-1930-CP, June 1989.
- DeZeeuw, D. and Powell, K., “An Adaptively Refined Cartesian Mesh Solver for the Euler Equations,” AIAA-91-1542-CP, 1991.
- Coirier, W.J. and Powell, K.G., “Solution-Adaptive Cartesian Cell Approach for Viscous and Inviscid Flows,” AIAA J., vol. 34, no.5, pp. 938-945, 1996.
- Bayyuk, A.A., Powell, K.G., and van Leer, B., “A Simulation Technique for 2D Unsteady Inviscid Flows Around Arbitrarily Moving and Deforming Bodies of Arbitrarily Geometry,” AIAA Paper 93-3391-CP, 1993
- Melton, J.E., Enomoto, F.Y., and Berger, M.J., “3D Automatic Cartesian Grid Generation for Euler Flows,” AIAA Paper 93-3386-CP.
- Aftosmis, M.J., Berger, M.J. and Melton, J.E., “Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry,” AIAA Paper No. 97-0196, 1997.
- Karman, S.L., “SPLITFLOW: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complete Geometries,” AIAA-95-0343, 1995.
- Wang, Z.J., “Proof of Concept – An Automatic CFD Computing Environment with a Cartesian/Prism Grid Generator, Grid Adaptor and Flow

- Solver,” in Proceedings of 4th International Meshing Roundtable, pp 87–99, October 1995, Albuquerque, New Mexico.
20. Wang, Z.J., “A Quadtree-Based Adaptive Cartesian/Quad Grid Flow Solver for Navier-Stokes Equations,” *Computers & Fluids*, vol. 27, No. 4, pp. 529-549, 1998.
  21. Smith, R.J., and Leschziner, M.A., “A Novel Approach To Engineering Computations for Complex Aerodynamic Flows,” Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations, Mississippi State University, MS, 1996.
  22. Tchon, K.F., Hirsch, C. and Schneiders, R., “Octree-Based Hexahedral Mesh Generation for Viscous Flow Simulations,” AIAA-97-1980-CP, 1997.
  23. Wang, Z.J., “An Automated Viscous Adaptive Cartesian Grid Generation Method for Complex Geometries,” in Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations, University of Greenwich, London, England, 1998.
  24. Bonet, J.A. and Peraire, “An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems,” *International J. Numerical Methods in Engineering*, vol. 31, pp. 1-17, 1991.
  25. Roe, P.L., “Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes,” *Journal of Computational Physics*, vol. 43, pp. 357, 1983.
  26. Chen, R.F. and Wang, Z.J., “An Improved LU-SGS Scheme with Faster Convergence for Unstructured Grids of Arbitrary Topology,” AIAA-99-0935, 1999.
  27. Schmitt, V. and Charpin, F., “Pressure Distributions on the ONERA M6-wing at Transonic Mach Numbers,” Experiment Data Base for Computer Program Assessment, AGARD AR-138, 1979.
  28. Ileim, E.R., “CFD Wing/Pylon/Finned Store Mutual Interference Wind Tunnel Experiment,” AEDC-TSR-91-P4, Arnold Engineering Development Center, Arnold AFB, TN, Jan. 1991.

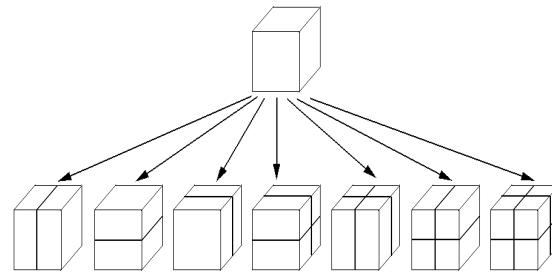
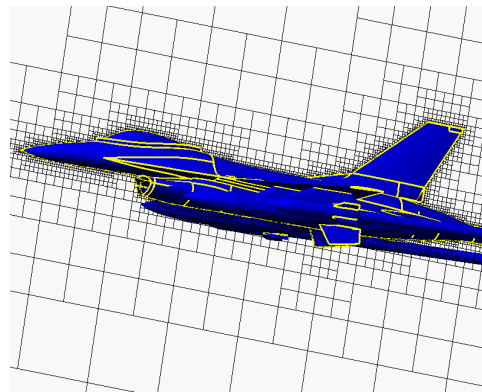
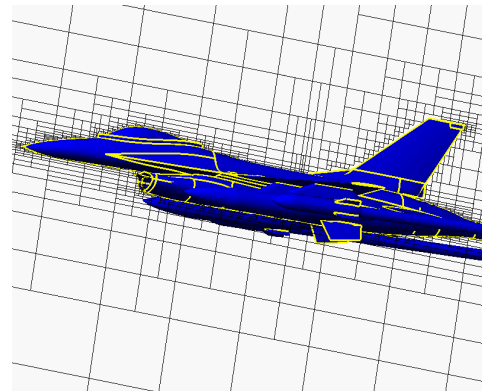


Figure 1. Cartesian Cell Subdivisions Supported by  $2^n$  Tree Data Structure



(a) Octree



(b)  $2^n$  tree

Figure 2. Side Views of Cartesian Grids for the F-16 Generated Using Octree and  $2^n$  Tree

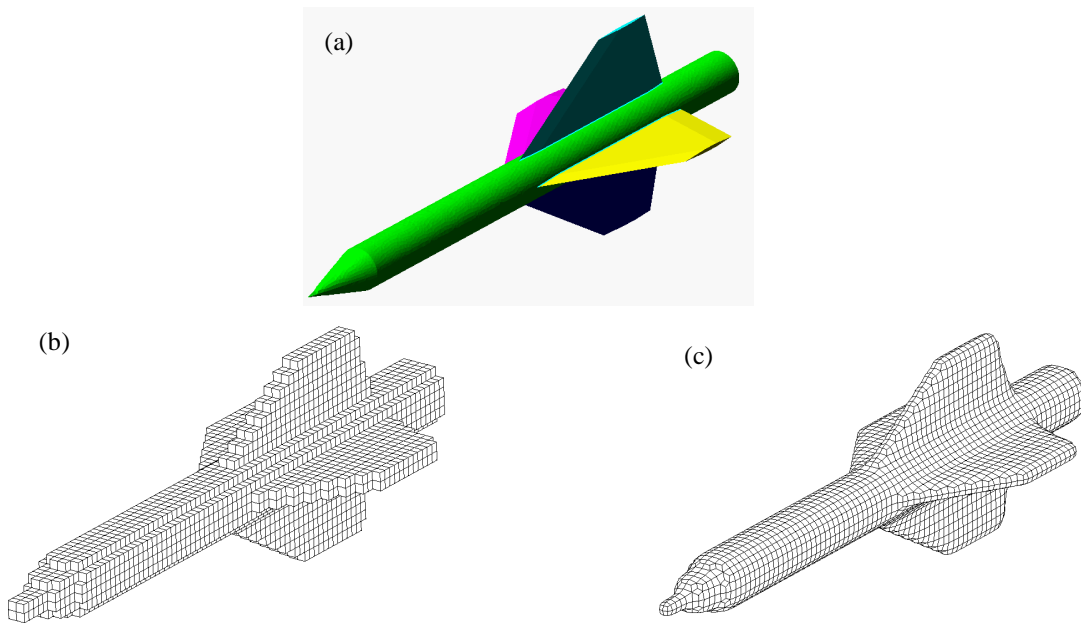


Figure 3. (a) A Missile Geometry, (b) the Non-Smooth Cartesian Grid Front, and (c) the Smoothed Cartesian Grid Front

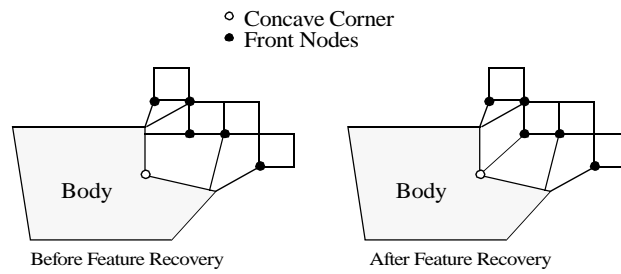


Figure 4. Illustration of a Smeared Concave Corner in Front Projection

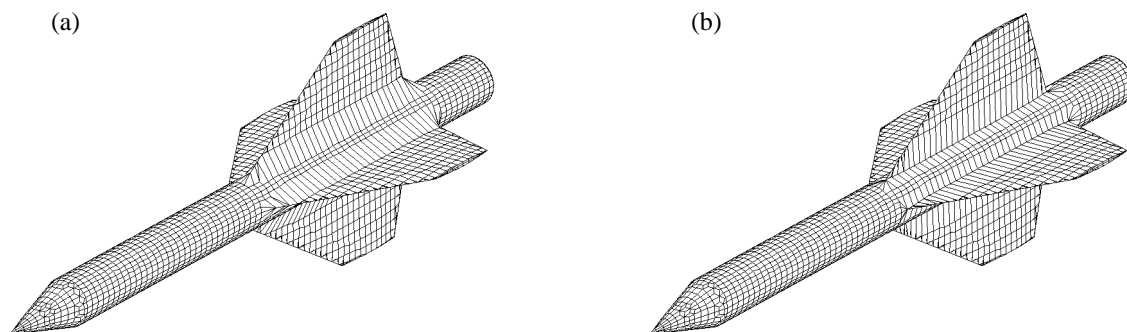


Figure 5. Comparison of Projected Surface Grids with (a) and without (b) Geometric Feature Recovery

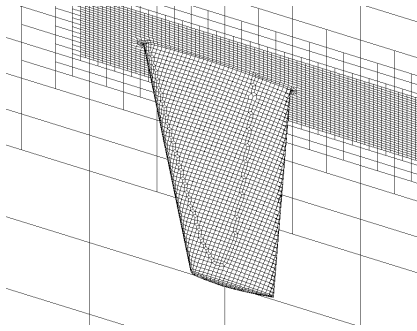


Figure 6. Initial Viscous Cartesian Grid for the ONERA M6 Wing (131,657 cells, 417,551 faces)

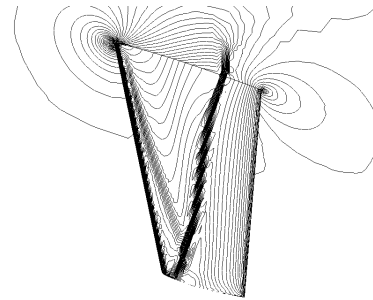


Figure 7. Computed Pressure Contours on the Upper Surface at Mach = 0.84, and  $\alpha = 3.06^\circ$

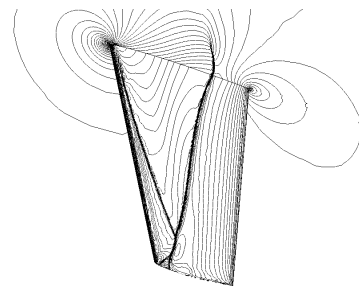
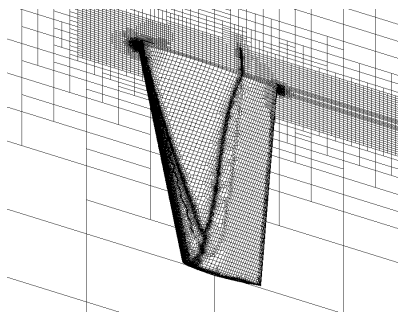


Figure 8. Level 3 Solution-Adaptive Viscous Cartesian Grids and Computed Pressure Contours for ONERA Wing

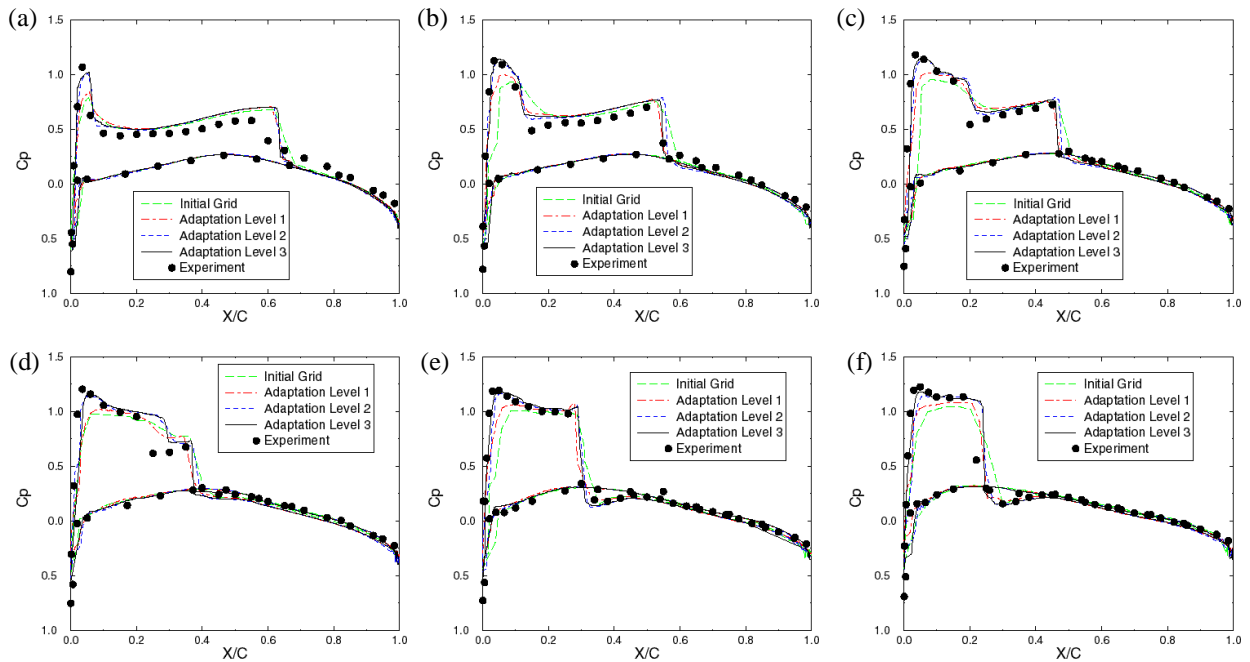


Figure 9. Comparison between Computed and Experimental Surface Pressure Coefficient at Different Spanwise Stations (a) 20% Semispan, (b) 44% Semispan, (c) 65% Semispan, (d) 80% Semispan, (e) 90% Semispan, and (f) 95% Semispan

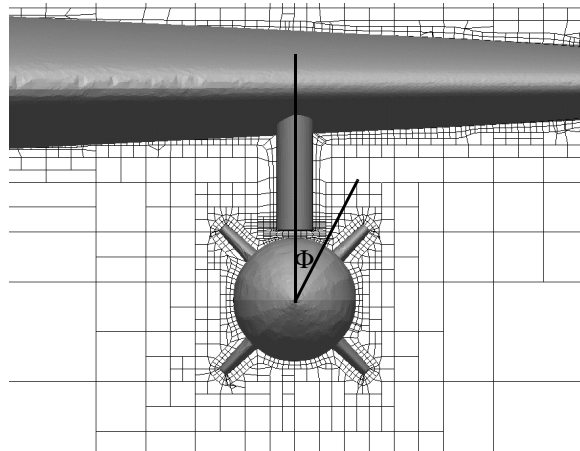


Figure 10. A Cross-Section Cut in the Viscous Cartesian Grid near the Gap Between Pylon and Store

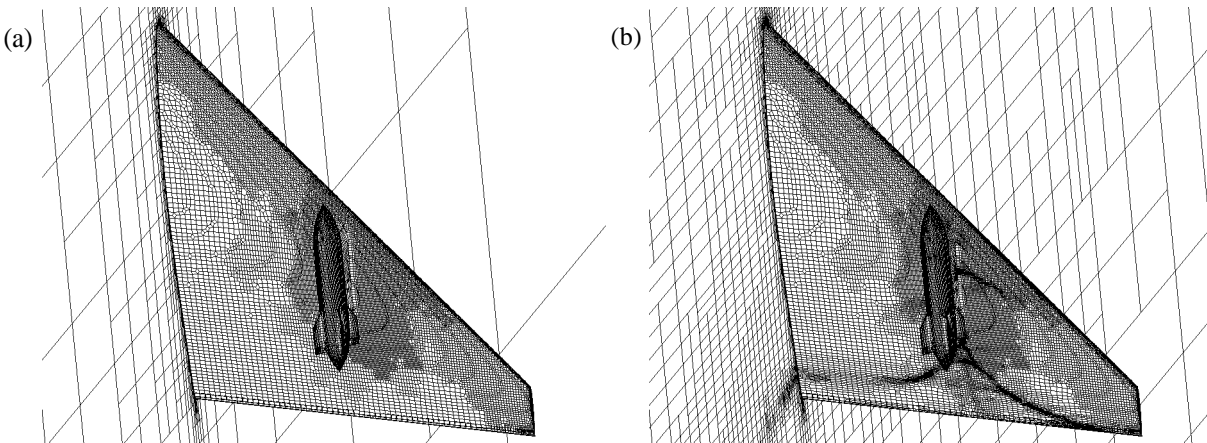


Figure 11. Initial and Level 3 Solution Adaptive Viscous Cartesian Grids on the Lower Wing Surface and the Plane of Symmetry

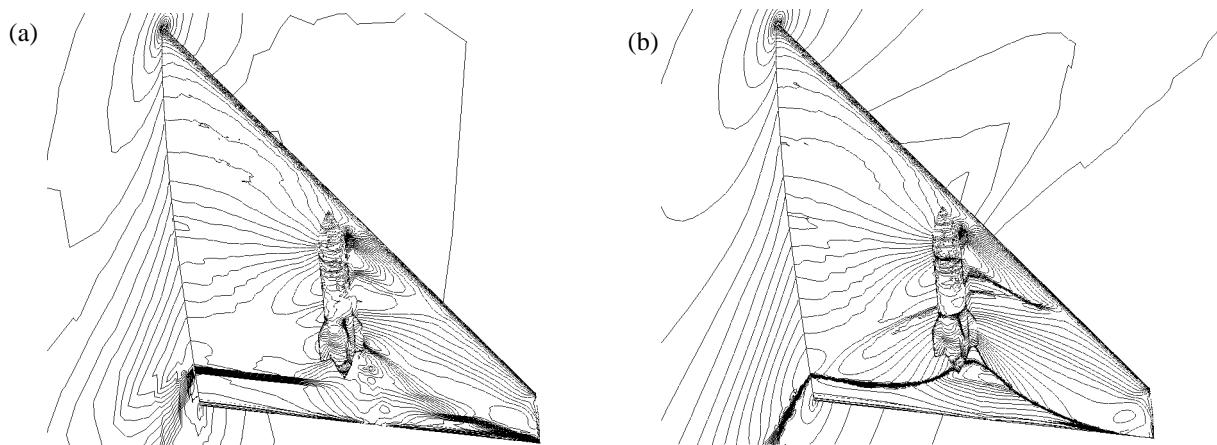


Figure 12. Computed Pressure Contours on the Lower Wing Surface at Mach = 0.9,  $\alpha = 0$

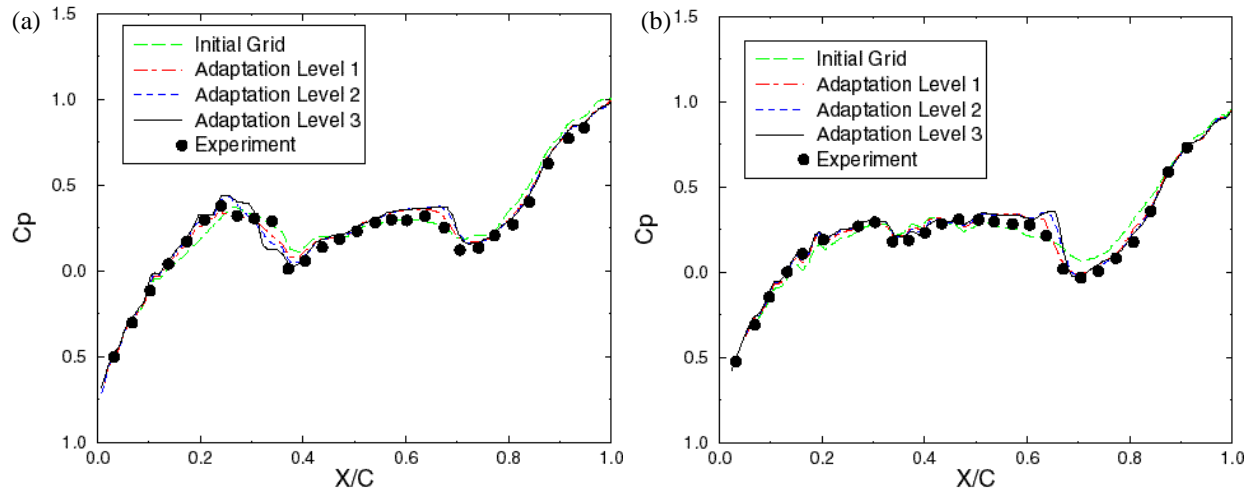


Figure 13. Comparison of Pressure Coefficient on the Store at Stations of (a)  $\Phi = 185$ , and (b)  $\Phi = 95$  Degrees, Refer to Figure 10 for the Definition of  $\Phi$

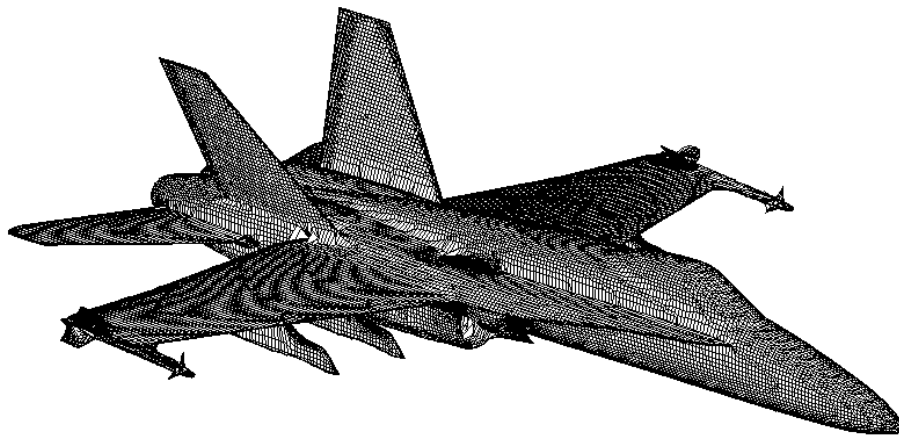


Figure 14. The Surface Grid Generated on the F/A 18 Fighter Aircraft with the Viscous Cartesian Grid Method

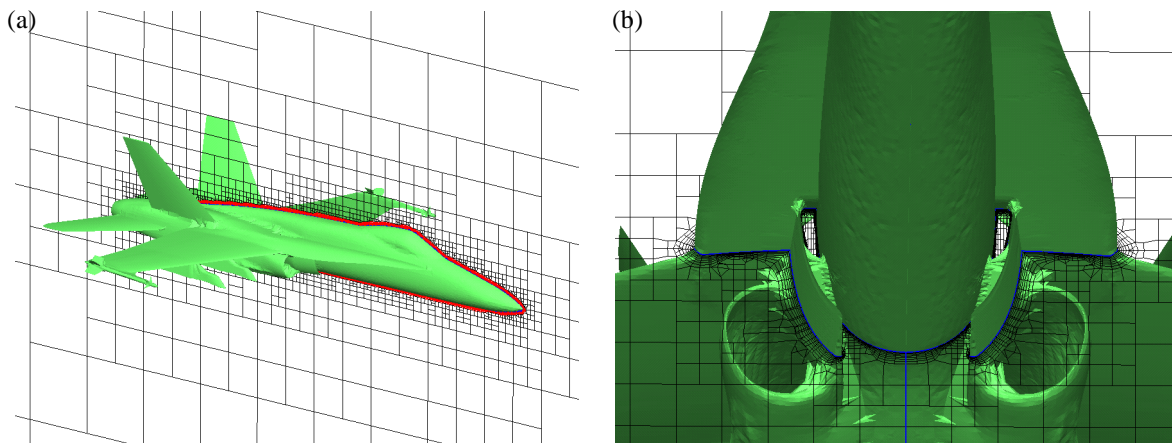


Figure 15. The Viscous Cartesian Grid on the Plane of Symmetry, and a Cross-Section Grid near the Main Inlets