Flow Simulation Method based on Hexahedra Grid

by
Paulus R. Lahur
Institute of Space Technology and Aeronautics, Japan Aerospace Exploration Agency

ABSTRACT

Research in automatic grid generation around realistic 3D geometry is being conducted in Japan Aerospace Exploration Agency. This research aims at reducing the time required to generate computational grid for inviscid as well as viscous flow simulations. The method is based on hexahedra grid method, as the extension of Cartesian grid method. As the result, the method inherits significant advantages of Cartesian grid: it is automatic and fast. Because an efficient viscous flow simulation requires grid with high aspect ratio near solid surface, a hybrid with prismatic grid is necessary. The hexahedra grid extends naturally to this approach, because there is consistency in grid topology (both Cartesian and prismatic grids have the same hexahedral shape). The main idea of the method is as follows: construct Cartesian grid, remove the cells near solid surface, and construct hexahedral cells based on the Cartesian cells' faces toward the surface. Note that both grids are not constructed from the solid surface. This leads to another advantage: the method is not sensitive to defects in solid surface (e.g. crack, overlap), because it actually reconstruct the solid surface grid. However, because of the way the hexahedral cells are constructed (simple projection from Cartesian grid cells' faces toward solid surface), there is a disadvantage: it can not capture sharp concave features. A method to overcome this problem is devised in the present research. This paper provides the summary of the method being developed as well as some current results.

1. Introduction

This research is motivated by the fact that grid generation for a complex, 3D geometry is still a major bottleneck in CFD, especially for the case of structured grid, where significant amount of time and manual labor is required. Our objective is to automatically generate grid suitable for inviscid and viscous flow computation within a short time.

A number of grid generation methods were considered. Some methods are body-fitted, i.e. they are generated from the solid surface outward, such as structured grid,[1] prismatic grid,[2] and tetrahedron-based unstructured grid.[3] The other is non-body-fitted approach, usually known as Cartesian grid.[4,5] For automatic grid generation, usually tetrahedron-based unstructured grid or Cartesian grid is employed.

In computation of viscous flow, especially at high Reynolds number, we have a strict requirement for the grid near solid boundary. This flow region is best handled by prismatic grid. Consequently, it is common to take the hybrid approach: one grid to resolve boundary layer, and either tetrahedron-based unstructured grid[6,7] or Cartesian grid[8-13] to resolve the rest of the domain.

A hybrid between Cartesian grid and prismatic grid is chosen in this study. There are two options in constructing the interface between the two grids: cutting the Cartesian grid cells to fit the prismatic grid (which is constructed from solid surface),[8-10] or constructing the prismatic grid from Cartesian grid toward the solid surface.[11-13] The second approach is chosen because it produces only hexahedral cells. In contrast, the first approach results in cutcells of polyhedral shape, with less regular size and shape distribution, which is not desirable in viscous computation (Fig. 1). This also simplifies the grid generation algorithm, because the uniformity in grid topology allows one to worry about geometric issue only (in the cutcell approach we have to compute both topology and geometry). Another significant advantage is that the approach tolerates a certain degree of defects in solid surface definition, such as overlaps or small gaps, which can reduce the effort necessary to clean up the geometry obtained from CAD. In fact, the method reconstructs the solid surface grid. The grid generation method will be discussed in more detailed in the following.

2. Grid Generation

The main procedure of the grid generation is as follows:

1) Generate Cartesian grid
2) Remove Cartesian cells near solid surface
3) Construct surface grid
4) Construct hexahedra layer
5) Recover features

The solid surface taken as input by the present implementation of this method is assumed to be in its discretized form. The output grid is in an unstructured format, suitable for a flow solver developed in-house, UPACS-UGS (Unified Platform for Aerospace Computational Simulation – Unstructured Grid Solver).

2.1. Cartesian Grid Generation

Generating Cartesian grid is a simple task, since the grid follows the Cartesian coordinate system (i.e. non-body-fitted grid). To increase local resolution, a cell is divided into eight sub-cells. This type of local refinement can be handled easily by octree data structure. Each cell points at its parent and children. A neighboring cell can be found by traversing the tree to find common ancestor.

The grid generation starts with a single Cartesian grid cell covering the whole computational domain. The cell is then refined. The children cells that intersect the body surface are subsequently refined until a certain grid resolution is achieved. To ensure smooth transition between cell sizes, the neighboring cells within a certain distance are also refined. Further refinement takes place around curved body surface, which is determined by evaluating the angle between surface elements intersecting a Cartesian grid cell. If the angle between the normal vectors of any two faces i, j is greater than a specified value, then the cell needs to be refined. In terms of dot product, this can be expressed as follows.

$$\min\left(\hat{n}_i \bullet \hat{n}_j\right) < \cos\left(\theta_{\max}\right), \quad \forall i, j, \quad i \neq j \tag{1}$$

Another tree data structure, Alternating Digital Tree (ADT),[14] is also used to speed up the process in finding the faces of solid surface grid possibly intersected by a given Cartesian grid cell.

2.2. Cartesian Grid Cell Removal

For computational grid, only the Cartesian grid cells that do not have children are used. Thus, all parent cells are removed. Additionally, the following Cartesian grid cells are also removed:

1) all cells inside the solid
2) all cells intersected by the solid surface
3) some cells around solid surface, to allow for some space in hexahedra layer construction

The third point needs further consideration, because simply removing cells within a certain distance from the solid surface may cause problem at later stage, particularly during the

construction of surface grid. A "proper" surface may not touch, or worse, intersect itself. In other word, at any given point on the surface, the neighborhood of the point is topologically like a disk. Thus there are certain cell configurations that are not allowed, as shown in Fig. 6. For the configuration at the bottom left, for example, a point at the shared edge does not have a disk topology. In the current implementation, check for illegal cell configuration is performed, and when found, removed. The logic to perform this task can be rather tricky, because the cells may belong to different level of refinement. The sample result of this stage is shown in Fig. 2.

2.3. Surface Grid Construction

The outer faces of the remaining Cartesian grid cells now form the boundary surface of computational domain. The portion close to the original solid surface forms a surface grid. As discussed previously, this surface has a proper topology. It is also watertight and its elements have regular size and shape. Geometrically, the surface is an approximation of the original solid surface, albeit a poor one, due to its stair-like appearance. Nevertheless this surface forms the basis for the next steps, which will successively improve the level of approximation.

Winged-edge data structure[15] is used to represent the surface. This structure is widely used in surface modeling. The structure is known for its efficiency in adjacency search (a search of any item adjacent to any item, where an item can be a node, a face, or an edge).

The surface grid is then smoothed to make the subsequent step easier. To obtain the new position (x) of a vertex, a weighted average of the surrounding faces' center coordinates are computed, as in Eq. (2).

$$x_v^{new} = (1-w)x_v + w\frac{\sum d_{vf}^{p} x_f}{\sum d_{vf}^{p}} \qquad (2)$$

where the subscript v indicates a vertex, f indicates face surrounding the vertex, and d_{vf} is the distance between the vertex and the face center. The weighting factor w is set to 0.5, and the power p is usually taken as unity. The smoothing is carried out 5 times.

To further enhance the quality of the grid, after each smoothing step, the hanging node is repositioned so that it is exactly in the middle between its two opposing neighbor nodes. The sample result of the smoothing is shown in Fig. 3.

2.4. Hexahedra Layer Construction

A layer of hexahedra grid cells is generated on the surface grid. Initially, the new surface of the layer coincides with the old one (thus the layer has zero thickness). Then each node of the new surface is "projected" onto the solid surface. The projection is carried out simply by moving a node to the closest location on the solid surface.

The search for solid surface elements that are candidates for projection can be performed at reasonable speed because the elements are already stored in ADT. The actual distance computation is performed only for these candidates. The sample result is shown in Fig. 4.

Having done this, prismatic layers can be constructed by dividing the layer as appropriate, as shown in Fig. 5. In some cases, however, the thickness of the layers is not sufficient to capture the whole boundary layer of the flow. To increase the thickness, it is necessary to deform the surrounding Cartesian grid cells outward, or to allocate more space during the Cartesian grid removal step. This will be considered in future work.

Note that the simple projection works even if the solid surface contains defects such as overlap, small gap, or even internal elements, because it always ignores the elements that are located farther away than other elements, as in Fig. 7. However, this is also the source of the method's biggest weakness: capturing concave features. In some cases, this may not be a problem. One case is when high fidelity of solid boundary representation is not very important, usually when the flow is dominated by separation around blunt objects. Another is when the feature is rounded, so further grid refinement may be sufficient to capture the feature. Alternative approach is required, however, when the concave feature is also sharp, hence grid refinement is not a practical choice. This is the subject of the next step: feature recovery.

2.5. Feature Recovery

First the features of the solid surface is extracted. A feature consists of a series of edges. An edge of a feature is shared by two faces whose normal vectors form an angle greater than a specified value.

Next, we need to determine the faces of the surface grid that fails to capture the solid feature. An estimate of approximation error, e, is used. After projection, the position of a surface grid's vertex is exactly on the solid surface. It is reasonable to guess that the worst approximation error is somewhere around a face center. Thus the face center is chosen as a convenient location to estimate the maximum approximation error. In Eq. (3), the distance from the face center to the solid surface, d_S is used for this purpose. Note that the value is normalized using the characteristic length of the element (estimated from its area), in order to put more weight on small elements, which are supposed to capture the solid surface better than the larger ones. A face is said to have a large value of error if it exceeds a threshold value, which is determined from the mean value and standard deviation of the error distribution, as shown in Eq. (4). Here the constant c is taken as unity.

$$e_i = \frac{d_{Si}}{\sqrt{A_i}} \qquad (3)$$

$$e_i > \bar{e} + c\, e_{sdev} \qquad (4)$$

Now we have a set of features and a set of surface grid's faces that have large approximation error. Shown in Fig. 8 is the case where there is one feature and a set of faces. The method to capture the feature is as follows:

1) Capture the endpoints of the feature by moving the closest vertex to these points.
2) Capture the feature curve by traversing from the starting vertex obtained in 1) to the other.

3. Flow Solver

The flow solver is based on finite volume method for Euler equations, currently under extension to Navier-Stokes equations. It is capable of handling a cell of arbitrary shape, in an unstructured format. A cell is allowed to have any number of faces. The face, in turn, can have any number of vertices, which can be listed without any particular order. Thus, all types of cell described above can be treated by this flow solver as a general case. A major advantage of the unified approach is that the code can be streamlined, and no modification is necessary when other types of cells are to be used in the future.

At its present state, both explicit and implicit methods are available for time integration, whereas for flux computation, Roe and AUSMDV methods are used. To achieve second order accuracy, the solution at cell center is extrapolated to cell face. The flow solution gradient is computed using a least square method, which is particularly well suited for unstructured grid containing cells of irregular shape.

4. Results

The grid generation algorithm outlined above has been successfully applied in the following cases. All grid generations are automatic, and the time consumed is quite reasonable. All results exhibit globally smooth and fine grid, although in some region such as wing's leading edge, especially toward the tip, refinement is still not sufficient. At this stage, the flow solution is shown only to demonstrate the integrity of the grid. A study of the solution accuracy will be performed later.

4.1. ONERA M6 Wing

This is a simple case, because the solid surface is either convex or flat. The number of cells, faces, and vertices are around 600,000, 1,900,000, and 700,000 respectively. The time to generate the grid is less than 15 minutes (CPU time). The grid is shown in Figs. 2-5 during the generation process, and the final grid is in Fig. 9. The pressure distribution is shown in Fig. 10, which suggests that the well-known lambda-shaped shock pattern on upper surface is captured.

4.2. ONERA M5 Aircraft Model

This is a more difficult case, because now we have concave region in the wing-fuselage junction. However, since the concave feature is rounded, it can be handled simply by refining the cells in that region. The final grid is shown in Figs. 11-13. In Figs. 12 and 13 we can see that the fillet in wing-fuselage junction is well captured. Other fillets in the horizontal-tail junction and vertical-tail junction are also captured. The number of cells, faces and vertices are around 400,000, 1,100,000, and 400,000, respectively. The time to generate is less than 15 minutes. The flow solution is shown in Fig. 14. As before, the lambda-shaped shock pattern is also captured.

4.3. DLR F6 Aircraft Model

This is a difficult case, because there are sharp concave regions in the wing-fuselage junction and the engine pylon junctions. The final grid is shown in Figs. 15-18. The grid contains around 800,000 cells, 2.6 million faces, and 1 million nodes. The surface mesh contains 124,000 faces. The time to generate is around 30 minutes. Note that even though the grid is reasonably fine around the features, improvement is still needed, as can be seen in Fig. 16. In Fig. 17, the feature lines of the solid surface are shown, together with faces of large approximation error (which can be seen as dark spots), as computed using Eqs. (3,4). These faces appear most prominently in the regions of wing-fuselage junction, pylon junctions, engine inlet, and the leading edge of the outboard wing. Of these regions, only the junctions are sharp feature. The other regions are rounded, but they are still not sufficiently resolved. This demonstrates that the error indicator, Eqs. (3,4), is quite effective in pinpointing the problematic area. The traversing method captures the feature cleanly, as shown in Fig. 18.

5. Concluding Remarks

A hexahedra grid generation method has been presented. The results suggest that the method is quite effective. It is automatic and fast. It is also quite robust, at least for the class of geometries tested here. As a future work, we would like to be able to capture sharp features of more complex geometry, and to improve the overall grid quality, especially in the concave region, as well as the region interfacing Cartesian grid and prismatic grid. The assessment for proper construction of prismatic grid is also currently underway, together with the completion of the extension to viscous flow solver. Higher order projection is also being considered, because in some cases, the original solid surface grid (on which the re-meshing is performed) may not be fine enough. By incorporating these points, we believe the method can still be improved significantly.

References

1) Yamane, T., Yamamoto, K., Enomoto, S., Yamazaki, H., Takaki, R., and Iwamiya, T., "Development of A Common CFD Platform – UPACS," Proc. Parallel CFD 2000 Conf., Elsevier Science, 2001, pp. 257-264.
2) Kallinderis, Y. and Ward, S., "Prismatic Grid Generation for Three-Dimensional Complex Geometries," AIAA Journal, Vol. 31, No. 10 , 1993, pp.1850-1856.
3) Lohner, R., "Generation of Unstructured Grids Suitable for RANS Calculations," AIAA-99-0662, 1999.
4) Aftosmis, M.J., "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries," VKI Lecture Series, 1997-02, 1997.
5) Lahur, P.R. and Nakamura, Y., "Anisotropic Cartesian Grid Adaptation," AIAA 2000-2243, 2000.
6) Ito, Y. and Nakahashi, K., "Unstructured Hybrid Grid Generation based on Isotropic Tetrahedral Grids," AIAA 2002-0861, 2002.
7) Kallinderis, Y., Khawaja, A., and McMorris, H., "Hybrid Prismatic/Tetrahedral Grid Generation for Viscous Flows around Complex Geometries," AIAA Journal., Vol. 34, No. 2, 1996, pp. 291-298.
8) Leatham, M., Stokes, S., Shaw, J.A., Cooper, J., Appa, J., and Blaylock, T.A., "Automatic Mesh Generation for Rapid-Response Navier-Stokes Calculations," AIAA 2000-2247, 2000.
9) Deister, F. and Hirschel, E.H., "Adaptive Cartesian/Prism Grid Generation and Solutions for Arbitrary Geometries," AIAA 99-0782, 1999.
10) Karman, S.L.Jr., "SPLITFLOW: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries," AIAA 95-0343, 1995.
11) Tchon, K.F., Hirsch, C., and Schneiders, R., "Octree-based Hexahedral Mesh Generation for Viscous Flow Simulations," AIAA 97-1980, 1997.
12) Wang, Z.J. and Chen, R.F., "Anisotropic Solution-Adaptive Viscous Cartesian Grid Method for Turbulent Flow Simulations," AIAA Journal, Vol. 40, No. 10, 2002, pp. 1969-1978.
13) Wang, Z.J. and Srinivasan, K., "An Adaptive Cartesian Grid Generation Method for 'Dirty' Geometry," Int. J. Numer. Meth. Fluids, Vol. 39, 2002, pp. 703-717.
14) Bonet, J. and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," Int. J. Numer. Meth. Eng., Vol 31, 1991, pp. 1-17.
15) Baumgart, B.G., "Winged-edge Polyhedron Representation for Computer Vision," National Computer Conference, 1975. Also available from: http://www.baumgart.org/winged-edge/winged-edge.html

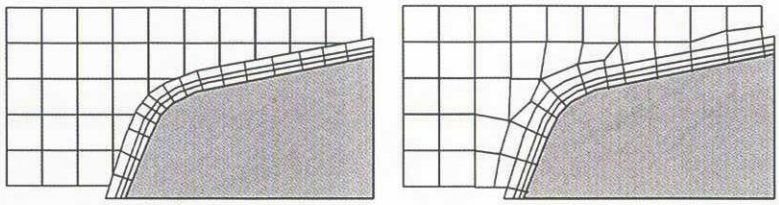

Figure 1. Cartesian grid with prismatic layers: (left) cutcell approach, (right) hexahedra approach.

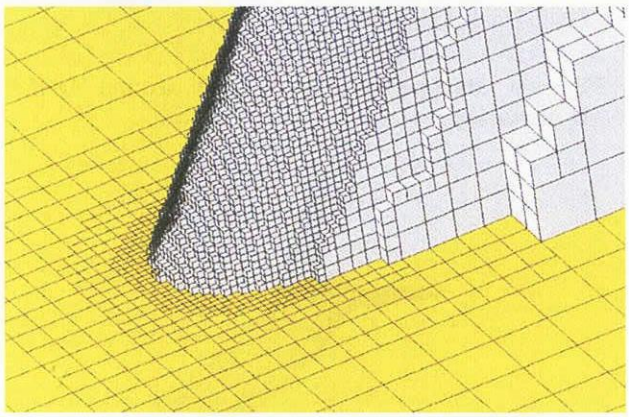

Figure 2. Surface grid from Cartesian grid front.

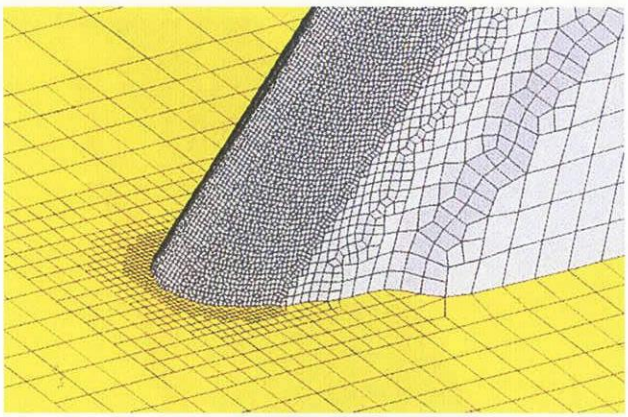

Figure 3. Smoothed surface grid.

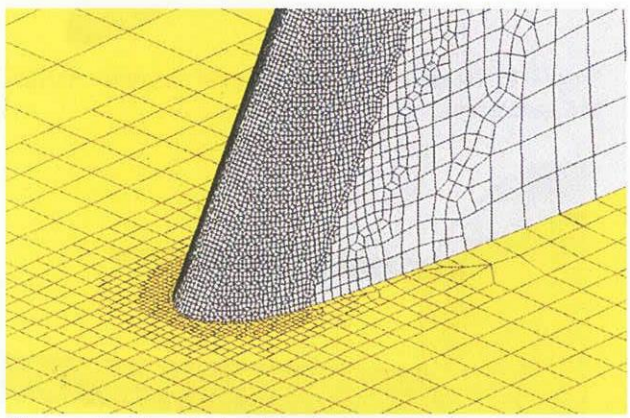

Figure 4. Hexahedra layer.

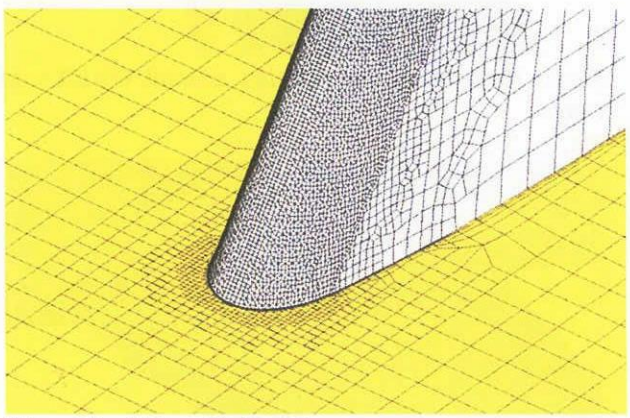

Figure 5. Prismatic layers.

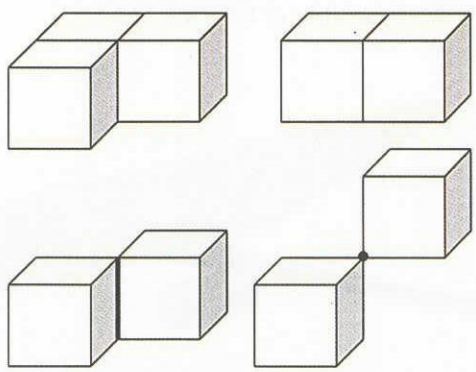

Figure 6. (top) Legal cell configuration, (bottom) illegal cell configuration.

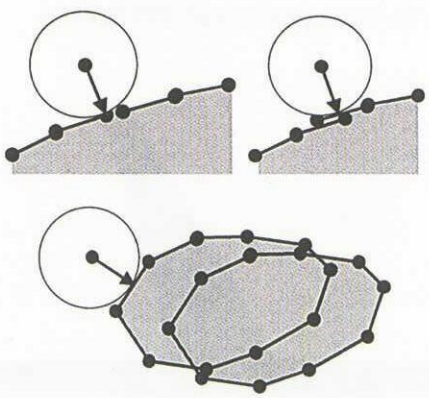

Figure 7. Projection in the case of gap, overlapped elements, and internal elements.

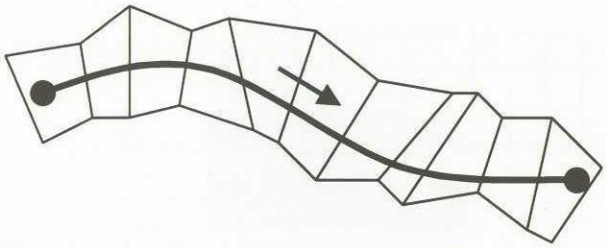

Figure 8. Feature recovery by traversing.

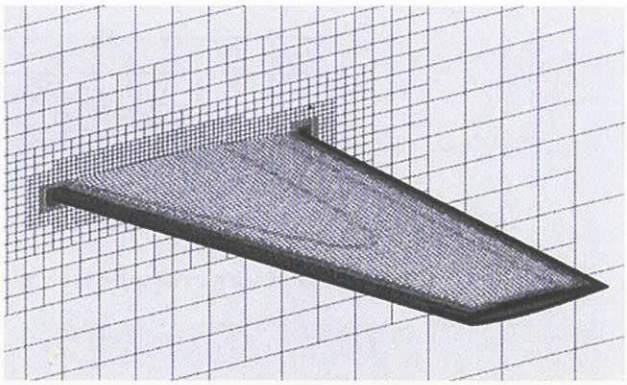

Figure 9. Hexahedra surface grid around ONERA M6 geometry.

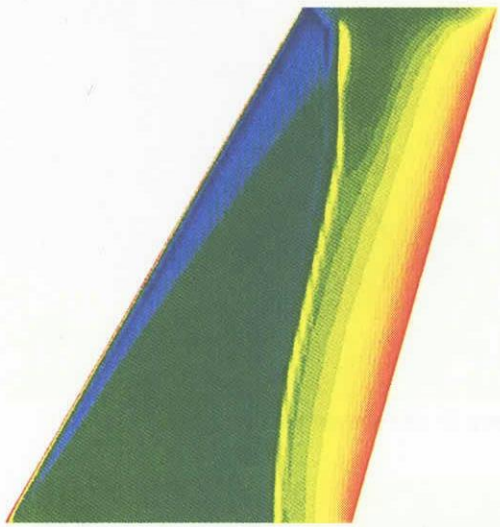

Figure 10. Pressure distribution around ONERA M6 wing.

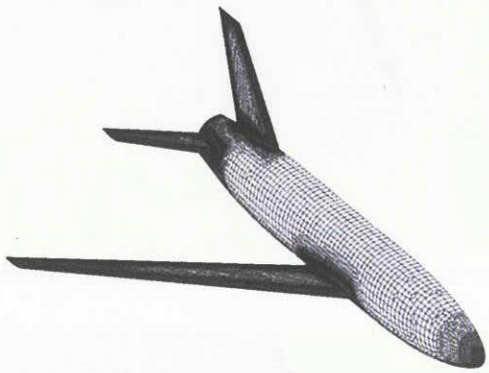

Figure 11. Hexahedra surface grid around ONERA M5 geometry.

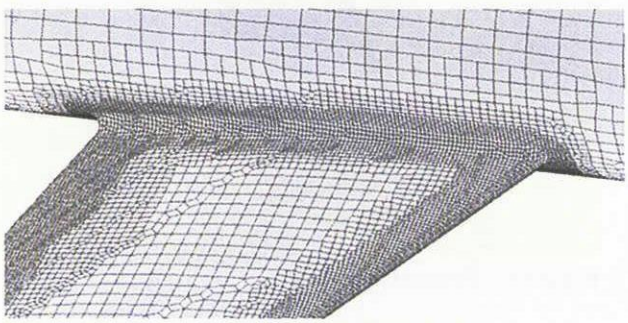

Figure 12. Wing junction of ONERA M5.

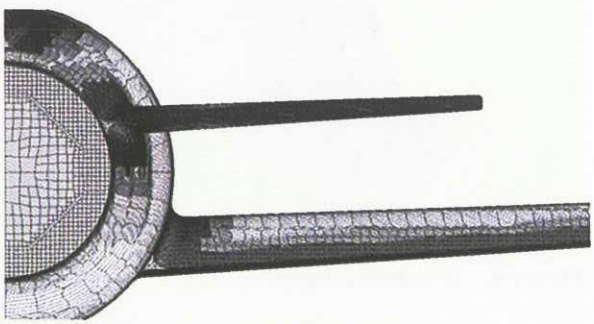

Figure 13. Rear view of ONERA M5.

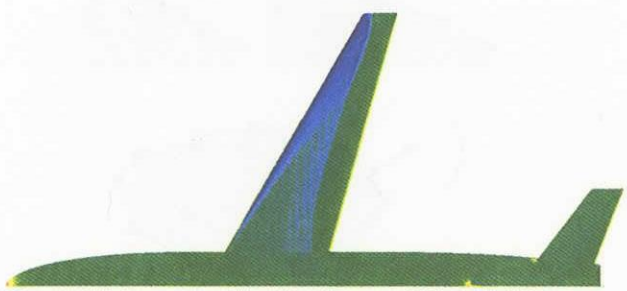

Figure 14. Pressure distribution on ONERA M5.

Figure 15. Hexahedra surface grid around DLR F6 geometry.

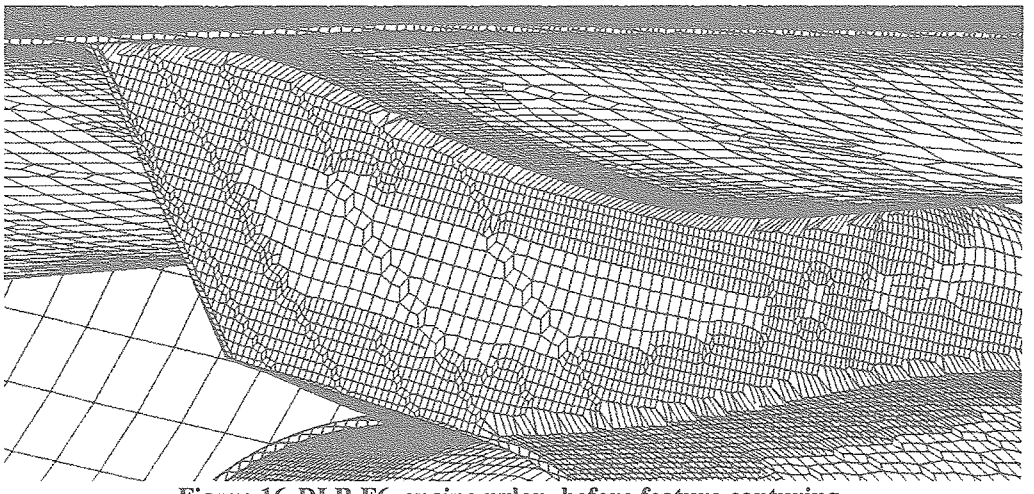

Figure 16. DLR F6, engine pylon, before feature capturing.

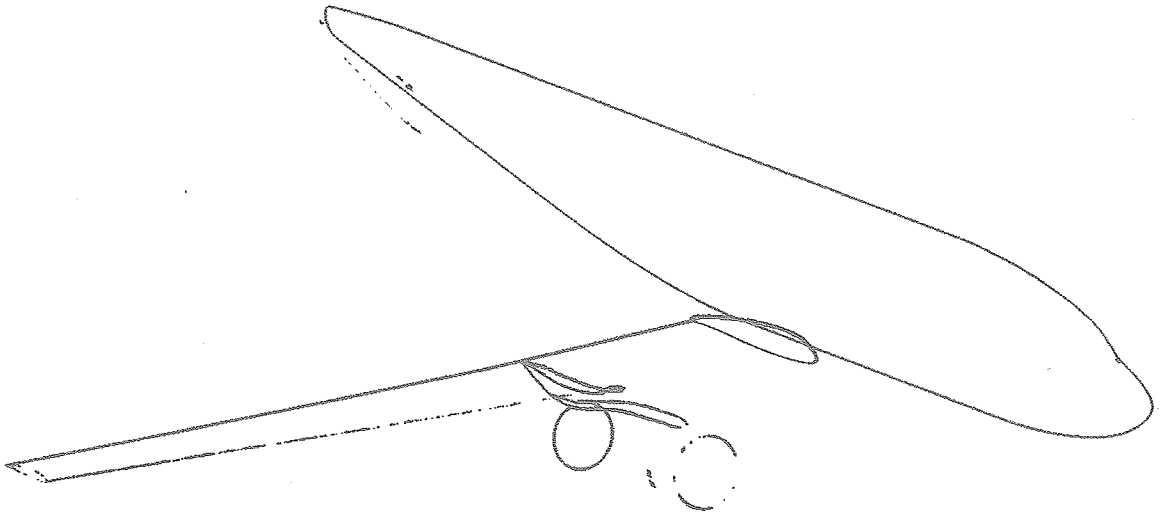

Figure 17. Features of DLR F6 geometry (grey lines), and face elements that have large approximation error (dark spots).

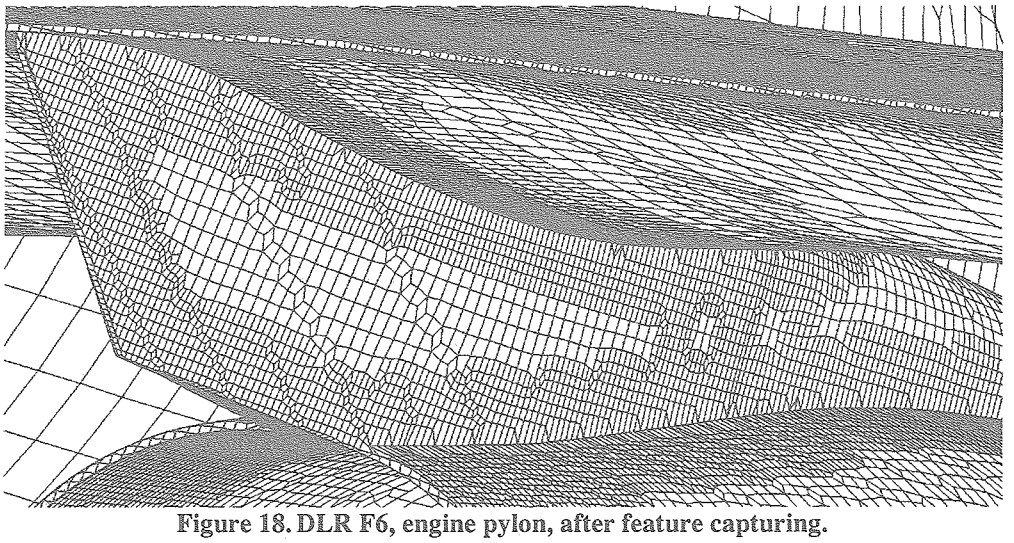

Figure 18. DLR F6, engine pylon, after feature capturing.