Robustness Study on Automatic Hexahedra Grid Generation

by
Paulus R. Lahur
Japan Aerospace Exploration Agency

ABSTRACT

This paper discusses the robustness aspects of an automatic hexahedra grid generation method. The method is capable of automatically generating hexahedra grid around solid surface comprising overlapping components, offering a significant benefit in design environment, because the geometry of individual component and their relative positions can be readily modified. The method is based on Cartesian grid method, which is well known for its efficiency and speed in filling a computational domain. The cells near solid surface are hexahedral cells of general shape, which includes degenerate forms. The cells are generated from the interior of computational domain toward the solid surface, resulting in tolerance to surface defects such as small gap and overlap, which are not uncommon. The robustness issues and their solution at each step of the method are discussed in this paper. In particular, the most difficult issue: capturing sharp concave features, has been successfully addressed. The method has been applied to a number of geometries, including a model of DLR F6 with engine and pylon.

1. Introduction

For treatment of complicated geometry, there are mainly three major types of grid, know by their popular names as: (1) "structured grid," (2) "unstructured grid," and (3) "Cartesian grid." The "structured grid" popularly known today is actually a hybrid approach between structured and unstructured grid. The grid consists of a number of blocks. An example can be found in Ref. [1]. Within the block, the grid is structured, whereas between the blocks the relationship is unstructured. The blocks either fit each other on the boundaries, or overlap each other. Although in the hands of an expert this type of grid can produce accurate solution, it takes a considerable amount of manual effort and time to generate. It is not uncommon for such grid generation to take months, making it impractical for many design purposes.

The "unstructured grid" addresses the difficulties faced by structured grid [2-4]. Its generation is automatic and fast. Although in principle unstructured grid may consist of any cell shape, the most popularly employed shape is tetrahedron. This shape, which is the simplest of all polyhedra, along with its unstructured nature, is responsible for its advantages. However, it is also this shape that puts limit on its solution accuracy.

The method outlined here belongs to the third type: "Cartesian grid" [5-13]. Its generation is fast and automatic. Furthermore, it has a preferred shape in terms of solution accuracy. The issue to consider with this type of grid is how to treat the solid surface properly. Because a method that can generate grid for inviscid and viscous flows is desirable in this study, a hybrid with prismatic grid has to be considered. There are basically two options in generating such hybrid (Fig. 1):

1) Hybrid with cutcell approach, where prismatic grid is generated first on the solid surface (Boundary to Interior approach) [6-8].
2) All-hexahedra grid, where the prismatic grid is generated toward solid surface (Interior to Boundary approach) [9-13].

The present research follows the second approach because, unlike the first method, it does not require high quality surface grid. A solid surface does not have to be water-tight, thus gaps between surface elements and element overlaps are allowed (Fig.2). Recently the present method has been extended to direct treatment of solid surface consisting of components [13]. This is highly advantageous in design environment.

The grid generation method is outlined below, describing topics such as how to overcome the problem of capturing sharp concave feature, a common problem to all Interface to Boundary approach. Discussion on robustness is given, with special attention given to the most problematic issues. Test cases that include DLR F6 with engine nacelle and pylon are presented as well.

2. Grid Generation Method

This method can generate computational grid around a solid surface consisting of a set of triangular elements. Compared to other methods, the requirement imposed on the solid surface is not strict. Water-tight surface is not required, so gap between elements is allowed, provided that its size is small in comparison to local size of grid cells. The elements may overlap and intersect each other. Furthermore, there is no strict requirement on the size of elements, although it is desirable that they are sufficiently small so as not to degrade solution accuracy significantly. An important requirement is that the normal vectors of all solid elements must point outward.

The method consists of the following steps, as described in [13]:
(1) Cartesian grid generation
(2) Removal of cells and construction of "quad surface"
(3) Construction of layer of cells
(4) Construction and capturing of solid features
(5) Quality improvement

Brief description and discussion on robustness for each step is given below, along with a sample of intersecting spheres (Fig. 3). Note that the robustness of the method as a whole is determined by the least robust step. It is therefore very important to address robustness issue at each step.

2.1. Cartesian Grid Generation

This step starts with one Cartesian cell that covers the whole computational domain. Refinement is applied on the cell and repeated on the resulting smaller cells until a targeted grid resolution is achieved. The refinement is carried out by dividing a cell into two in each Cartesian axial direction, resulting in a total of eight child cells of equal size and shape (isotropic division). As in all grid generation methods, the grid resolution is made higher around the solid surface, and even higher if the surface is curved.

This step is very robust. It relies on a well known octree data structure, which also allows for a very efficient grid generation. In fact, this is the reason that makes Cartesian grid especially attractive. Extensive discussion can be found in [5].

2.2. Removal of Cells and Construction of "Quad Surface"

Cells around solid surface are removed to make room for the construction of hexahedral cells in the next step (See Section 2.3). Faces of the remaining cells around solid surface form a surface termed "quad surface," because almost all of these faces are of quadrilateral shape. The rest of faces are not quadrilateral, because they contain hanging nodes that occur when the size of a cell differs to its neighbor, a typical characteristic of methods based on Cartesian grid.

The quad surface is in fact a geometric approximation of the wetted surface of the solid body, albeit a crude one, due to its

staircase-like appearance and its "inflated" geometry. The surface is then smoothed to ease the next task (Section 2.3).

This step is also robust. The potential pitfall is failure to form a valid quad surface. Note that a valid surface is defined here as one that does not self-intersect, even if the intersection is merely on a single point. Self-intersecting surface will result if cells are removed in such a way that *the only way to travel from one cell to its neighbor is through an edge or a point*. See Fig. 4. In this implementation, during cell removal, check for such case is performed. When found, the offending cells are removed.

Another possible problem is due to a gap on solid surface that is bigger than local grid cells. The algorithm mistakenly assumes that the gap is indeed a valid hole on the solid surface, resulting in failure to remove cells *inside* the solid body. The gap has to be closed or at least made smaller before a proper grid generation. The result of this step is shown in Figs. 5 and 6.

2.3. Construction of Layer of Cells

In this step, first a new quad surface is constructed. Initially it is just a copy of the quad surface from the previous step. Connecting the two surfaces is a layer of cells. The new surface is then snapped onto the solid surface. Note that a snap is defined as moving an object to the closest target object. In this case, nodes on the new surface are moved to the closest position on solid surface. As the result, all nodes of the projected quad surface now lie exactly on the wetted solid surface. See Fig. 7.

This step is very robust, because the algorithm works even when the solid surface contains gap and faces that overlap and intersect each other. One pitfall is when a gap is too big in comparison to local grid cell size, resulting in snapping into the inside of solid body. However, if such gap does exist, it already causes problem in the previous step.

2.4. Construction and Capturing of Solid Features

The snapping procedure in the previous step captures the solid surface very well, except in the region where the surface is highly concave. The very nature of the algorithm prohibits snapping onto concave features. In this step, such deficiency is corrected. More attention is given to the discussion for this step, because it is the most problematic in terms of robustness. The source of the difficulty is the complexity of solid surface, which usually has numerous features. The problem is made worse by the existence of gap and element intersection.

There are two independent tasks in this step: (1) to construct solid features and (2) to capture the features. The first step is not difficult to implement, because the procedure boils down to simple operations involving a pair of triangles. Features are formed by intersection between components of solid surface, as well as by sudden change in angular orientation of the surface. See Fig. 8.

Depending on the complexity of the geometry, the features may form a simple line, or they may form a complicated arrangement. It is found that the key to successful feature capturing is to capture only valid concave features. A concave feature is either within a component, or the result of intersection between components. Thus these other features are ignored:

(1) Convex feature, because the snapping at the previous step has already captured this.
(2) Features inside solid body and that on "the other side" of solid body, because they violate geometry integrity.

The second task starts with identifying which quad faces fail to capture the solid surface properly. Proposed in this study is the use of "approximation error" of quad surface in capturing the wetted solid surface. This is computed for each quad face by measuring the maximum distance between the face to the solid surface. It is found that simply measuring the distance from the center of quad face is inaccurate when the face is not flat, resulting in compromised robustness. Therefore, for a non-flat face, distances are measured from the mid-points of *all* combinations of the face's vertices, and the maximum value is taken as the approximation error. See Fig. 9. Quad faces with large approximation error are the ones that will be fixed.

Both identification of valid concave features and identification of the quad faces with large error are crucial in isolating the problem. Now the problem domain is reduced to a much simpler set of sharp concave features and a handful of quad faces with large approximation error. The next job is to fit the quad faces to the features. All scenarios in fitting one quad face to a feature line are shown in Fig. 10. Because of the simplification above, most situation falls into this case. There are a number of possible ways to fit a feature line, where one, two, or three node(s) of quad face may be moved onto the line.

The feature capturing takes place from one problematic quad face to another next to it. When moving the neighbor face to a solid feature, care is taken so that it moves in synergy with the face that already captures the feature. This synchronizes the movement of the quad faces sharing a feature line, and resulting in a robust and good quality capture. The result is shown in Fig. 11.

The techniques above reduce the complexity of the feature construction and capturing. It is found that this is the key to a successful treatment and the solution to the robustness issue.

2.5. Quality Improvement

Although the feature is captured, the grid quality around the feature is still quite low in term of smoothness. Improvement can be achieved by means of selective smoothing, where quad faces that already capture solid features are prohibited from moving.

This step is very robust because it only involves a smoothing procedure, whose algorithm is well tested. See Fig. 12.

3. Test Cases

3.1. Simple Configuration of DLR F6 Aircraft Model

The model consists of body and wing, as shown in Fig. 13. Figures 14-16 show the final hexahedra grid, which contains about 200,000 cells, 700,000 faces, and 250,000 nodes. The time to generate is around 7 minutes on a Pentium4-based machine. As can be seen, the grid is rather coarse, especially that around the wing. It is indeed the purpose for this case to demonstrate that the method works quite successfully even when the grid is coarse. As in other methods of grid generation, increasing the resolution of the grid makes the task easier to perform.

3.2. More Complex Configuration of DLR F6 Aircraft Model

The model consists of four components: fuselage, wing, engine nacelle and pylon, as shown in Fig. 17. Figures 18-20 show the final grid, which contains about 900,000 cells. The time to generate is around 50 minutes on a Pentium4-based machine. The amount of time is very short when compared with grid generation methods that rely on manual labor such as multi-block structured grid, which can take time in the order of weeks. This is indeed a significant advantage in a design environment. As in the first case, this case is also handled successfully.

4. Conclusion

A brief discussion on the robustness aspects of a hexahedra grid generation method has been presented. The method can generate grid around solid surface expressed as overlapping components. Cartesian grid cells fill most of the computational domain, whereas generic hexahedral cells fill the region around solid surface. Because the method generates grid towards the solid surface instead of from the surface, it tolerates surface defects such as gaps and element overlaps. Recent advancement in this study exploits this property further by generating grid around intersecting geometry components.

It is found that feature capturing causes the most trouble in terms of robustness, due to the complexity of solid surface. However, it is also found that, by reducing the complexity of the problems, as discussed in the relevant section, a robust algorithm can be constructed. A test case of DLR F6 geometry comprising of body, wing, engine nacelle and pylon has been used to demonstrate the capability of the method with satisfactory results.

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) Lohner, R., "Generation of Unstructured Grids Suitable for RANS Calculations," AIAA-99-0662, 1999.
3) Ito, Y. and Nakahashi, K., "Unstructured Hybrid Grid Generation based on Isotropic Tetrahedral Grids," AIAA 2002-0861, 2002.
4) 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.
5) Aftosmis, M.J., "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries," VKI Lecture Series, 1997-02, 1997.
6) Deister, F. and Hirschel, E.H., "Adaptive Cartesian/Prism Grid Generation and Solutions for Arbitrary Geometries," AIAA 99-0782, 1999.
7) 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.
8) Karman, S.L.Jr., "SPLITFLOW: A 3D Unstructured Cartesian/Prismatic Grid CFD Code for Complex Geometries," AIAA 95-0343, 1995.
9) Tchon, K.F., Hirsch, C., and Schneiders, R., "Octree-based Hexahedral Mesh Generation for Viscous Flow Simulations," AIAA 97-1980, 1997.
10) 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.
11) 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.
12) Lahur, P.R.., "Hexahedra Grid Generation Method for Flow Computation," AIAA 2004-4958, 2004.
13) Lahur, P.R. "Automatic Hexahedra Grid Generation Method for Component-based Surface Geometry," AIAA 2005-5242, 2005.

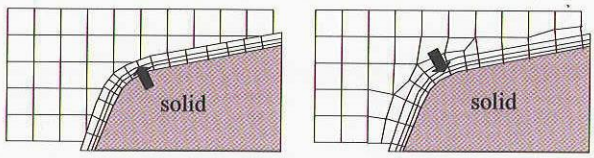

Figure 1. Comparison between the approach of Boundary To Interior (left) and Interior To Boundary (right).

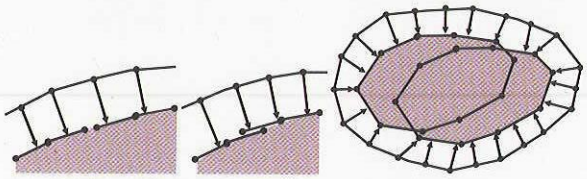

Figure 2. Solid surface that contains gap (left), element overlap (middle), and component overlap (right).

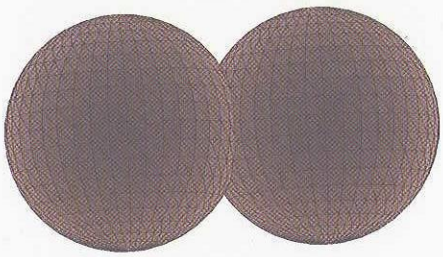

Figure 3. Two solid surfaces intersecting each other.

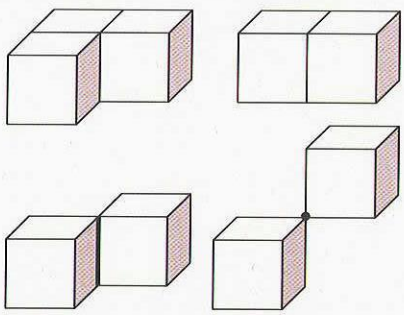

Figure 4. Legal configurations (top) and illegal configurations (bottom).

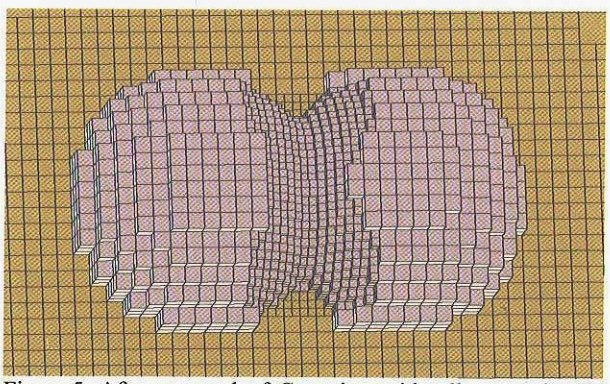

Figure 5. After removal of Cartesian grid cells around solid body.

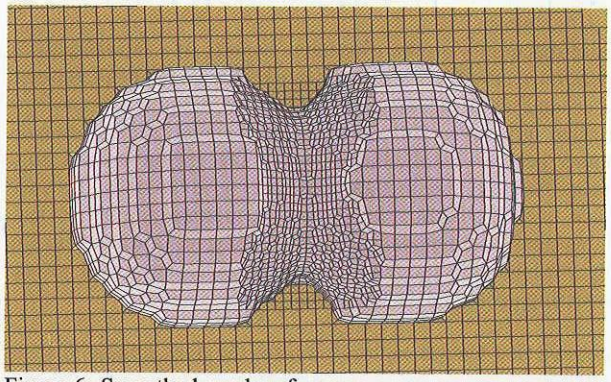

Figure 6. Smoothed quad surface.

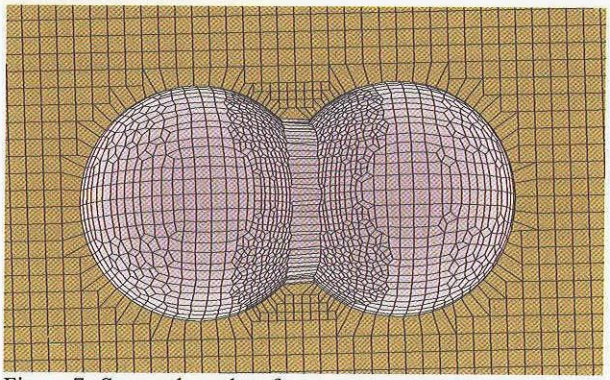

Figure 7. Snapped quad surface.

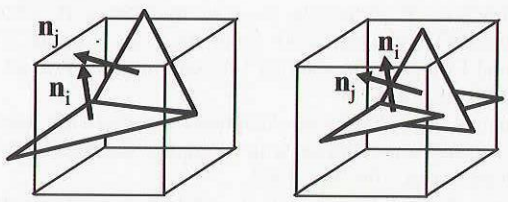

Figure 8. Cartesian grid cell containing solid faces that forms non-planar surface in a single component (left) and multi-component (right).

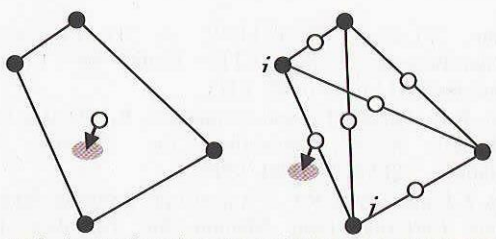

Figure 9. Approximation error of quad face, estimated as the distance to solid surface, measured from face center (left) and midpoints of vertices (right).

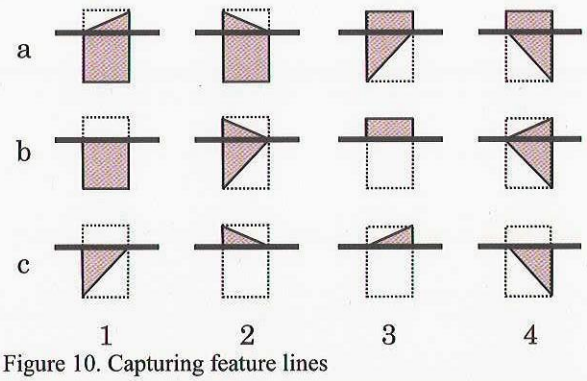

Figure 10. Capturing feature lines

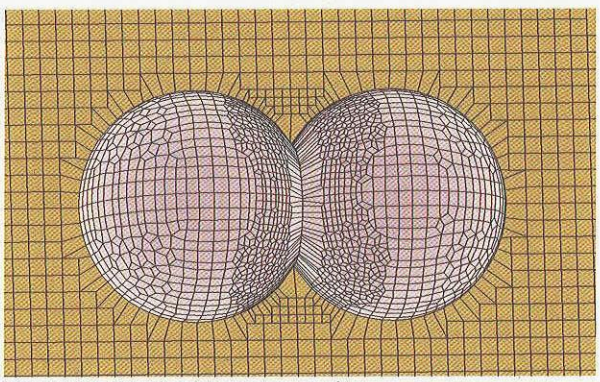
Figure 11. Result of feature capturing.

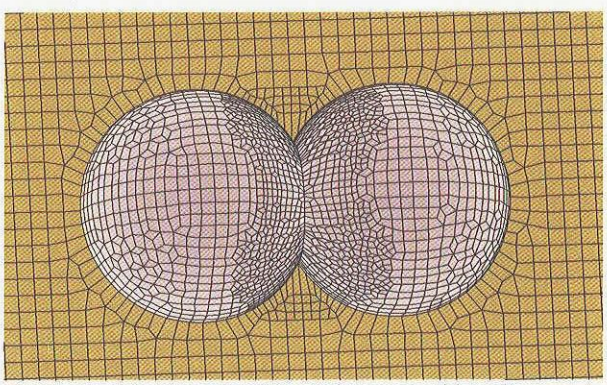
Figure 12. Selective smoothing to improve grid quality.

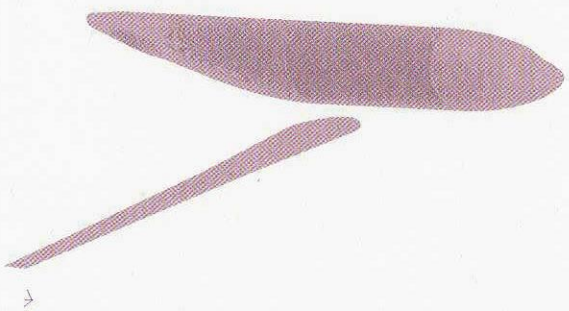
Figure 13. Exploded view of DLR F6 model, comprising fuselage and wing.

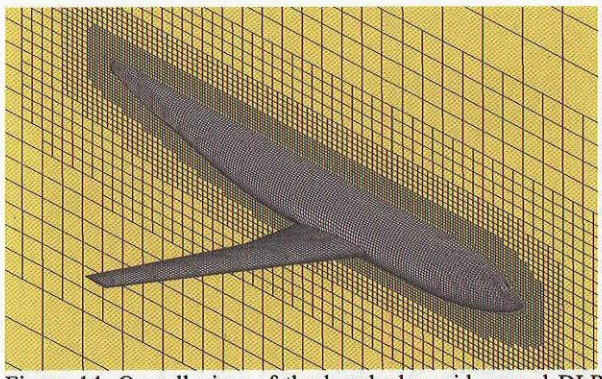
Figure 14. Overall view of the hexahedra grid around DLR F6.

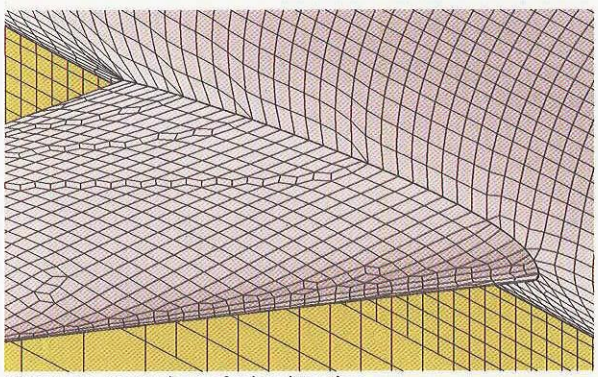
Figure 15. Top view of wing junction.

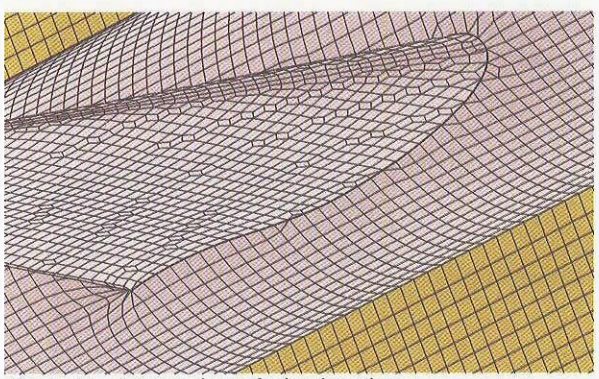
Figure 16. Bottom view of wing junction.

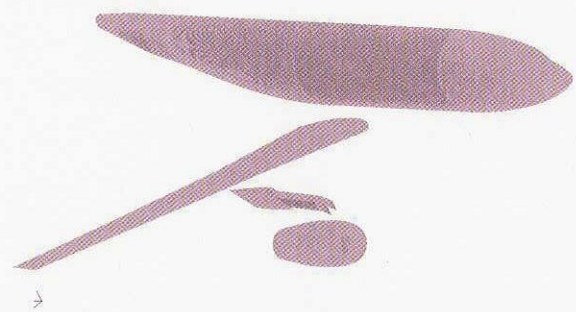
Figure 17. Exploded view of DLR F6 model, comprising fuselage, wing, engine nacelle and pylon.

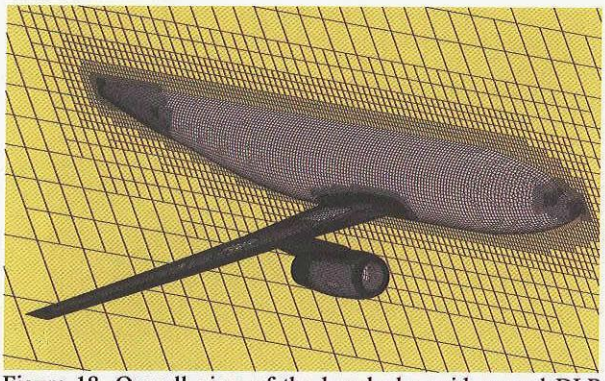

Figure 18. Overall view of the hexahedra grid around DLR F6.

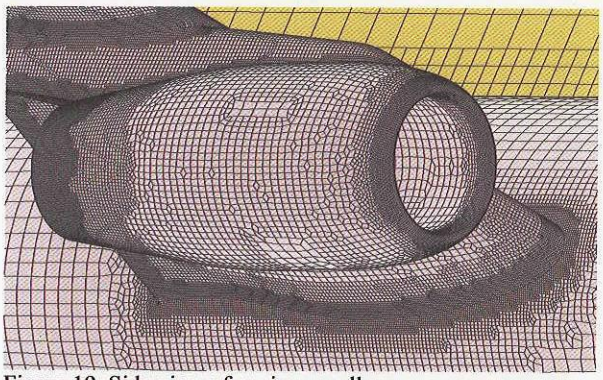

Figure 19. Side view of engine nacelle.

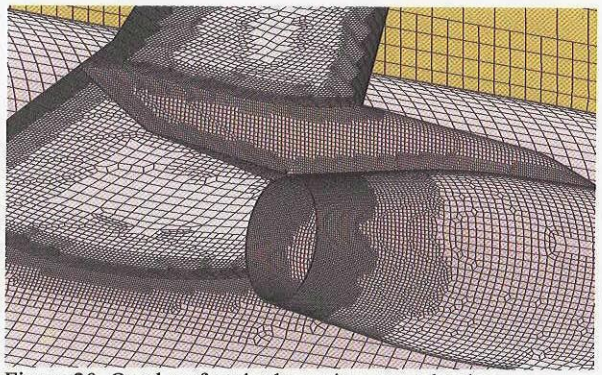

Figure 20. Quad surface in the region around pylon.