AIAA JOURNAL
Vol. 36, No. 2, February 1998

Hybrid Prismatic/Tetrahedral Grid Generation for Viscous Flow Applications

Dmitri Sharov* and Kazuhiro Nakahashi†
Tohoku University, Sendai 980-77, Japan

A method for the automatic generation of unstructured grids composed of tetrahedra and prisms is proposed. The prismatic semistructured grid is generated around viscous boundary surfaces and covers viscous regions, whereas the tetrahedral grid covers the rest of the computational domain. The Delaunay approach for tetrahedral grid generation is used. The proposed prismatic grid is structured in directions normal to the boundary faces, but the number of prisms generated from one boundary face is variable from face to face. Unlike conventional prismatic grid generators, this technique works well even in regions of cavities and gaps. The Delaunay background grid generated for surface nodes serves as an efficient data structure to check possible intersections of prisms. Particular attention is given to the boundary-constraining problem. A robust algorithm for the boundary recovery by edge swapping followed by a direct subdivision of tetrahedra is used. Grid examples for internal and external flow problems of complex shapes demonstrate the efficiency of the method.

I. Introduction

FLUID dynamics computations in complex geometries impose heavy demands on the generation of computational grids. Unstructured grids offer great promise for the solution of this problem because of their flexibility to fit complex computational geometries and their refinement capabilities. There are many general and robust algorithms for the generation of two-dimensional grids in a complex domain; however, their extension to three dimensions still involves a problem, and automatic generation of three-dimensional unstructured grids is recognized to be one of the central problems of computational fluid dynamics unstructured grid computations. Grid generation for viscous flow applications is especially complicated because different requirements are imposed on viscous and inviscid parts of the flowfield.

Usually viscous flow regions require high-aspect-ratio grid cells. The problem is the generation of such stretched elements and the avoidance of large grid angles (obtuse triangles/tetrahedra) leading to exceedingly high truncation error. The only reliable way of achieving stretched elements in the context of unstructured grids is to generate 90-deg triangles or rectangles in shear layer regions. In the three-dimensional case, tetrahedra should contain an edge that is normal to one of its faces or prismatic and/or hexahedral elements should be generated. Depending on the capability of the flow solver to handle different types of elements, the prisms and hexahedra can be divided into tetrahedra (see, for instance, Refs. 1 and 2). However, a more efficient technique is to retain prismatic or hexahedral elements and, moreover, to merge tetrahedra to prisms, if possible.[3,4] This merging can considerably reduce the number of cell interfaces, which directly influences the code efficiency. Mavriplis and Venkatakrishnan[5] have shown that merging can save up to 30% of the computational time.

Available unstructured viscous grid generation algorithms can be divided into three groups. One group involves fully unstructured uniform algorithms, which use different point placement strategies in viscous and inviscid regions. Among them are viscous extensions of the advancing front method: the advancing layer method of Pizadeh[6] and the normal point placement technique for the Delaunay method proposed by Rebay[7] and Müller[8] for two-dimensional cases and by Marcum[3] for two- and three-dimensional cases. These methods provide fully triangular/tetrahedral grids.

The second group also produces a tetrahedral grid, but directional refinement is used to generate stretched tetrahedra in viscous regions.[9,10]

The third group involves hybrid algorithms, which treat viscous and isotropic grids separately and merge them into one nonoverlapped grid. This approach seems to be attractive because most natural methods can be used for both viscous and inviscid regions. These methods usually start from a structured or prismatic grid generator for a stretched grid and apply the advancing front method to generate tetrahedra in the remainder of the domain (see, for instance, Refs. 1, 2, and 11). Fully prismatic grids[12] are suitable for external flow problems with fairly simple shapes; however, their two-dimensional analog is a conventional rectangular structured grid. Thus, prismatic grids cannot take full advantage of unstructured grids to fit complex geometry. Retaining the prismatic structure from one triangular layer to another might produce a distorted grid in regions of cavities and gaps. Kallinderis et al.[11] applied the generation of prismatic grids only in viscous regions, and the rest of the domain was filled by tetrahedra using the advancing front method; however, the use of this method for gridding of cavities and gaps still might be problematic. The effective technique here is to remove structured grid elements (prisms or hexahedra) when they become distorted or intersect each other. This idea was applied by Löhner[2] on a structured grid in the viscous region and used by Connell and Braaten[1] for prisms. After such a semistructured grid is generated, the rest of the domain is filled by tetrahedra using the advancing front approach.

The objective of this paper is to propose a new efficient, robust, and automatic method for hybrid grid generation in complex geometries. The hybrid algorithm presented here follows the idea of mixing semistructured prismatic and tetrahedral grids, but the Delaunay method is used to generate tetrahedra and to obtain a search data structure for checking prism intersections.

The choice of the method for tetrahedral grid generation is crucial for the efficiency and robustness of the code. The advancing front or Delaunay (incremental insertion) approach is usually used. The Delaunay incremental insertion method seems to be attractive for the following reasons.

1) Unlike the advancing front method, the Delaunay method requires no global search or special tree-search algorithms.

2) The Delaunay method does not require the generation of a separate background grid, which usually requires user input. Moreover, a very natural background grid is automatically obtained after the insertion of boundary points into triangulation and boundary recovery.

Received April 24, 1996; presented as Paper 96-2000 at the AIAA 27th Fluid Dynamics Conference, New Orleans, LA, June 17–20, 1996; revision received Sept. 2, 1997; accepted for publication Sept. 16, 1997. Copyright © 1997 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Assistant Professor, Department of Aeronautics and Space Engineering; currently Researcher, Fujitsu Ltd., 9-3, Nakase 1-Chome, Mihama-ku, Chiba 261, Japan. Member AIAA.

†Professor, Department of Aeronautics and Space Engineering. Member AIAA.

3) The Delaunay approach offers the advantage of a strong mathematical basis.

4)The main bottleneck in Delaunay tetrahedral meshing, the boundary recovery problem, can now be solved more or less efficiently.

The issue of three-dimensional boundary recovery merits more detailed discussion. Several approaches to solving this problem have been proposed.[13–17] The simplest method assumes that a tetrahedral mesh contains all faces conforming to the boundary surface. In this case, tetrahedra located outside the computational domain are removed and boundaries of the remaining mesh represent new boundary surfaces. Unfortunately, this method does not work if some tetrahedra penetrate through the boundary surface. Moreover, this method produces new boundary surface triangulation that is not exactly equivalent to the original one. This is unacceptable for hybrid grid generation, where altering of the boundary surface (inflated surface produced by prismatic grid) is not desirable.

Other approaches[13,16] are aimed at preserving the original surface triangulation. The method proposed by Weatherill and Hassan[13] recovers the original boundary surfaces by inserting extra points along the boundary. This method directly cuts all tetrahedra penetrating through the boundary faces and appears to be a very efficient tool for recovering complex boundaries. However, the position of the extra points cannot be controlled, and thus distorted boundary surface triangulation may occur.

The method proposed by George et al.[16] uses three-dimensional edge swapping along with extra point insertion inside the volume grid. This method is mathematically well justified and preserves the boundary surface exactly. However, in practice, the grid quality problems might arise because the location of the extra points cannot be properly controlled. Implementation of the method may also run into problems associated with round-off errors when the extra point position is determined.

We use here a combination of the preceding three approaches that is suitable for the recovery of inflated surfaces. Unlike in Refs. 13 and 16, our method reduces extra point insertion to a minimum based on algorithms described by the authors in Ref. 17. The main feature of the implementation to the hybrid grid generation case is advancing front insertion of boundary points along with boundary recovery by three-dimensional swapping.

In the present approach, the Delaunay method is used not only for the generation of the tetrahedral mesh but also for checking the possible intersections of prisms. The proposed prismatic grid is structured only in directions normal to the boundary surface, but the number of prismatic layers is arbitrary for different boundary faces. Therefore, if an inappropriate prism is produced, the advancement from the face is terminated. As a result, a stepped, inflated boundary surface is formed. The next stage is the construction of a tetrahedral grid starting from this new boundary.

II. General Algorithm

In this section the basic strategy of the proposed grid generation method is given. Details on important steps of the procedure will be considered in following sections. We assume here that surface triangulation is given in advance. We use surface grids obtained by the advancing front surface generator.[18] The proposed hybrid grid generation algorithm is as follows.

1) Generate the Delaunay tetrahedral background grid containing all boundary nodes.

2) Recover all boundary faces.

3) Compute a distribution function for all boundary points.

4) Generate a prismatic grid using the Delaunay grid as a search data structure for the detection of possible prism intersections. Use the distribution function for automatic termination of advancement from each face.

5) Generate a new Delaunay background grid containing new boundary nodes. The inflated surface obtained after the generation of prisms serves as a new boundary surface.

6) Recover all faces belonging to the new boundary and remove the external tetrahedra.

7) According to the point distribution function computed for the boundary points, create new internal points by the Steiner advancing front insertion.

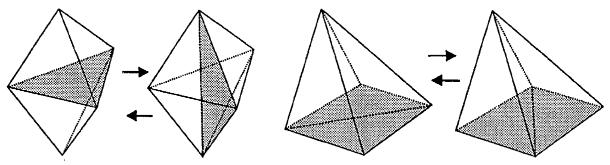
Fig. 1 Swappable five-node configurations.

8) Optimize the tetrahedral grid according to the MinMax criterion and remove possible slivers.

III. Generation of Tetrahedra

The method for tetrahedral grid generation is based on the three-dimensional edge swapping concept described in Ref. 19 and based on the Lawson theorem.[20] This approach has been used, for instance, by Marcum and Weatherill[21] and by the authors.[17,22] There are some advantages of this method over the widely used Watson method. First, this method can produce grids optimized by some mesh quality measure other than the Delaunay circumsphere test. This is important because the three-dimensional Delaunay grid can result in poorly shaped tetrahedra. Second, this method can be applied to recover boundary faces.[16,17] Unfortunately, the boundary recovery by the three-dimensional edge swapping is not always successful, and other heuristic methods should be applied in this case.

The three-dimensional incremental insertion methods are very sensitive to round-off errors. If the error occurs during the convexity test (computation of tetrahedra volumes), the code may generate tetrahedra with negative volume that results in a fatal error in data structure. We use integer arithmetic to overcome this problem.[22]

Details on the procedure of incremental insertion by three-dimensional edge swapping can be found, for instance, in Ref. 22. We give here just a brief description of swapping cases because the procedure is used not only for point insertion but for boundary recovery as well.

For each face to be swapped, we consider a five-node configuration: three nodes belonging to the face and two nodes opposite the face. The configuration is swappable if the five-node configuration is convex.[19] The swappable configurations are shown in Fig. 1. Three cases can be realized.

1) Swapping of two tetrahedra forming a convex configuration to three tetrahedra,

2) Swapping of three tetrahedra forming a convex configuration to two tetrahedra,

3) Swapping of two tetrahedra having two faces belonging to one plane to two tetrahedra.

The criterion of swapping is usually the Delaunay circumsphere test, but other quality measures are possible. If the circumsphere test is used and the insertion is into an existing Delaunay triangulation, the Delaunay grid will be produced.[23]

IV. Advancing Front Insertion of Boundary Nodes Along with Boundary Recovery

The following algorithm minimizes the amount of search during Delaunay incremental point insertion and simultaneously provides recovery of the majority of missing faces by three-dimensional swapping without altering the boundary triangulation. The method is suitable for hybrid grid generation when altering of the boundary surface is not desirable. The algorithm is based on the advancing front concept and consists of the following steps.

1) *Preparation of the front.* The initial front is formed by one triangle. Consider any face from the boundary triangulation, which is not yet recovered, and insert its three vertices into the tetrahedral construction using the Delaunay insertion algorithm. Mark the tetrahedral face coinciding with the boundary face, and put its three edges into the front list. The marked face cannot be altered thereafter. Mark the frontal edges as active.

2) *Advancement of the front.* Pick up any active edge from the front (Fig. 2). Its two endpoints are already inserted into the tetrahedral construction. Locate the next node to be inserted from this edge according to the boundary surface triangulation. To locate the tetrahedron containing the new point, only a few tetrahedra surrounding the frontal edge must be visited. Insert the node directly into the

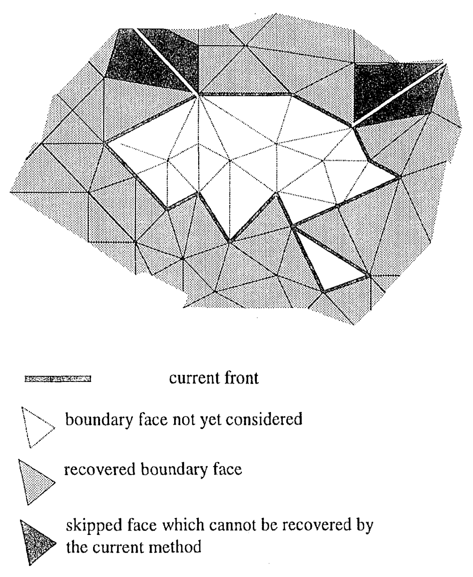
current front
boundary face not yet considered
recovered boundary face
skipped face which cannot be recovered by the current method

Fig. 2 Advancing front recovery/insertion algorithm.

tetrahedron, and apply edge swapping with the Delaunay criterion. Check whether the new boundary triangle exists in the tetrahedral grid. This is a local search through tetrahedra sharing the edge under consideration. If the face does exist, go to step 6; otherwise go to step 3.

3) *Recovery of the edges.* Check all edges of the boundary face. If all edges exist in the tetrahedral mesh, go to step 4; otherwise perform successive three-dimensional edge swapping for all faces intersecting the missing edge; see details in Ref. 17. If all edges are recovered, go to step 4. If some edges cannot be recovered, go to step 5.

4) *Recovery of the face.* If the face is already recovered, go to step 6; otherwise recover the face by successive three-dimensional swapping of all faces intersecting the considered triangle. If the face is recovered, go to step 6; otherwise go to step 5.

5) The face cannot be recovered by three-dimensional swapping. Skip the face. Mark the frontal edge as nonactive, and go to step 2.

6) Mark the face as a boundary face. Update the front. If the front is not empty, go to step 2.

7) If some boundary triangles were not considered (there are two or more separate surfaces), go to step 1.

8) If some faces could not be recovered by the preceding algorithm (black faces in Fig. 2), perform edge swapping of the boundary surface for all edges separating two missing faces. At first, all such edges are swapped according to the local MaxMin (Delaunay) criterion. If some missing edges remain, examine all edges separating two boundary faces that are not yet recovered. If at least one swapped face exists in the tetrahedral construction, swap the edge. If a swapped edge exists in the tetrahedral construction, swap the edge. Repeat swapping until no more edges can be swapped. If the dihedral angle between the two boundary faces exceeds some threshold or the two faces belong to different surfaces, swapping is not permitted.

9) Perform direct division of tetrahedra for all missing faces.

Step 9 implies extra point insertion along the boundary. Actually, its employment is undesirable, and it should be used only if all other methods fail. This might occur, for instance, when two sharp corners are near each other or more than two surfaces are located closer than their point spacing. In such cases, extra nodes are inserted in the triangulation. A direct division of tetrahedra is used to recover the missing faces. This method was proposed originally by Weatherill and Hassan.[13] We have proposed a simplified algorithm, which provides the same result, but the number of cases to consider is significantly reduced.[17]

V. Generation of Prisms

We assume here that the Delaunay background grid composed of boundary nodes is already generated and the boundaries are recovered. This grid serves as an efficient search data structure for finding possible intersections of prismatic layers and for checking local isotropic grid spacing. The prism generation procedure is composed of the following steps.

1) Compute the distribution function for every boundary node as an average of the boundary edge lengths surrounding the node.

2) Identify all viscous boundary surfaces. Each viscous boundary point is associated with a node-normal.

3) Initialize the data structure for normal point placement. The viscous boundary surface serves as an initial zero layer. For each boundary point, save its layer level and pointer to the preceding point. For each normal, assign the termination flag and number of the last layer obtained from the normal. By default, all flags are off and the last-layer numbers are zero. Set the counter of current level l to be generated to 1.

4) Determine face normals for layer $(l-1)$.

5) Determine the node-normal vectors for nodes from layer $(l-1)$. The marching direction is based on the node manifold, which consists of the group of faces surrounding the node to be marched. The normal vectors for any point must be visible from any face surrounding the node and belonging to the same prismatic layer. According to Ref. 11, the node-normal vector lies on the bisection plane of the two faces on the manifold that form the wedge with the smallest angle. Its location on this plane is determined by bisecting the visibility region on the plane.

6) Smooth the node-normals V by weighted Laplacian smoothing:

$$V_i = \frac{1}{\sum_j d_{ij}^{-1}} \sum_j \left(d_{ij}^{-1}\right) V_j^*$$

where V_i^* and V_i are the initial and final marching vectors of node i, respectively; index j denotes all surrounding nodes belonging to the manifold of node i; and d_{ij} is the distance between nodes i and j. Smoothing is not performed for corner points whose manifold angle departs from 180 deg by more than 20 deg. This smoothing step enhances the quality of generated prisms near sharp corners.

7) For current layer l, determine the marching step size $d_l = \Delta n_0 \mathrm{st}^{l-1}$ and compute the characteristic distance between the current layer and the boundary surface:

$$D_l = \Delta n_0 \sum_l \mathrm{st}^{l-1}$$

These distances are fixed for the entire layer and determined only by the stretching factor st and the current layer number l (Fig. 3).

8) For each point from the previous layer $(l-1)$ whose termination flag is off, advance along the smoothed node-normal V_i. The advancing distance d_i is computed as $d_i = d_l(\sin\alpha_i)^{-1}$, where α_i is the minimum angle between the averaged node normal V_i and the manifold of node i. For convex manifolds whose angle departs from 180 deg by more than 20 deg, the advancing distance is decreased by a factor of 0.8 to smooth the convex corners. It is not necessary to increase the marching step for concave corners because, unlike conventional prismatic generators, advancement along any normal can be terminated if a distorted prism is expected. Thus, fine grid is generated inside the concave corners (Fig. 4).

9) Using the background Delaunay grid as a search data structure, check the distance between the prospective point and the closest boundary surface. If the distance is less than the current layer thickness D_l, reject the point and switch the termination flag for this normal to on. This step is used to eliminate possible mutual intersections of prisms.

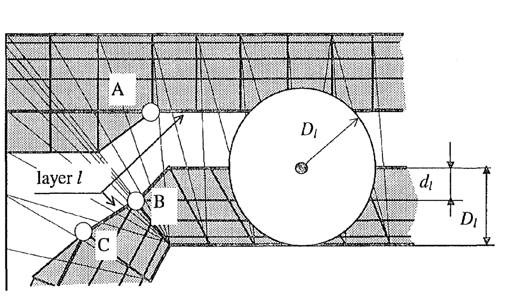

Fig. 3 Definitions of D_l and d_l. Dashed lines correspond to Delaunay background grid.

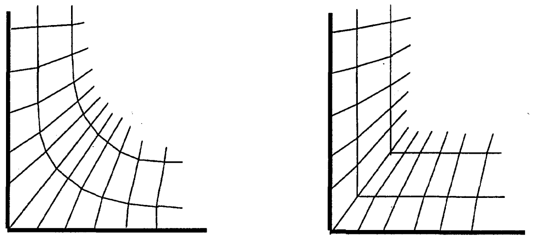
a) Conventional prismatic method b) Our method

Fig. 4 Treatment of concave corner.

10) Using the Delaunay grid, interpolate the distribution function to the prospective point. If the distance from the preceding layer d_l exceeds the distribution function for the prospective point, reject the point and switch the termination flag for this normal to on. This step is a natural termination of stretched grid generation when the marching step becomes comparable to the local isotropic grid spacing.

11) Place accepted prospective points into the normal placement data structure. It is important that we do not insert these points into the Delaunay grid.

12) Examine the shape of newly generated prisms. If a distorted prism is produced, remove its last three vertices and terminate the corresponding normals.

13) For each normal that was advanced, check the surrounding normals. If there are normals whose final layer level differs from the current level by two, reject the newly inserted point and terminate the normal. This is necessary to prohibit jumps of the inflated surface by more than one layer.

14) For each marched normal, check its surrounding normals. If there are normals whose final layer differs from the current level by one, merge the lately inserted point to its origin and terminate the normal (see points A, B, and C in Fig. 3). This step is performed to smooth the inflated surface. Some pyramids and tetrahedra can be obtained on this stage.

15) If some points were inserted, increment the current layer l by 1 and go to step 4.

16) Form new boundary data structure for Delaunay tetrahedral grid generation.

17) If some boundary edge is to be swapped during the boundary recovery procedure, translate the changes back to the prismatic grid. Make sure that no negative volumes occur after this step. If such negative volumes do occur, place a new node into the middle of the considered boundary edge and translate the new node back to the prismatic grid.

Symmetry planes and inlet/exit surfaces are not required to be inflated. We apply special treatment for these surfaces. The algorithm is modified to keep the corresponding normal vectors along the surfaces. After the prismatic grid is generated, the surfaces are retriangulated according to the new inflated boundaries.

VI. Interior Node Generation

The interior nodes are generated by a frontal Delaunay refinement technique similar to those proposed by Marcum[3] and Muller.[8] This technique does not require tracking of the front. The current implementation of the method is as follows.

1) The distribution function for an inflated boundary is already known from the step of prismatic grid generation.

2) The active flag is assigned to all tetrahedra. The flag is off if the tetrahedron cannot be refined. This is assumed to occur when its maximal side length is less than the average distribution function for the tetrahedron.

3) All faces are inspected. If the face is an interface between an active tetrahedron and tetrahedron that is turned off or if the face belongs to the boundary and its neighboring tetrahedron is active, a new point is created by advancing from the face in a direction normal to the selected face. The distance is determined by the average distribution function for the face.

4) The element containing the new point is searched.

5) The distribution function is interpolated to the new point. The existing points whose distance from the new point is less than 0.7 times the point distribution function are searched. If the new point is too close to an existing point, the new point is rejected and step 3 is performed.

6) Insert the new point using the Delaunay incremental insertion algorithm. Assign the active flag to all newly created or altered tetrahedra.

7) Repeat steps 3–6 until no more points can be inserted.

8) Optimize the mesh by swapping with the MinMax criterion.

For some applications, the computational domain may contain considerably different boundary spacing scales. In the case of an airplane, its surface and outer boundary spacings have considerably different scales. This may result in sharp transitions between fine and coarse tetrahedra. Thus, some stretched high-aspect-ratio elements might be generated near leading edges or tips of a wing. To avoid such situations, at step 5 of the original procedure, we determine the distribution function for the new point by taking a minimum instead of interpolating. After the point is inserted, we change its distribution function to the original (interpolated) one.

VII. Examples

Several three-dimensional configurations are presented in this section to demonstrate the capability of the method.

A. Three-Dimensional Internal Geometry

An example of a hybrid grid for an internal geometry is shown in Fig. 5. The geometry represented here is an 11-deg sector of a cylinder. Part of the boundary surface cut by a plane is shown. The upper part shows the prismatic part of the grid. This simple geometry is characterized by a sharp concave corner and two walls that are close to each other. An enlarged area of the corner is shown on the right. Conventional prismatic grids usually have trouble treating such cases. However, the proposed method of removing intersecting prisms provides an automatic and good solution to the problem. Recovery of the inflated surface using the proposed algorithm causes no problem despite the very small distance between two inflated surfaces.

B. Boeing 747

A hybrid grid for a Boeing-747 wing-fuselage-nacelle configuration is presented in Fig. 6. The inflated surface for this case is extremely complex. A part of this surface corresponding to the junction between the wing, pylon, and nacelle is shown (Fig. 6, right). No problems occur during boundary recovery. Some cuts of the grid across the wing are also shown. A smooth transition from prismatic to tetrahedral elements can be observed.

C. ONERA M5

A part of a grid for an ONERA M5 airplane is shown in Figs. 7 and 8. Part of the inflated surface around the tail is shown, as well as the cutting plane across the tail. A stepped structure of the inflated surface can be observed. This can also be seen on the cutting plane. The steps of the inflated surface are a result of the local distribution function, which terminates the generation of prisms where the tetrahedral grid spacing is small (around leading edges and corners). Thus, the prismatic layer is thin around leading edges of wings.

All of these grids were generated automatically, and no extra points were added during the boundary recovery process. The SGI Indy (MIPS R4600, 100 MHz) workstation was used. The required CPU time strongly depends on the ratio between the number of prismatic and tetrahedral elements. Tetrahedra are much more expensive than prisms because edge swapping is required. Generation of 260,000 tetrahedra takes approximately 1 CPU hour. The CPU time for tetrahedral grid generation is greater than that reported by

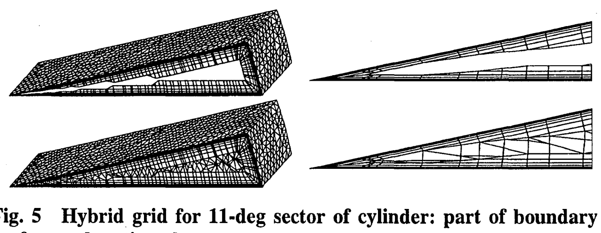
Fig. 5 Hybrid grid for 11-deg sector of cylinder: part of boundary surface and cutting plane.

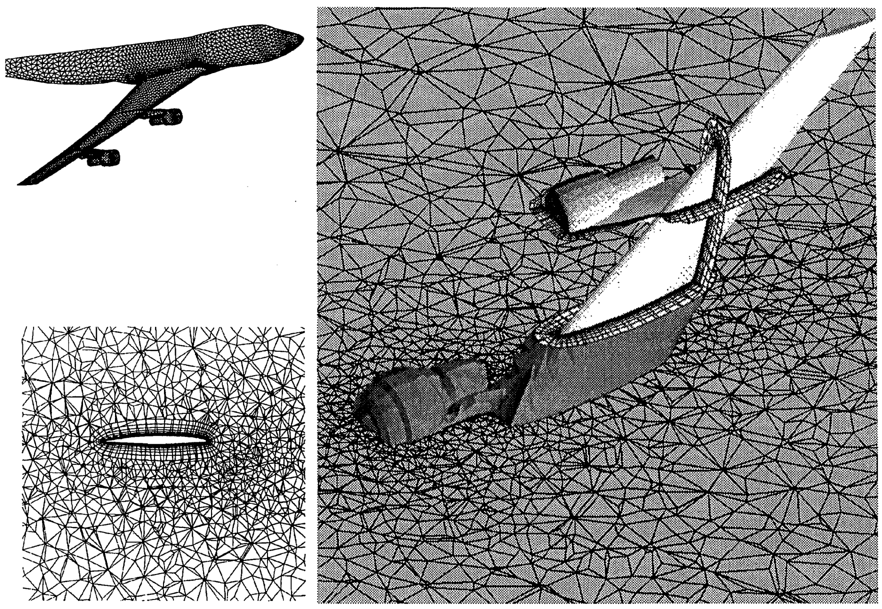
Fig. 6 Hybrid grid for wing-fuselage-nacelle configuration of Boeing 747-200.

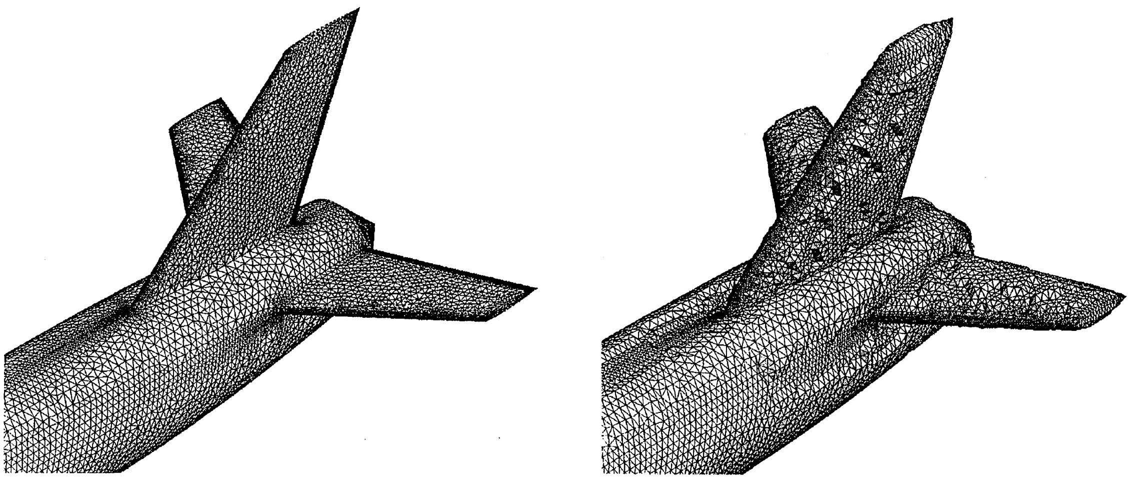
Fig. 7 ONERA M5 grid: original and inflated surfaces around tail.

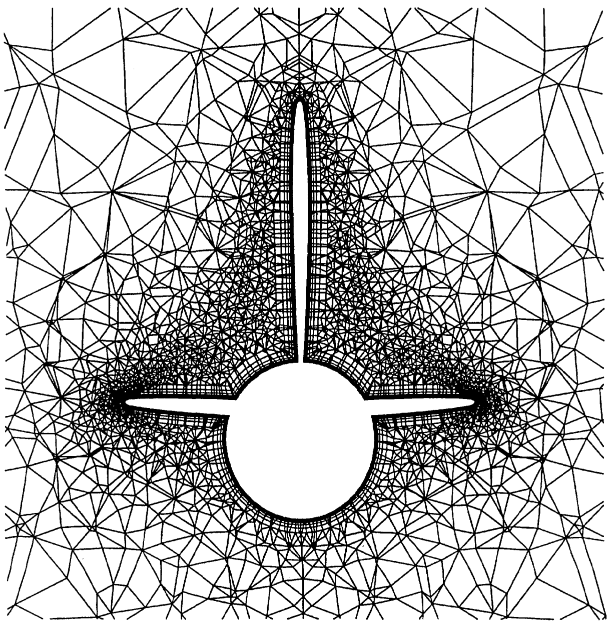
Fig. 8 ONERA M5: grid cut across tail.

other authors (see, for instance, Ref. 21. There are several reasons. One of them is that we do the local edge swapping every time that the new node is inserted. This procedure guarantees retention of the Delaunay property. The other reason is the boundary recovery procedure, which is designed to maintain the original surface triangulation as much as possible, but a price must be paid in an extra CPU time. The required CPU time can be significantly reduced, but it was not necessary in our computations; therefore, our priorities were mainly in the field of robustness and grid quality.

The method is able to generate the stretched grids of any aspect ratio and of any initial normal spacing. This follows naturally from the prismatic grid generation algorithm. The only problem may arise in the regions of sharp concave corners (see, for instance, Fig. 5), where generation of prisms is problematic. However, usually it is not necessary to have high-quality stretched grid there because of the physics of the flow in such regions.

VIII. Conclusions

The proposed method has been shown to be robust for complex geometries. No user intervention is required during the grid generation process, and only the boundary surface should be specified in advance. The efficiency of using the Delaunay background grid as the data structure for checking prism intersections and for the automatic termination of prism generation has been shown. The method is able to generate stretched grids of any respect ratio. The combined advancing front boundary recovery method permits the generation of tetrahedral grids for complex surface geometries.

Acknowledgment

T. Sakawa, whose help is appreciated by the authors, generated the surface grid for ONERA M5.

References

[1]Connell, S. D., and Braaten, M. E., "Semi-Structured Mesh Generation for 3D Navier-Stokes Calculations," *Proceedings of the AIAA 12th Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1995, pp. 369–380 (AIAA Paper 95-1679).

[2]Löhner, R., "Matching Semi-Structured and Unstructured Grids for Navier-Stokes Calculations," *Proceedings of the AIAA 11th Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1993, pp. 555–564 (AIAA Paper 93-3348).

[3]Marcum, D., "Generation of Unstructured Grids for Viscous Flow Applications," AIAA Paper 95-0212, Jan. 1995.

[4]Marcum, D., "Generation of High-Quality Unstructured Grids for Computational Field Simulation," *Proceedings of the 6th International Symposium on CFD*, Vol. 4, 1995, pp. 72–79.

[5]Mavriplis, D. J., and Venkatakrishnan, V., "A Unified Multigrid Solver for the Navier-Stokes Equations on Mixed Element Meshes," AIAA Paper 95-1666, June 1995.

[6]Pizadeh, S., "Unstructured Viscous Grid Generation by the Advancing-Layers Method," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1735–1737.

[7]Rebay, S., "Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer–Watson Algorithm," *Journal of Computational Physics*, Vol. 106, No. 1, 1993, pp. 125–138.

[8]Müller, J.-D., "Quality Estimates and Stretched Meshes Based on Delaunay Triangulations," *AIAA Journal*, Vol. 32, No. 12, 1994, pp. 2372–2379.

[9]Borouchaki, H., Castro-Diaz, M. J., George, P. L., Hecht, F., and Mohammadi, B., "Anisotropic Adaptive Mesh Generation in Two Dimensions for CFD," *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Mississippi State Univ., Mississippi State, MS, 1996, pp. 197–206.

[10]Peraire, J., and Morgan, K., "Viscous Unstructured Mesh Generation Using Directional Refinement," *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Vol. 2, Mississippi State Univ., Mississippi State, MS, 1996, pp. 1151–1163.

[11]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.

[12]Nakahashi, K., "Marching Grid Generation for External Viscous Flow Problems," *Transactions of the Japan Society for Aeronautical and Space Sciences*, Vol. 35, No. 108, 1992, pp. 88–102.

[13]Weatherill, N. P., and Hassan, O., "Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints," *International Journal for Numerical Methods in Engineering*, Vol. 37, No. 12, 1994, pp. 2005–2039.

[14]Baker, T. J., "Shape Reconstruction and Volume Meshing for Complex Solids," *International Journal for Numerical Methods in Engineering*, Vol. 32, No. 4, 1991, pp. 665–677.

[15]Baker, T. J., "Unstructured Meshes and Surface Fidelity for Complex Shapes," AIAA Paper 91-1591, June 1991.

[16]George, P. L., Hecht, F., and Saltel, E., "Automatic Mesh Generator with Specified Boundary," *Computer Methods in Applied Mechanics and Engineering*, Vol. 92, No. 3, 1991, pp. 269–288.

[17]Sharov, D., and Nakahashi, K., "A Boundary Recovery Algorithm for Delaunay Tetrahedral Meshing," *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Mississippi State Univ., Mississippi State, MS, 1996, pp. 229–238.

[18]Nakahashi, K., and Sharov, D., "Direct Surface Triangulation Using the Advancing Front Method," *Proceedings of the AIAA 12th Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1995, pp. 442–451 (AIAA Paper 95-1686).

[19]Barth, T. J., "Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations," Special Course on Unstructured Grid Methods for Advection Dominated Flows, AGARD-R-787, May 1992, pp. 6-1–6-61.

[20]Lawson, C. L., "Properties of *n*-Dimensional Triangulations," *Computer Aided Geometric Design*, Vol. 3, April 1986, pp. 231–246.

[21]Marcum, D., and Weatherill, N., "Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection," *AIAA Journal*, Vol. 33, No. 9, 1995, pp. 1619–1625; also AIAA Paper 94-1926, June 1994.

[22]Sharov, D., and Nakahashi, K., "Constrained Tetrahedral Grid Generation via Edge Swapping," *Proceedings of the 6th International Symposium on CFD*, Vol. 3, 1995, pp. 1117–1122.

[23]Rajan, V. T., "Optimality of the Delaunay Triangulation in R^d," *Proceedings of the 7th ACM Symposium on Computational Geometry*, 1991, pp. 357–372.

D. S. McRae
Associate Editor