CHAPTER EIGHTEEN

Unstructured Grid Generation

The structured grid generation presented in Chapter 17 is restricted to those cases where the physical domain can be transformed into a computational domain through one-to-one mapping. For irregular geometries, however, such mapping processes may become either inconvenient or impossible to apply. In these cases, the structured grid generation methods are abandoned and we turn to unstructured grids where transformation into the computational domain from the physical domain is not required. Even for the regular geometries, an unstructured grid generation may be preferred for the purpose of adaptive meshing in which the structured grids initially constructed become unstructured as adaptive refinements are carried out.

Finite volume and finite element methods can be applied to unstructured grids. This is because the governing equations in these methods are written in integral form and numerical integration can be carried out directly on the unstructured grid domain in which no coordinate transformation is required. This is contrary to the finite difference methods in which structured grids must be used.

There are two major unstructured grid generation methods: Delaunay-Voronoi methods (DVM) and advancing front methods (AFM) for triangles (2-D) and tetrahedrals (3-D). Numerous other methods for quadrilaterals (2-D) and hexahedrals (3-D) are available (tree methods, paving methods, etc.). We shall discuss these and other topics in this chapter.

18.1 DELAUNAY-VORONOI METHODS

A two-dimensional domain may be triangulated as shown in Figure 18.1.1a (light lines). Each side line of the triangles can be bisected in a perpendicular direction such that these three bisectors join a point within the triangle (heavy lines in Figure 18.1.1a), forming a polygon surrounding the vertex of each triangle, known as the Voronoi polygon (diagram) [Voronoi, 1908]. A collection of Voronoi polygons is known as the Dirichlet tessellation [Dirichlet, 1850], and the resulting triangles as Delaunay triangulation [Delaunay, 1934].

Any three points in the plane may be connected by a circle, called the circumcircle (Figure 18.1.1b). The center of this circle, called circumcenter, may (triangle ABC) or may not (triangle DEF) remain within the triangles, although perpendicular bisectors

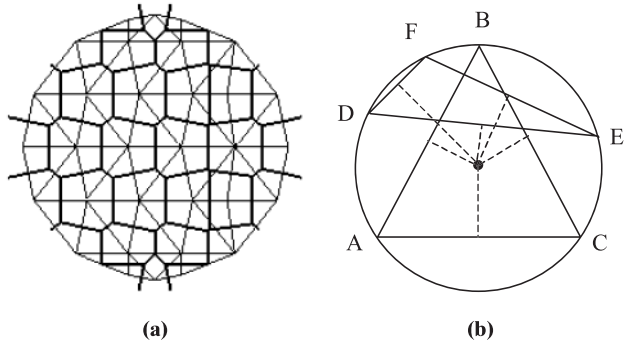


Figure 18.1.1 Delaunay triangulation with Voronoi polygon, and triangle's circumcircle. **(a)** Delaunay triangulation and Voronoi polygon. **(b)** Triangle and its circumcircle.

of sides of both triangles meet at the circumcenter. Here triangle ABC is accepted whereas triangle DEF is rejected, an obvious preference toward a triangle as close to an equilateral triangle as possible versus distorted triangular shapes.

The above geometrical requirements call for distinct points in the plane, P_1, $P_2, \ldots, P_N$ with the sets V_i $(i = 1, 2, \ldots, N)$ such that

$$V_i = \{x : \|x - P_i\| < \|x - P_j\|, \forall j \neq i\} \tag{18.1.1}$$

Thus, V_i represents a region of the plane whose points are nearer to node P_i than any other node and is an open convex polygon (a Voronoi polygon) whose boundaries are portions of the perpendicular bisector lines joining node P_i and P_j when V_i and V_j are contiguous. Connecting the node P of adjacent polygons forms a triangle T_k. The set of triangles $\{T_k\}$ constitutes the Delaunay triangulation.

A triangulation must satisfy the in-circle criterion that no point of the set P_i is interior to the circumcircle of any triangle $T(P_i)$. This criterion may be demonstrated in Figure 18.1.2 in which triangles A-B-C and A-C-D are avoided, but instead triangles A-B-D and D-B-C are chosen.

In implementation of Delaunay-Voronoi methods, various algorithms have been developed. Among them are the Watson algorithm [Watson, 1981; Cavendish, Field, and Frey, 1985] and Bowyer algorithm [Bowyer, 1981; Weatherill, 1992]. These algorithms are presented below for 2-D geometries. Extensions to 3-D geometries are also included.

18.1.1 WATSON ALGORITHM

Consider that three given nodes will form a Delaunay triangulation if and only if the circumdisk (interior of the circumcircle) defined by these nodes contains no other node points in its interior. In effect, Watson's algorithm is to reject, from the set of all possible triangles which might be formed, those with nonempty associated circumdisks. Then, those triangles not rejected form the Delaunay triangulation.

In practice, the procedure begins with inserting nodes, one by one, re-triangularizing upon insertion of each node. The detailed procedure is as follows:

(1) Initialize the algorithm by calculating the x and y coordinates of boundary points surrounding all the node points to be inserted (Figure 18.1.5). Also calculate the circumradius and x and y coordinates of the circumcircle.

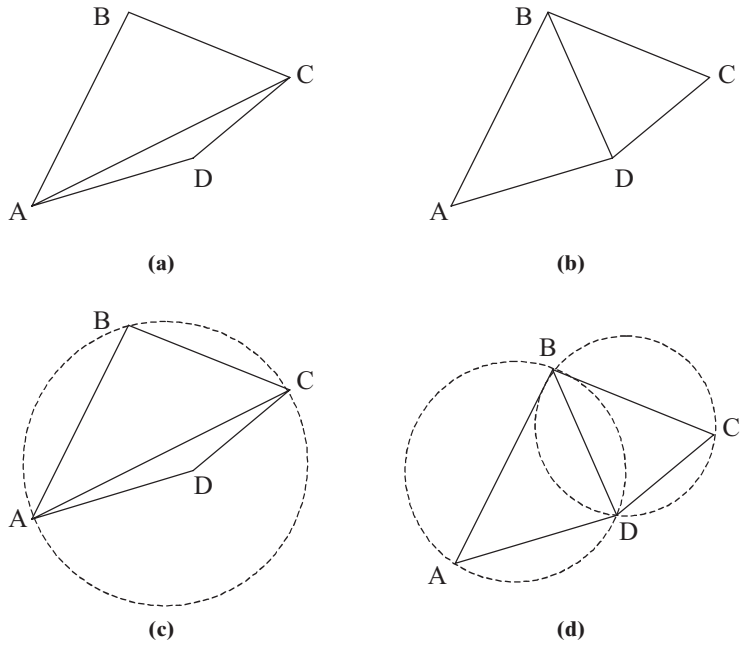


Figure 18.1.2 A triangulation must satisfy the in-circle criterion that no point of the set P_i is interior to the circumcircle of any triangle $T(P_i)$. **(a)** Undesirable triangle, maximum-minimum criterion is not satisfied. **(b)** Desirable triangulation maximum-minimum criterion is satisfied. **(c)** Unacceptable because the circumcircle ABC includes point D interior to the circumcircle. Similarly, if circumcircle ACD is drawn, then B will be interior to it. **(d)** Acceptable because no point is interior to the circumcircles (ABD or BCD).

(2) Introduce a new point.
(3) Conduct a search of all the current triangles to identify those whose circumdisks contain the new point. For each such disk, the associated triangle is flagged for removal.
(4) With the union of all such triangles, an insertion polygon is formed. Here no previously inserted node is contained in the interior of the polygon. Also, each boundary node of the polygon may be connected to the new node by a straight line lying entirely within the polygon. These lines form a new triangulation of the region, which can be shown to be a new Delaunay triangulation.
(5) Repeat Steps 2 through 4 until all nodes have been inserted.

To illustrate the procedure described above, consider triangle 2-4-6 and neighboring triangles 1-2-6, 2-3-4, and 4-5-6 as shown in Figure 18.1.3a. Introduce a new point inside the triangle 2-4-6 (denoted by 7). Each triangle has a circumdisk as defined by the circles containing all three vertices. By default, a new point lies on the circumdisk of the new triangle upon which it was introduced. Check to see if the new point lies within the circumdisk of the neighboring triangles by comparing the distance between the new point and the circumcenter to the radius for each triangle. Point 7 lies within the circumdisks of neighboring triangles 2-3-4 and 4-5-6, but not triangle 1-2-6 as shown in Figure 18.1.3b. Flag those triangles for removal that have circumdisks which contain the new point. In the example, triangles 2-3-4, 4-5-6, and 2-4-6 are flagged for

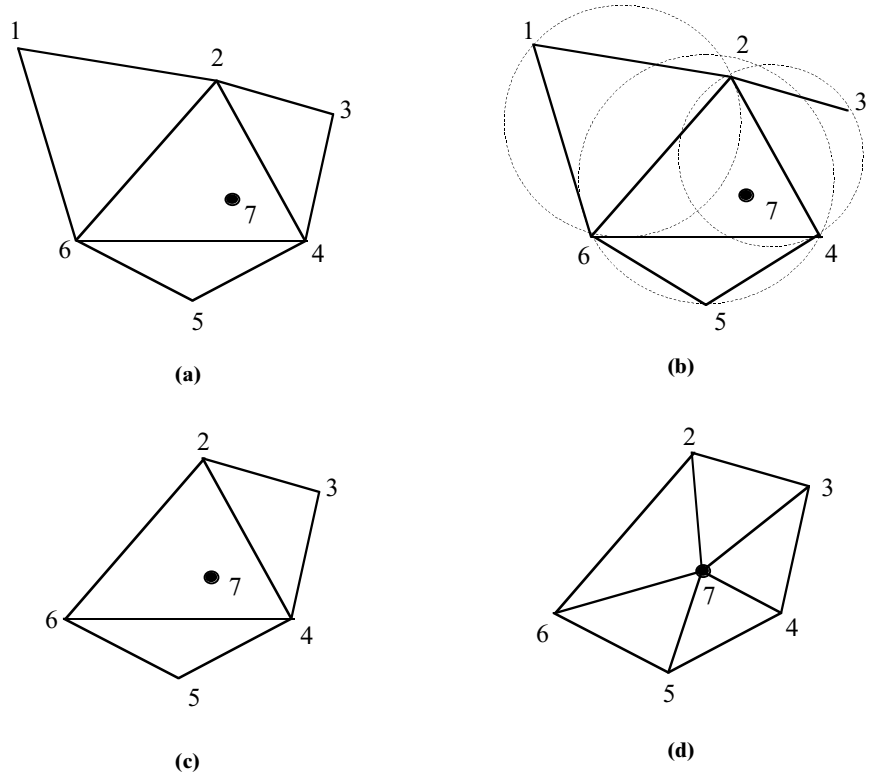


Figure 18.1.3 Watson algorithm of Delaunay-Voronoi method. **(a)** Initial stage. **(b)** New point 7 and circumcircles for all triangles. **(c)** Flag triangles for removal. **(d)** Triangulated insertion polygon and new triangles.

removal. Find the insertion polygon, the polygon remaining after the flagged triangles have been removed. The insertion polygon, which contains the flagged triangles, is shown in Figure 18.1.3c. When the insertion polygon is triangulated, the number of triangles is increased by the number of sides of the polygon. The sides of each triangle are the sides of the polygon plus the straight lines from the end points of the sides to the new point as shown in Figure 18.1.3d.

For 3-D geometries, we begin with a tetrahedron containing all the points to be inserted. New internal tetrahedra are formed as points are inserted one-by-one. This is done by testing a new point to determine which circumballs (interior of circumsphere) of existing tetrahedra contain the point. These tetrahedra are removed, leaving an insertion polyhedron which contains the new point. New tetrahedra are created by forming edges connecting the new point to all triangular faces of the insertion polyhedron. There exist some difficulties with the Watson algorithm which must be addressed: These are the problems of degenerate cases and slivers [Cavendish et al., 1985].

Degenerate cases occur in practice when a newly inserted node appears to lie on the surface of the circumsphere associated with an existing tetrahedron. The problem becomes apparent when the distance from a newly entered nodal point to an existing circumsphere is less than the accumulated truncation error. When this happens, there is a danger of making an incorrect or inconsistent decision regarding acceptance or rejection of a given tetrahedron. This in turn produces structural inconsistencies in

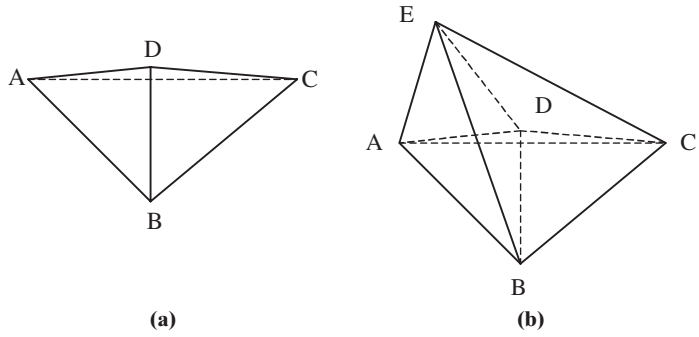


Figure 18.1.4 Treatment of undesirable of elements: **(a)** silver (badly distorted, D being slightly out of the plane of A-B-C) **(b)** Share a common vertex at E.

the mesh (overlapping tetrahedra or gaps in the mesh). A solution to this problem is to slightly perturb the coordinates of a newly entered point whenever that point is found to lie ambiguously on a circumsphere. At the completion of the triangulation, all perturbed nodes are restored to their original positions.

A sliver is a thin, badly distorted tetrahedron whose faces are well-proportioned triangles but whose volume can be made arbitrarily small (Figure 18.1.4a). In practice these are identified when the ratio

$$a = \frac{\textit{radius of inscribed sphere}}{\textit{radius of circumsphere}}$$

becomes "small" (less than 0.01). Slivers are removed in one of two ways, depending on how the tetrahedron fits into the mesh. Consider a tetrahedron $ABCD$ which is determined to be a sliver (Figure 18.1.4b). First we must determine the four tetrahedra that neighbor $ABCD$. If two of these share a common vertex, say node E, the sliver is removed from the collection of tetrahedra, and elements $\{ABDE, BCDE\}$ are replaced by elements $\{ABCE, ACDE\}$. When no two of the surrounding tetrahedra share a common vertex, the node point D is arbitrarily moved to improve the aspect ratio of the sliver.

Finally, we must post-process the mesh to obtain the final mesh over the given geometry. The above described process leads to a triangulation of the original tetrahedron. The tetrahedra associated with interior element nodes are distinguished because they have none of the four initial points as vertex. Of these interior tetrahedra, we remove the ones that lie outside of the geometry to be meshed. These are the ones whose centroids lie outside of the boundary surface.

For illustration, let us consider triangulation of a circle. The step-by-step procedure is described as follows:

(1) First of all, we define the convex hull within which all points will lie. Specify required points as shown in Figure 18.1.5a.
(2) Introduce a new point. Check to see if the new point lies on the circumdisk and if the distance from the new point to the circumcenter is less than the circumradius. Flag those triangles that contain the new point. Find the insertion polygon, the polygon remaining after the flagged triangles have been removed. First, identify the flagged triangles. Then, for each side of the triangles, check on the neighbor

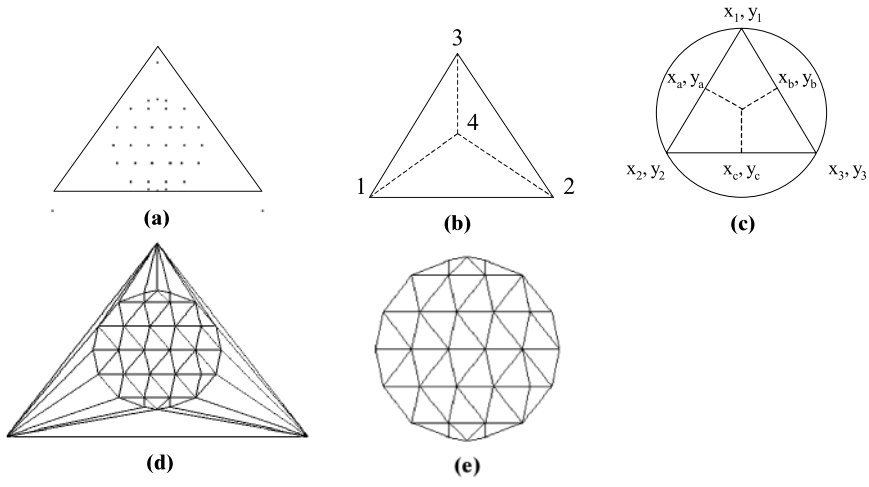


Figure 18.1.5 Illustration of the triangulation of a circle. **(a)** Convex hull. **(b)** Introduce a new point. **(c)** Circumradius and circumcenter. **(d)** Delaunay triangles with convex hull. **(e)** Final grids.

triangle to that side. If that triangle is to be removed also, then the side is not part of the polygon. If that triangle is not to be removed, or is a boundary, then that side is part of the polygon. When the polygon is triangulated, the number of triangles is increased by the number of sides of the polygon. The sides of each triangle are the side of the polygon plus the straight lines from the end points of the side to the new point.

(3) The insertion polygon resulting from adding point 4 is shown as solid lines and the triangulation results in the dashed lines as shown in Figure 18.1.5b.

(4) Calculate the circumradius and circumcenter, as seen in Figure 18.1.5c, the slope of the perpendicular bisector being the negative reciprocal of the slope of the triangle sides.

$$m_a = \frac{x_1 - x_2}{y_2 - y_1} = \frac{y_a - y_{center}}{x_a - x_{center}}$$
$$m_b = \frac{x_2 - x_3}{y_3 - y_2} = \frac{y_b - y_{center}}{x_b - x_{center}}$$
$$m_c = \frac{x_1 - x_3}{y_3 - y_1}$$

Thus, we obtain

$$y_{center} = y_a - m_a(x_a - x_{center})$$
$$m_b = \frac{y_b - y_a + m_a(x_a - x_{center})}{x_b - x_{center}}$$
$$m_b(x_b - x_{center}) = y_b - y_a + m_a(x_a - x_{center})$$
$$x_{center}(m_a - m_b) = y_b - y_a + m_a x_a - m_b x_b$$
$$x_{center} = \frac{y_b - y_a + m_a x_a - m_b x_b}{m_a - m_b}$$

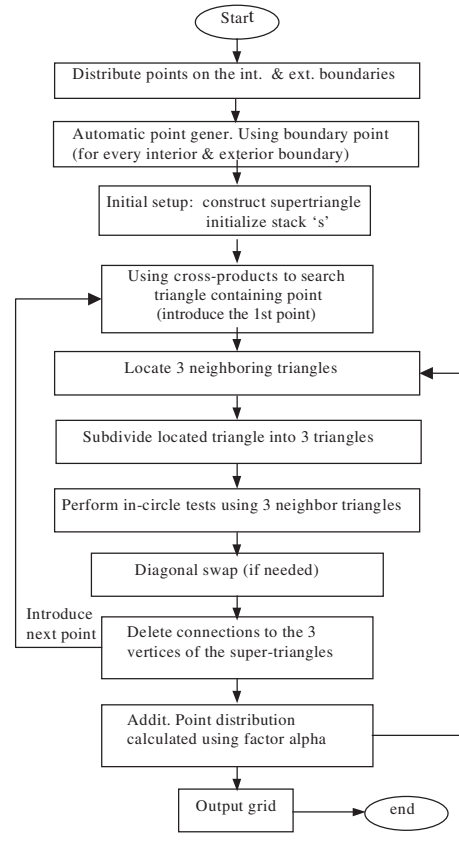


Figure 18.1.6 Delaunay-Voronoi-Watson algorithm flow chart for airfoil grid generation.

This gives the circumradius

$$r = \sqrt{(x_1 - x_{center})^2 + (y_1 - y_{center})^2}$$

(5) Degenerate case. This occurs when a newly inserted node appears to lie on the surface of a circumcircle/circumsphere. This can be resolved by slightly perturbing the coordinates of the newly entered point.
(6) The procedure described above leads to the results shown in Figure 18.1.5d,e.

The computer code flow chart and examples for mesh generation of a circle using the Delaunay-Voronoi method with Watson algorithm are shown in Figure 18.1.6 and Figure 18.1.7, respectively.

18.1.2 BOWYER ALGORITHM

In this algorithm, we utilize the forming points (points which define a Delaunay triangle and Voronoi vertex (vertex of a Voronoi polygon) as shown in Figure 18.1.8.

We recognize that it is possible to completely describe the structure of the Voronoi diagram and Delaunay triangulation by constructing two lists for each Voronoi vertex. These are a list of forming points for the vertex, and a list of the neighboring Voronoi vertices.

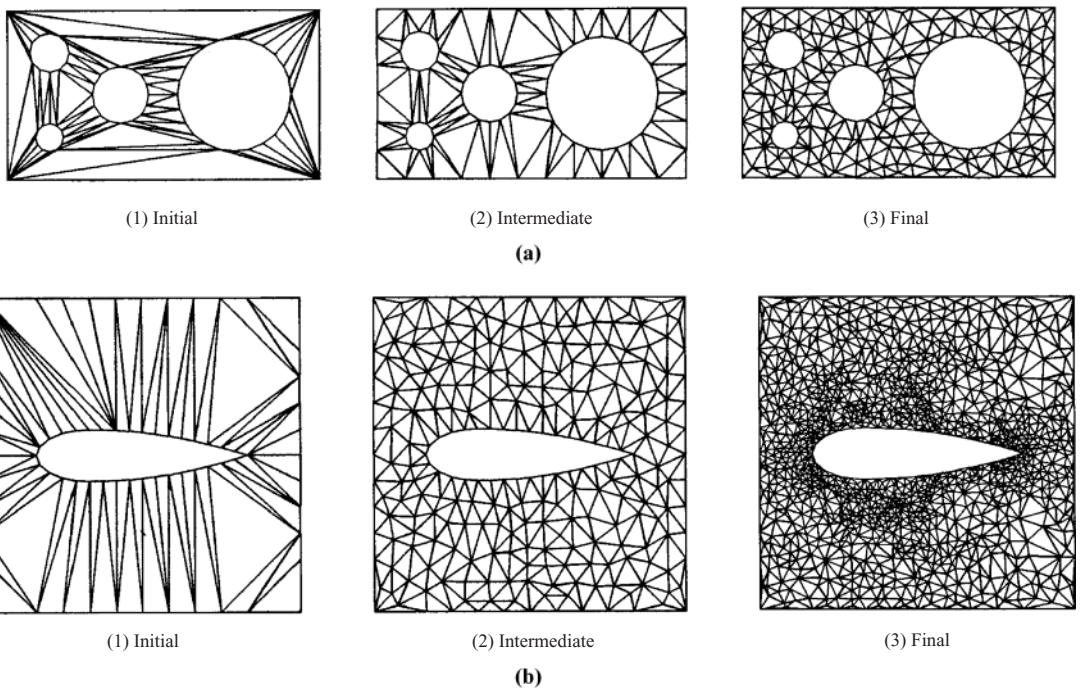
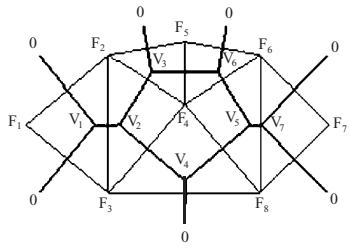


Figure 18.1.7 Delaunay-Voronoi triangulation with Watson algorithm. **(a)** Triangulation around circles. **(b)** Triangulation around airfoil.



Vertex	Forming Point	Neighboring Vertices
V_1	$F_1F_2F_3$	V_2 0 0
V_2	$F_2F_3F_4$	$V_1V_3V_4$
V_3	$F_2F_4F_5$	V_2V_6 0
V_4	$F_3F_4F_8$	V_2V_5 0
V_5	$F_4F_6F_8$	$V_4V_6V_7$
V_6	$F_4F_5F_6$	V_3V_5 0
V_7	$F_6F_7F_8$	V_5 0 0

Figure 18.1.8 Forming points (F_1-F_8) and Voronoi vertices (V_1-V_7).

Similar to the previously described Watson algorithm, this is a sequential process. Each new point is introduced into the structure, one at a time, and the structure is reformulated onto a new Delaunay triangulation. The steps are as follows:

(1) Define a convex hull within which all points will lie. Specify four points with the associated Voronoi diagram.
(2) Introduce a new point.
(3) Determine all vertices of the Voronoi diagram to be deleted. A vertex to be deleted is one whose circumcircle (defined by three forming points) contains the new point. This is similar to step 3 in Watson's algorithm.
(4) Find the forming points of deleted Voronoi vertices, which are contiguous points to the new point. This is similar to step 4 of Watson's algorithm in which the new point is connected to the insertion polygon by straight lines.
(5) Determine the neighboring Voronoi vertices to the deleted vertices which have not been themselves deleted. These data provide the necessary information to enable valid combinations of contiguous points to be constructed.
(6) Determine the forming points of the Voronoi vertices. These must include the new point together with two other points which are contiguous to the new point, and form an edge of the neighboring triangle.
(7) Determine the neighboring Voronoi vertices to the new Voronoi vertices. From step 6, the forming points of all new vertices have been computed. For each new vertex, conduct a search through the forming points of the neighboring vertices found in step 5 to identify common pairs of forming points. When a common combination occurs, then the two associated vertices are neighbors of the Voronoi diagram.
(8) Reorder the Voronoi diagram data structure overwriting the entries of deleted vertices.
(9) Return to step 2 until all points have been inserted.

This process will generate regions that are both interior and exterior to the domain. For grid generation purposes, it is necessary that such triangles which are not within the domain of interest be removed before the next step of the procedure. To do this, in the initial generation of the list of points defining the physical domain, the outer domain boundary points should be listed in a counterclockwise fashion while any and all interior boundaries be listed in clockwise fashion. With this method, the sign of the cross-product of the face tangent vector with a vector to the cell centroid can be used to determine if a triangle lies either to the interior or exterior of the boundary and then

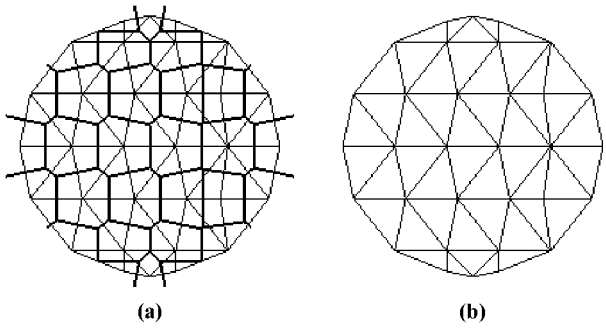


Figure 18.1.9 Bowyer algorithm for triangulating a circle. **(a)** Voronoi polygon. **(b)** Delaunay triangle.

can be easily removed (by defining the triangle connectivities) if it should lie outside the desired domain. Once the initial triangulation of the domain has been performed, all triangles that have a node associated with the initial user-defined superstructure are removed. Following this process, the Voronoi polygons and the final triangulation are shown in Figure 18.1.9.

In summary, the Watson and Bowyer algorithms are quite similar. Each algorithm starts with an initial grid surrounding the geometry to be discretized. New points are introduced one at a time, and triangles whose circumdisk contain the new point are deleted. The region is then re-triangularized by connecting points on the deleted triangles to the new point. The basic difference between the Watson and Bowyer algorithms, however, is in the initial superstructure and the data structures. Note that the Bowyer algorithm maintains essentially a list of only Voronoi polygons and can then form the triangle lists from the Voronoi diagram, whereas the Watson algorithm chooses simply to maintain a list of the triplets of node numbers which represent the completed triangles, in which a running list of circumcircle center and circumradius for each formed triangle is kept.

18.1.3 AUTOMATIC POINT GENERATION SCHEME

In both the Watson and Bowyer algorithms, "a new point is introduced." The method for producing the points, however, has not been addressed. An algorithm for automatic generation of points can be developed as follows [Weatherill, 1992]:

(1) Compute the point distribution function for each boundary point x_i, y_i:

$$dP_i = \frac{1}{2}\left[\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}\right]$$

where the points $i + 1$ and $i - 1$ are contiguous to i.

(2) Generate the Delaunay triangulation of the boundary points.

(3) For all triangles within the domain:
   (a) Define a prospective point to be at the centroid of the triangle.
   (b) Derive the point distribution, dP_m, for the prospective point by interpolating the point distribution from the nodes of the triangle.
   (c) Compute the distances, d_m $(m = 1,\ 2,\ 3)$ from the prospective node to each of the triangles. Then,

If $d_m < \alpha d P_m$, then reject the point and return to step 3a.
If $d_m > \alpha d P_m$, then insert the point using the Delaunay triangulation algorithm where the coefficient α is the parameter which controls the grid point density.

(d) Assign the interpolated value of the point distribution function to the new node.
(e) Move on to the next triangle.

18.2 ADVANCING FRONT METHODS

In contrast to the Delaunay-Voronoi methods (DVM), the advancing front methods (AFM) seek to achieve internal nodal formation and triangulation by marching techniques that advance front cell faces from the domain boundary, with or without background grid configurations. Various schemes of AFM have been reported [Lo, 1985, 1989; Peraire et al., 1987; Lohner, 1988] for both two dimensions (triangular elements) and three dimensions (tetrahedral elements). The AFM concept may be extended to a generation of quadrilateral elements [Zhu et al., 1991; Blacker and Stephenson, 1991]. We shall examine these and other topics in this section.

The simplest description of AFM begins with specification of boundaries, as shown in Figure 18.2.1 where the exterior boundaries move counterclockwise and interior boundaries (if they exist, i.e., multiply connected domain) move clockwise. For example, for the case of a simply connected domain (Figure 18.2.2a), exterior boundaries (nodes 1 through 6, Figure 18.2.2b) are used as initial active front faces. Node 7 is created to form a triangle 1-2-7 and then side 1-2 is deleted so that we now have two new front faces 1-7 and 2-7 (Figure 18.2.2c). Choose a new interior node 8 (Figure 18.2.2d) which will then allow side 2-3 to be deleted. The process continues (Figure 18.2.2e through Figure 18.2.2j) until all front faces are deleted. Deleted sides then represent the generated mesh.

The unstructured mesh generation by AFM described above may be controlled with node spacing more favorably maintained (node space control method). This method begins by constructing a coarse background grid of triangular elements which completely covers the domain of interest (Figure 18.2.3a). For the elements to be generated (Figure 18.2.3b), it is convenient to define a node spacing δ, the value of a stretching parameter s, and a direction of stretching α. Then the generated elements will have typical length $s\delta$ in the direction parallel to α and a typical length δ normal to α as shown in Figure 18.2.3b.

At each node on the background grid, nodal values of δ, s, α must be specified. During grid generation, local values will be obtained from interpolation of the nodal values on the background mesh. Note that if δ is required to be initially uniform and

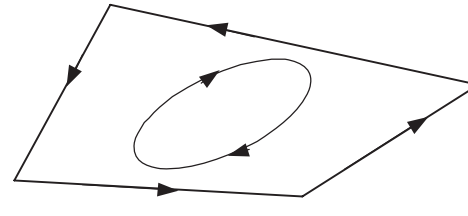
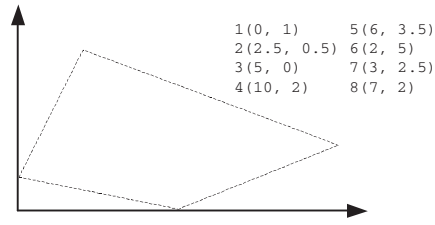
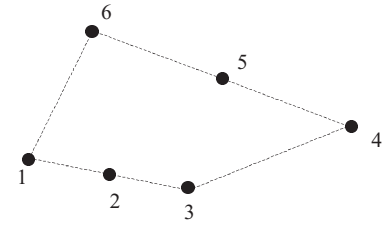


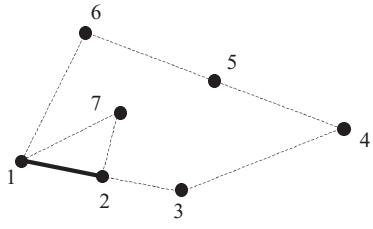
Figure 18.2.1 Multiply connected domain, counterclockwise advancing for outer boundaries, clockwise advancing for inner boundaries.



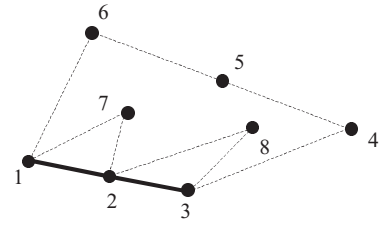
(a) Given domain



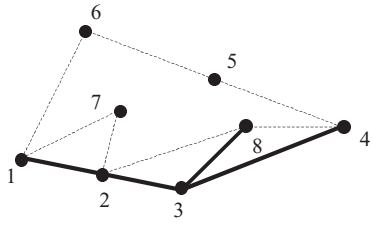
(b) Background boundary nodes (initial fronts)



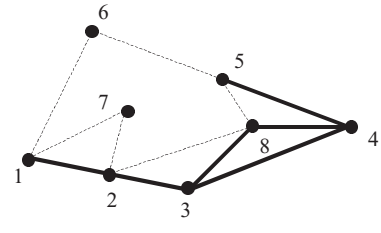
(c) Choose new node (7), delete side (1-2), new active fronts (1-7, 2-7)



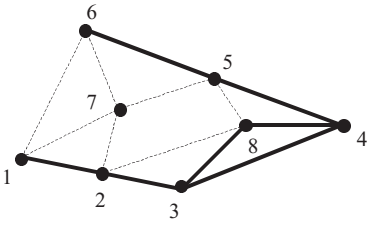
(d) Choose new node (8), delete side (2-3), new active fronts (2-8, 3-8)



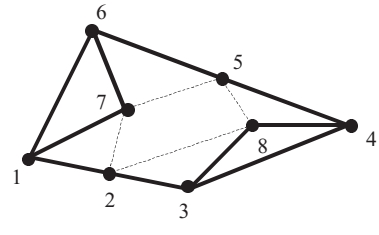
(e) Delete sides (3-8, 3-4), new active front (8-4)



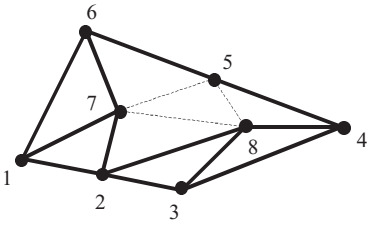
(f) Delete sides (4-8, 4-5), new active front (5-8)



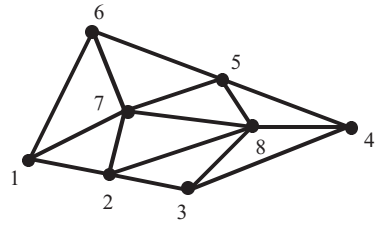
(g) Delete side (5-6), new active fronts (5-7, 6-7)



(h) Delete sides (6-7, 6-1, 1-7), no new active front



(i) Delete sides (2-7, 2-8), new active front (7-8)



(j) Delete sides (7-8, 7-5, 8-5), no new active front

Figure 18.2.2 General procedure for advancing front methods (AFM) for simply connected domain, boundaries of dash areas represent active advancing fronts. Deleted lines constitute the generated mesh.

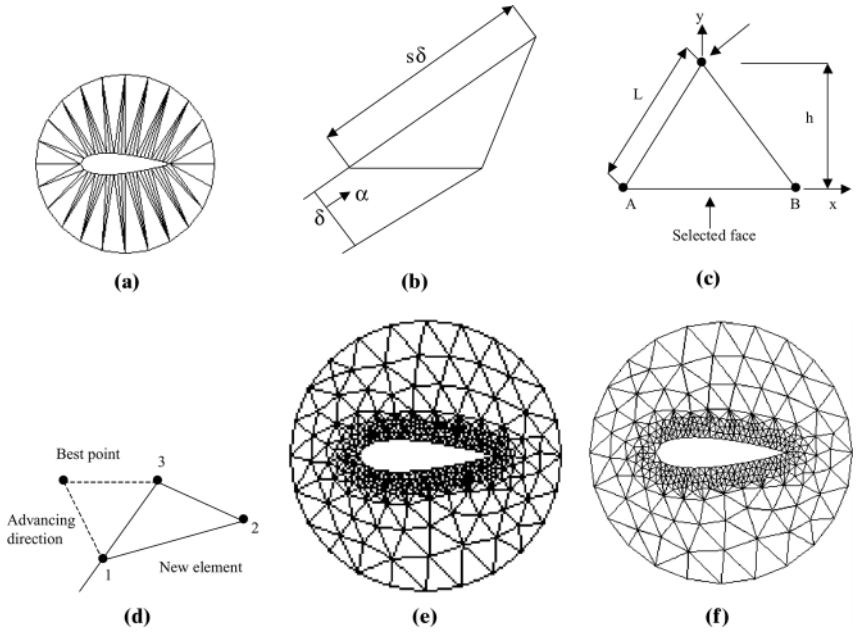


Figure 18.2.3 AFM procedure. **(a)** Background mesh. **(b)** Determination of mesh parameter. **(c)** Search for best point. **(d)** Undesirable element. **(e)** Finalized mesh. **(f)** Close-up view.

no stretching is to be specified, then the background grid need be only one triangle covering the entire domain.

Nodes are placed on the boundaries first, and the exterior boundary nodes are numbered counterclockwise, while any interior boundaries run clockwise. Thus, as the boundaries are traversed, the region to be triangulated always lies to the left.

At the start of the process, the front consists of the sequence of straight-line segments which connect consecutive boundary points. During the generation process, any straight-line segment that is available to form an element side is termed active, whereas any segment that is no longer active is removed from the front.

The following steps are involved in the process of generating new triangles in the mesh.

(1) Set up a background grid to define the spatial variation of the size, the stretching, and the stretching direction of the element to be generated (Figure 18.2.3b).
(2) Define the boundaries of the domain to be gridded, using the algebraic equations for each boundary.
(3) Using the information from Step 2, set up the initial front of faces. These faces are defined as segments between two consecutive points along the boundaries.
(4) Select the next face to be deleted from the front. In order to avoid large elements crossing over regions of small elements, the face forming the smallest new element is selected as the next face to be deleted from the list of faces.
(5) The following procedure is used for face deletion:
    (a) The "best point" is calculated as shown in Figure 18.2.3c (equilateral).
    (b) Determine whether a point exists in the already generated grid that should

be used in lieu of the new point. This step is accomplished by creating a list containing the node number of those nodes that fall within a circle centered at the "best point" and with a radius of $n\overline{AB}$ ($n = 3 \sim 5$). Also, the point must form a triangle with a positive area to be included in the list as shown in Figure 18.2.3d.

(c) Determine whether the element formed with the selected point does not cross any given faces. If it does, select a new point and try again.

(6) Add the new element, point, and faces to their respective lists.
(7) Find the generation parameters for the new faces from the background grid.
(8) Delete the known faces from the list of faces.
(9) If there is any face left in the front, go to step 4. The finalized mesh is shown in Figure 18.2.3e,f.

Note that the inclusion of stretching is achieved by using a local transformation that maps the real plane, in which stretching is desired, into a fictitious space, in which triangles satisfying the stretching conditions will appear to be equilateral. This transformation simply consists of a rotation of the axes to make α coincide with the x_1 axis, and a scaling by a factor s of the x_1 axis, and the inverse rotation to take the x_1 axis to the original position.

Recall that in the Delaunay-Voronoi methods, points are inserted in a previously determined manner, and then the entire mesh is re-triangulated. In contrast, the advancing front methods determine where to put the points directly from the space control scheme.

Mesh Smoothing

Practical implementations of either advancing front or Delaunay-Voronoi grid generators indicate that in certain regions of the mesh, abrupt variations in element shape or size may be present. These variations appear even when trying to generate perfectly uniform grids. The best way to circumvent this problem is to improve the uniformity of the mesh by smoothing.

The so-called Laplacian smoother or the "spring-analogy" smoother may be used. In this method, the sides of the element are assumed to represent springs. These springs are then relaxed in time using explicit time stepping, until an equilibrium of spring forces has been established [Spradley, 1999].

In each subdomain, the standard Laplacian smoother is employed. Each side of the element can be visualized to represent a spring. Thus, the force acting on each point is given by

$$f_i = c \sum_{j=1}^{ns_i} (x_j - x_i)$$

where c denotes the spring constant, x_i the coordinates or the point, and the sum extends over all the points, ns_i, surrounding the point i. The spring constant is set in the computation software, based on tests of the method.

The time advancement for the coordinates is accomplished as follows:

$$\Delta x_i = \Delta t \frac{1}{ns_i} f_i$$

At the boundary of the subdomain, the points are allowed to "slide" along the boundaries, but not to "leave" the boundary.

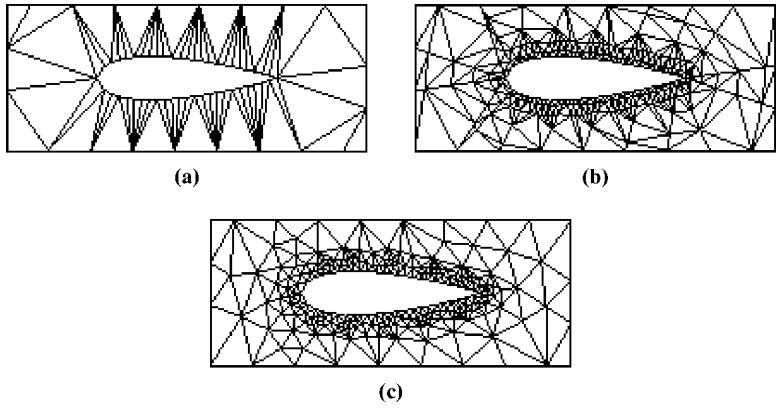


Figure 18.2.4 Mesh smoothing process, AFM. **(a)** Background mesh. **(b)** Finalized mesh without mesh smoothing. **(c)** After mesh smoothing.

The time step is also set in the code based on experience with using it. Usually, 5–10 time steps or passes over the mesh will smooth it sufficiently. The final results using the advancing front method without mesh smoothing and with mesh smoothing are shown in Figure 18.2.4. A sample program using C^{++} is listed in Figure 18.2.5.

```
//************************************************************************
// Module Name: Mesh Smoothing, Advancing Front Method
//************************************************************************
void Mesh_SmoothingMethod::meshSmoothing(int times)
                    // the parameter is the times of mesh smoothing, usually 10 is enough.
{
       int i, k;
       double deltaX, deltaY, deltaXY;
       int step[10]={10,9,8,7,6,5,4,3,2,1};

       numPoints=0;
       numTriangle=1;
       numEdge=0;

       readMeshFromFile(); // read triangle mesh from file
       formAllEdgeFromTriangleMesh(); // find all edges of triangle mesh
       findAllEdgeIndexForPoints(); // find point index for all edges

       for(k=0; k<times; k++)
       {
              calculateForceForPoints();  // calculate the force of points
                                          // f_i = c * sum_{j=1}^{ns_i} (x_j - x_i) , X and Y direction
                                          // Δx_i = Δt * (1/ns_i) * f_i
              for(i=0; i<numPoints; i++)
              {
                     if(pointSetData[i].type==SDC_INTERIOR)
                            // if the type of point is interior, then deform the position based
                            // on the force
                     {
                            deltaX=pointSetData[i].deltaX/step[k];
                            deltaY=pointSetData[i].deltaY/step[k];;
                            deltaXY=pointSetData[i].deltaXY/step[k];;
                            pointSetData[i].x+=deltaX;
                            pointSetData[i].y+=deltaY;
                     }
              }
       }
}
```

Figure 18.2.5 Mesh smoothing computer program using C^{++} [Z. Q. Hou, UAH].

Table 18.2.1 Comparison between Advancing Front and Delaunay-Voronoi Methods

	Advantages	Disadvantages
Advancing Front Methods	(1) A layer of well-positioned nodes allowing the front to advance. (2) Equilateral triangle with the frontal face and either stretch or compress it to match better the spacing requirements of the background mesh. (3) Refined process is straightforward, grids produced are quite regular. (4) High node distribution quality.	(1) Large amount of sorting and searching is needed to determine internal nodes. (2) Nodes generated may not be connected in an optimal way. (3) Generation process is cellwise, more costly than pointwise.
Delaunay-Voronoi Methods	(1) Each node is surrounded by its Voronoi region that compresses that part of the plane which is closer to this node than to any other node. (2) A unique triangulation is obtained by connecting the nodes whose Voronoi regions share a common boundary, forming a triangle with the three nodes that are closest to each other. (3) Generation process is pointwise, less costly than cellwise. (4) Optimal connectivity.	(1) Refined process is much more random; grids produced are not as regular. (2) Searching for the largest cell for a skewed cell with the largest circumcircle for each new node is very costly. (3) The skewness criterion is expensive (three square roots involved in the circle ratio).

The choice between DVM and AFM depends on the personal preference and the requirements for a given problem. This decision may be made upon the overall review of advantages and disadvantages presented in Table 18.2.1.

18.3 COMBINED DVM AND AFM

Having studied both DVM and AFM, it appears to be a logical approach to combine both methods in order to make use of advantages and discard disadvantages of both methods [Müller et al, 1993]. In this approach, we use a background grid to interpolate local mesh size parameters that is taken from the triangulation and create a set of nodes by means of AFM, and this set is subsequently introduced into the existing mesh, thus providing an updated DVM triangulation. The procedure is repeated until further improvements can not be obtained by inserting new nodes.

The results of a grid around a three element airfoil [Müller, Roer, and Deconinck, 1993] are shown in Figure 18.3.1. Here, the high node distribution quality of the AFM with the optimal connectivity of the DVM triangulation is demonstrated. Precise control of node spacing is achieved by the use of the initial triangulation of the boundary nodes as background mesh with no additional effort of the user. The node generation does not

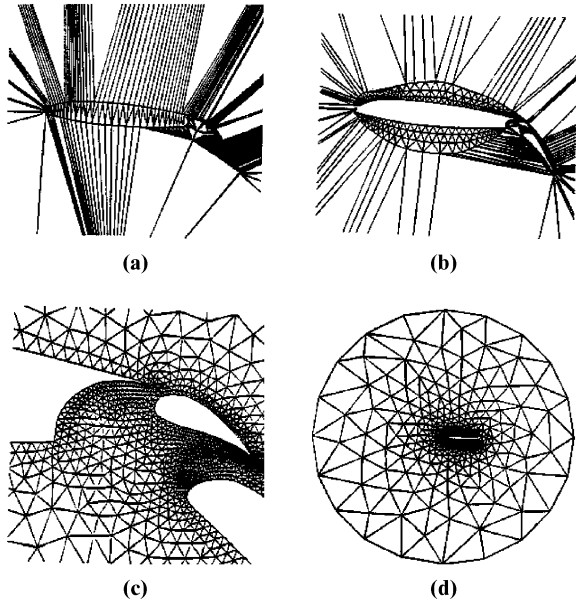


Figure 18.3.1 Three-element airfoil with combined DVM and AFM [Muller et al., 1993]. **(a)** Background grid. **(b)** After three rows of nodes inserted. **(c)** Details of grid around three-element airfoil. **(d)** Final grid of three-element airfoil.

require explicit tracking of the front and is independent of the order in which triangles are listed.

The resulting grids are very smooth and exhibit a high degree of regularity in cell shape and node distribution. This regularity is retained at singular points like corners or trailing edges. The use of a background grid that is derived from the initial triangulation of the boundary nodes results in a smooth variation in cell size of many orders of magnitude (about 10^5).

All features of this concept can be extended to three dimensions, where the optimal operation count and the simplicity of the front tracking and node construction of the method become even more attractive.

18.4 THREE-DIMENSIONAL APPLICATIONS

The basic concepts used in 2-D grid generations by means of DVM or AFM can be extended to 3-D geometries. Some of the earlier contributions include Baker [1989] for DVM and Löhner and Parikh [1988] for AFM. A brief review of these works is presented below.

18.4.1 DVM IN 3-D

Initially, the boundary surface grid generation using any one of the methods discussed in Section 18.3 is performed.

(1) The boundary points of the domain are created.

(2) Calculate the location of eight supplementary points in such a way that the hexahedron formed by these points contains all the points in the set.
(3) The mesh of this hexahedron using five tetrahedra (T_o) is created.
(4) Insert, one by one, the points of the set to obtain a mesh including these points as its element vertices. To this end, define:

$$T_n = \text{a triangulation including the first } n \text{ points of a set as vertices.}$$
$$x_{n+1} = \text{the next point in the set.}$$

According to step 3, point x_{n+1} is inside T_n, in which only three cases are possible:

(a) x_{n+1} is inside an element E_i of T_n.
(b) x_{n+1} is on the face common to two elements, E_j and E_k of T_n.
(c) x_{n+1} is on the edge common to several elements of E_i of T_n.

The fourth possibility corresponds to x_{n+1} being identical to one of the existing mesh points; but this is rejected as the points given are assumed to be distinct.

Using element(s) E_i, the set S of elements of T_n is created by a tree search, such that

(i) x_{n+1} is identical to the circumsphere associated with the elements of S. Triangulation of T_{n+1} is constructed in the same way as in (a) above:
(ii) Include the elements of T_n , not included in S, in T_{n+1}.
(iii) Remove the elements of S and remesh this set by joining point x_{n+1} to the external faces of S.

Once all points of the initial set have been introduced, the initial hexahedron is constructed from these tetrahedra.

18.4.2 AFM IN 3-D

The AFM in 3-D geometries follows basically the same procedure as in AFM for 2-D except that triangles are replaced by tetrahedra:

(1) Set up a background grid to define the spatial variation of the size, the stretching, and the stretching direction of the elements to be generated. The background grid consists of tetrahedra. At the nodes, we define the desired element size, element stretching, and stretching direction.
(2) Define the boundary surfaces of the domain to be gridded.
(3) Using the information stored on the background grid, set up faces on all these boundaries. This yields the initial front of triangular faces. At the same time, find the generation parameters (element size, element stretching, and stretching direction) for these faces from the background grid.
(4) Select the next face to be deleted from the front. In order to avoid large elements crossing over regions of small elements, the face forming the smallest new element is selected as the next face to be deleted from the list of faces.
(5) For the face to be deleted:
  (a) Select a "best point" position for the introduction of a new point.

(b) Determine whether a point exists in the already generated grid that should be used in lieu of the new point. If there is such a point, set this point as a new point and continue searching.
(c) Determine whether the element formed with the selected point does not cross any given faces. If it does, select a new point and try again.

(6) Add the new element, point, and faces to their respective lists.
(7) Find the generation parameters of the new faces from the background grid.
(8) Delete the known faces from the list of faces.
(9) If there are any faces left in the front, go to step 4.

18.4.3 CURVED SURFACE GRID GENERATION

Two approaches for surface grid generation may be considered: (1) Boolean operations on solids and (2) Boolean operations on surfaces.

(1) Boolean operations on solids – the domain to be gridded from primitives (box, sphere, cylinder, etc.). The user combines these primitives through Boolean operations (union, intersection, exclusion, etc.) to define the domain to be gridded. The surface is then obtained in a post-processing operation.
(2) Boolean operations on surfaces – here only the surface of the domain to be gridded is defined in terms of independent surface patches. The surface patches are then combined using Boolean operations to yield the final surface of the domain to be gridded.

Boolean operations include points, lines, and surfaces which are obtained similarly as in the case of structured grids of Chapter 17.

18.4.4 EXAMPLE PROBLEMS

Examples of unstructured AFM three-dimensional mesh generation using tetrahedral elements are demonstrated in Figures 18.4.1 through 18.4.3.

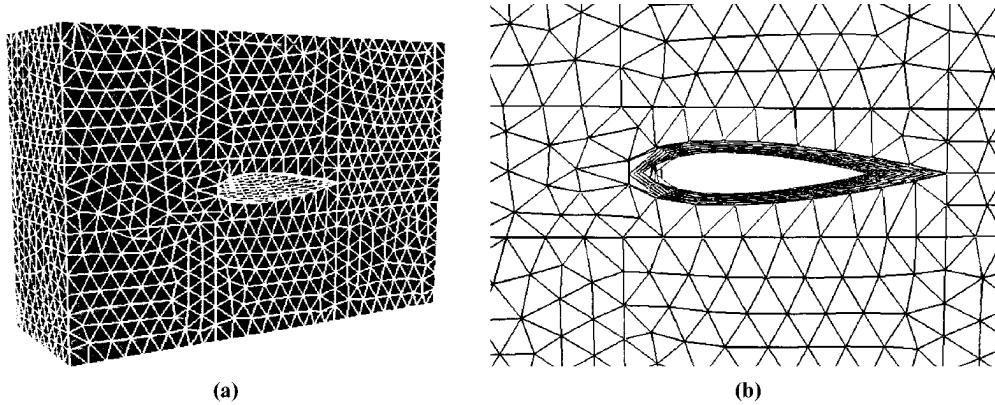
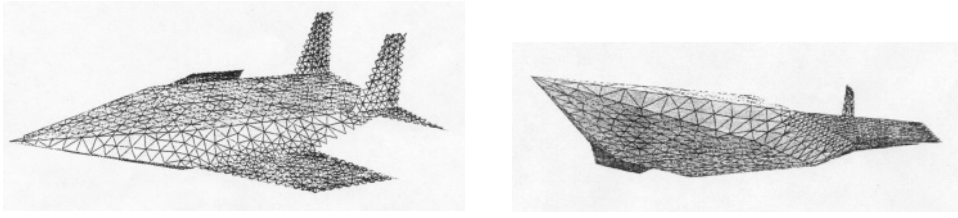


Figure 18.4.1 Unstructured tetrahedral AFM mesh for NACA0012 airfoil [Spradley, 1999]. **(a)** Surface mesh. **(b)** Close-up view.



(a) (b)

Figure 18.4.2 Unstructured AFM mesh for generic hypersonic plane [Spradley, 1999]. **(a)** Surface mesh. **(b)** Bottom view.

(1) NACA0012 Airfoil

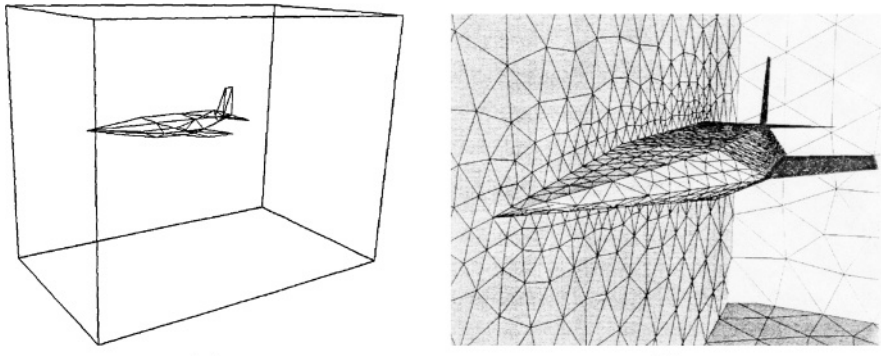
Figure 18.4.1a shows the mesh generation of NACA airfoil geometry [Spradley, 1999]. A tetrahedral mesh is generated with the element sizes and clustering kept simple for illustrating. The surface mesh shown in Figure 18.4.1a is convenient to view the unstructured mesh. The interior looks very much the same as the surface. A close-up view near the airfoil is shown in Figure 18.4.1b.

(2) Hypersonic Airplane

Figure 18.4.2 shows two views of the surface mesh of a hypersonic airplane with a relatively coarse mesh for illustration only [Spradley, 1999]. The three-dimensional CFD computational domain with tetrahedral mesh is demonstrated in Figure 18.4.3 [Spradley, 1999].

18.5 OTHER APPROACHES

There are many other approaches in generating unstructured grids. Among them are Zhu et al. [1991] for quadrilateral grids by means of modified AFM, Blacker and Stephenson [1991] also for quadrilateral grids through paving technique, Yerry and Shephard [1984] for 2-D and 3-D mesh generations using quadtree and octree methods, respectively.



(a) (b)

Figure 18.4.3 Unstructured AFM mesh for generic hypersonic plane with CFD domain [Spradley, 1999]. **(a)** CAD geometry for generic aircraft. **(b)** Unstructured AFM mesh.

In what follows, we shall briefly review these methods for the purpose of comparison. The reader should consult the original sources for details.

18.5.1 AFM MODIFIED FOR QUADRILATERALS

In this method, an even number of boundary nodes is first generated, according to the mesh size specification, on the boundary of the domain. The curvilinear boundary of the domain is then transformed into a polygon with an even number of sides. The main feature of the quadrilateral generation process is that first two triangles which have a common side are generated and then the triangles are combined to form a quadrilateral. This process continues until the domain is fully covered by nonoverlapping quadrilaterals.

The advantage of forming a quadrilateral by two triangles is that quadrilaterals with internal angles equal to and greater than 180° are allowed to be generated. These types of elements play an important role in the generation of transitions which are often necessary in a strongly graded mesh.

These steps can be readily achieved by any existing triangular mesh generator with some modifications; Zhu et al. [1991] utilized the AFM for this purpose. Note that the distinctive features of the AFM for triangular mesh generation are (1) that elements and nodes are generated simultaneously, and (2) that directional distribution of the elements can be achieved by introducing stretching in certain described directions so that the generated element size can be varied throughout the domain.

The following procedure is used for the quadrilateral element generation:

(1) Select an active side with nodes A and B in the generation front as a base to generate the first triangle (Figure 18.5.1a).
(2) Generate a triangle with nodes A, B, and C following the triangle generation process of AFM. The node D is determined according to the specified nodal spacing and it is either an active node in the current generation front or a newly constructed node in the region to be gridded.
(3) Set up the generation subfronts SF1 and SF2 (Figure 18.5.1b) by dividing the current generation front into two, if node D is an active node in the current generation front. Both SF1 and SF2 contain a closed loop of active sides and active nodes. No subfront is formed if D is a new node.

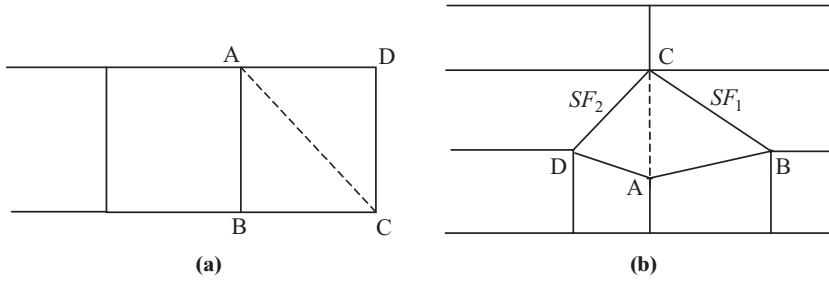


Figure 18.5.1 Generation of a quadrilateral element ABCD from AFM triangles. **(a)** No subfront is introduced after the generation of element ABCD. **(b)** Subfronts SF_1 and SF_2 are found.

(4) Examine the sides of triangle ABC. A side is flagged as active if it can be used as a side to form a new triangle. It is flagged as inactive otherwise.
(5) Select an active side which is also a side of triangle ABC as the base to generate the second triangle. Three possibilities may be encountered:
   (a) If no subfront has been formed in the generation of the first triangle ABC, in other words, node D is a new node being generated, then any side which is active in triangle ABC can be used to generate the second triangle.
   (b) If subfronts SF1 and SF2 are formed in the generation of triangle ABC, but one of the subfronts is empty, that is, all the sides in the subfront are flagged as inactive, then any active side in triangle ABC can again be chosen as the base for the second triangle generation.
   (c) Both subfronts SF1 and SF2 contain active sides. In this case one subfront, say SF1, contains an even number of active sides, the other subfront, say SF2, contains an odd number of active sides. The active side, which is in subfront SF2, in triangle ABC will be chosen as the base to generate the second triangle.
(6) Suppose the chosen active side for the second triangle has nodes A and C. Then the second triangle, say ACD, is generated by following the node and triangle generation process of AFM, with the requirement that each of the possible two new subfronts, formed as a result of the generation of the second triangle ACD, must contain an even number of active sides. Here steps (5) and (6) ensure that, after a quadrilateral is generated, every subfront formed in the element generation process contains a closed loop with an even number of active sides.
(7) The mesh parameters such as nodal spacing and element orientation for the new nodes are interpolated from the background mesh.
(8) Form quadrilateral element ABCD (Figure 18.5.1a,b). At this stage of the element generation, no consideration is given to the shape of the quadrilateral being generated. Indeed, there is no restriction on the shape of the quadrilaterals and any type of combination of two common side sharing triangles is allowed to form a quadrilateral. Such flexibility makes the generation of quadrilateral elements almost as easy as the generation of triangular elements. The enhancement of the element shape and therefore the enhancement of the quality of the mesh is an important part of this method. This can be achieved by means of (a) node elimination, (b) element elimination, (c) diagonal swapping, and (d) side elimination.
(9) Update the generation front (subfronts) so that the generation (sub)fronts(s) always form the boundary of the regions to be gridded. The sides that have been used to form the new element will be removed from the (sub)front(s), and the new sides will be included in the (sub)fronts(s).

The quadrilateral element generation proceeds sequentially or in parallel in each subfront. The generation procedure is complete only when every generation subfront is empty. The domain of interest is then covered entirely by quadrilateral elements.

In this method, no consideration has been given to the element orientation in the forming of the quadrilaterals. This is a subject of future research. Some typical results are shown in Figure 18.5.2 [Zhu et al., 1991].

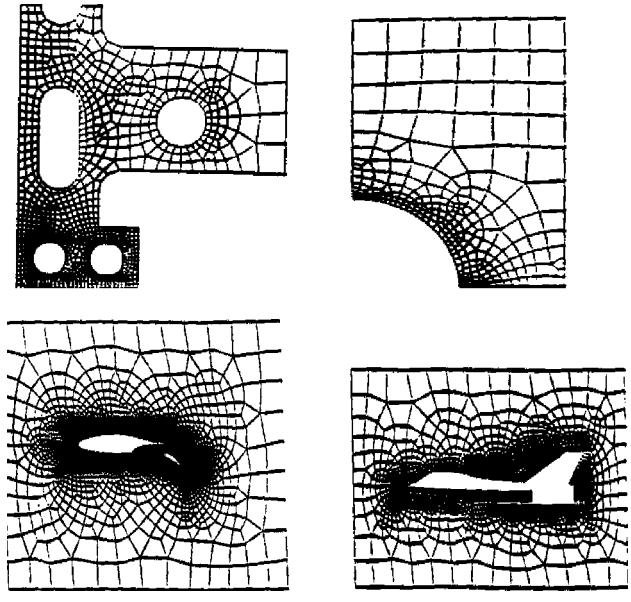


Figure 18.5.2 Examples of AFM modified for quadrilaterals [Zhu et al., 1991].

18.5.2 ITERATIVE PAVING METHOD

The paving technique meshes arbitrary 2-D geometries with quadrilaterals by iteratively layering or paving rows of elements to the interior of a region's boundary(ies) in a fashion similar to AFM. This technique was first proposed by Blacker and Stephenson [1991]. Paving allows varying element size distribution on the boundary as well as interior of a region. Similar to the AFM, the exterior boundary is ordered counterclockwise, and the interior boundary is ordered clockwise.

During mesh generation, the paving technique operates on boundaries of connected nodes referred to as paving boundaries. Initially, each paving boundary is identical to a permanent (fixed) boundary. As with permanent boundaries, paving boundaries are categorized as either exterior or interior boundaries, with exterior paving done counterclockwise and interior done clockwise.

The nodes are characterized into three types: (1) A fixed node is on a permanent boundary. (2) A floating node is any node not on a permanent boundary. (3) A paving node is any node on a paving boundary. Each paving node has an interior angle

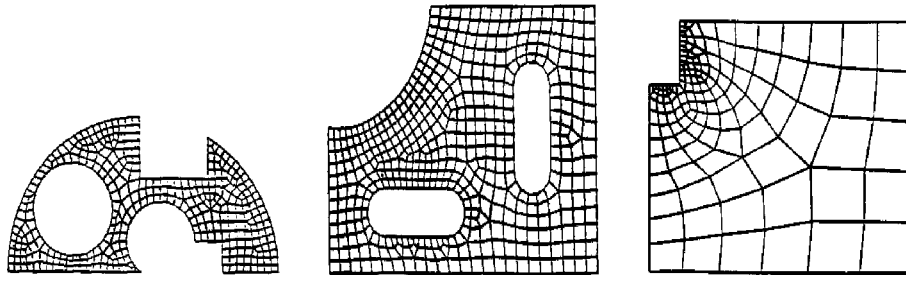


Figure 18.5.3 Examples of paving method [Blacker and Stephenson, 1991].

associated with it. This is the angle between a line connecting the node to the preceding node, and the line connecting it to the next node on the paving boundary. In order to generate an all quadrilateral mesh, each paving boundary must always contain an even number of nodes.

The propagation of the paving boundary involves the eight steps delineated below:

(1) Row choice – The beginning and ending of the next sequence or row of elements to be added is found.
(2) Closure check – A check is made to make sure that more than six nodes remain in the paving boundary. Specific closure techniques are used to conclude meshing for paving boundaries of six or fewer nodes.
(3) Row generation – The next row of elements identified in the row choice step is incrementally added to the boundary.
(4) Smooth – Floating nodes are adjusted to improve mesh quality and boundary smoothness.
(5) Seam – Small interior angles in the paving boundary are seamed or closed by connecting opposing elements.
(6) Row adjustment – The new row is adjusted by placing tucks or wedges into the row to correct for elements becoming too large or too small.
(7) Intersection – The paving boundary is checked for intersections with itself or with other paving boundaries. Intersections are connected to form new, often separate, paving boundaries.
(8) Cleanup – The completed mesh is adjusted where element deletion and/or addition improves the overall quality.

Some typical results using this technique are shown in Figure 18.5.3. Extension of the paving technique to 3-D geometries (hexahedral grid) can also be made.

18.5.3 QUADTREE AND OCTREE METHOD

Quadtree and octree methods were developed from the concept of superposition-deformation [Yerry and Shephard, 1984]. These methods construct a mesh of the domain under consideration, essentially from the data of points on its contour. A regular grid, or a grid based on a quadtree construction in three dimensions, is defined in such a way as to contain the domain. It is composed of squares and cubic boxes for quadtree and octree grids, respectively. This partitioning may then be deformed to resemble the real geometry. This process is shown in Figure 18.5.4 for a 2-D domain.

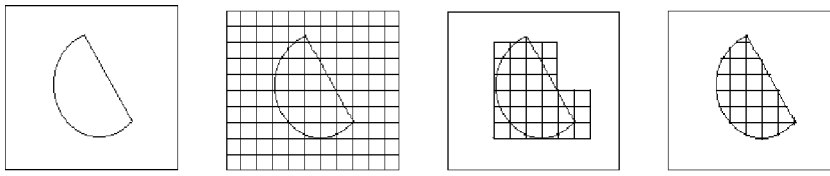


Figure 18.5.4 Quadtree method. **(a)** Initial domain. **(b)** Initial grid. **(c)** Removal of exterior grids. **(d)** Final mesh.

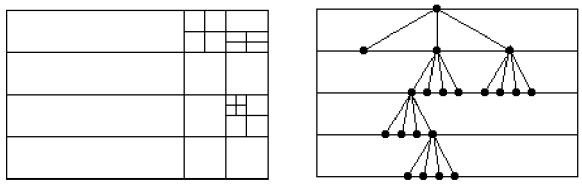


Figure 18.5.5 Quadtree data structure. (**a**) Recursive subdividing. (**b**) Parent-son-grandson relation.

Let us consider the quadtree shown in Figure 18.5.5a in which the mesh is refined in four levels. The parent is split into sons and grandsons as depicted in Figure 18.5.5b. For 3-D geometries, the method begins with a cube which is recursively split into sons and grandsons, that represents an octree structure. Thus, the final graded mesh can be constructed using this process.

18.6 SUMMARY

It was shown in this chapter that the two most popular methods of unstructured mesh generation are the Delaunay-Voronoi methods (DVM) and the advancing front methods (AFM). We examined the Watson and Bowyer algorithms for DVM, followed by AFM. Mesh smoothing in AFM was also discussed.

We presented the merits and demerits of DVM and AFM and showed that it is possible to combine both DVM and AFM taking advantage of the merits of both methods. We further examined three-dimensional applications for both DVM and AFM, followed by the surface grid generation. Other approaches such as AFM with quadrilaterals, iterative paving methods, and quadtree/octree methods were also examined. Unstructured grid generation is particularly useful in applications to adaptive methods. These and other subjects will be discussed in the next chapter.

REFERENCES

Baker, T. J. [1989]. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Eng. Comm*., 5, 161–75.

Blacker, T. D. [1991]. Analysis automation with paving: a new quadrilateral meshing technique. *Adv. Eng. Soft*., 13: 5/6.

Blacker, T. D. and Stephenson, M. B. [1991]. Paving: a new approach to automated quadrilateral mesh generation. *Int. J. Num. Meth. Eng*., 32, 811–47.

Bowyer, A. [1981]. Computing Dirichlet tessellations. *Comp. J*., 24, 2, 162–66.

Cavendish, J. C., D. A., Field, and W. H. Frey [1985]. An approach to automatic three-dimensional finite element mesh generation. *Int. J. Num. Meth. Eng*., 21, 329–47.

Delaunay, B. [1934]. Sur la sphere vide. Izvestiva Akademii Navk SSSR, *Math. Nat. Sci. Div*., 6, 793–800.

Dirichlet, G. L. [1850]. Uber die Reduction der positiven Quadratschen Formen mit drei undestimmten ganzen Zahlen. *Z. Reine Angew. Math*., 40, 3, 209–27.

Lo, S. H. [1985]. A new mesh generation scheme for arbitrary planar domains. *Int. J. Num. Meth. Eng*., 21, 1403–26.

———. [1989]. Delaunay triangulation of non-convex planar domains. *Int. J. Num. Meth. Eng*., 28, 2695–2707.

Löhner, R. [1988]. Some useful data structures for the generation of unstructured grids. *Comm. Appl. Num. Meth*., 4, 123–35.

Löhner, R. and Parikh, P. [1988]. Three-dimensional grid generation by the advancing front method. *Int. J. Num. Meth. Eng*., 8, 1135–49.

Müller, J. D., Roe, P. L., and Deconinck, H. [1993]. A frontal approach for internal node generation in Delaunay triangulations. *Int. J. Num. Meth. Fl.*, 17, 241–55.

Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O. C. [1987]. Adaptive remeshing for compressible flow computations. *J. Comp. Phys*., 72, 2, 449–66.

Spradley, L. W. [1999]. A generalized meshing environment for adaptive computational fluid dynamics. Ph.D. dissertation, The University of Alabama, Huntsville.

Voronoi, G. [1908]. Nouvelles applications des parametres contilnus a la theorie des formes quadratiques. Rescherches sur les paralelloedres primitifs. *J. Reine angew, Math*. 134.

Watson, D. F. [1981]. Computing the N-dimensional Delaunay tesselation with application to Voronoi polytopes. *Comp. J*., 24, 2, 167–72.

Weatherill, N. P. [1992]. Delaunay triangulation in computational fluid dynamics. *Comp. Math. Appl.*, 24, 5, 129–50.

Yerry, M. A. and Shephard, M. S. [1984]. Automatic 3D mesh generation by the modified-octree technique. *Int. J. Num. Meth. Eng*., 20, 1965–90.

Zhu, J. Z., Zienkiewicz, O. C., Hinton, E., and Wu, J. [1991]. A new approach to the development of automatic quadrilateral mesh generation, *Int. J. Num. Meth. Eng*., 32, 849–66.