JOURNAL OF COMPUTATIONAL PHYSICS **106**, 125–138 (1993)

Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer–Watson Algorithm

S. Rebay

Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano, Via Golgi 40, 20133 Milano, Italy

Received July 22, 1991

This work is devoted to the description of an efficient unstructured mesh generation method entirely based on the Delaunay triangulation. The distinctive characteristic of the proposed method is that point positions and connections are computed simultaneously. This result is achieved by taking advantage of the sequential way in which the Bowyer–Watson algorithm computes the Delaunay triangulation. Two methods are proposed which have great geometrical flexibility, in that they allow us to treat domains of arbitrary shape and topology and to generate arbitrarily nonuniform meshes. The methods are computationally efficient and are applicable both in two and three dimensions. © 1993 Academic Press, Inc.

1. INTRODUCTION

When looking for the numerical solution of fluid dynamic equations defined in the continuum such as the full potential, Euler, or Navier–Stokes equations, it is necessary to select a finite number of points in the domain and to connect them so as to define a grid, on which a discrete version of the original partial differential problem is constructed.

There are two main classes of grids, which differ in the way in which the mesh points are connected to each other. If the internal points are connected to their neighbours in a way independent of their position, the mesh is called structured. When the pattern of the connections varies from point to point, the mesh is called unstructured. In the structured case, the connectivity of the grid is implicitly taken into account by storing the point data into the elements of a matrix. On the contrary, the connectivity of unstructured grids must be explicitly described by an appropriate data structure, thus making the solution algorithms on unstructured grids more expensive than those on structured grids. However, the greater geometrical flexibility offered by unstructured grids can be crucial when dealing with domains of complex geometry or when the mesh has to be adapted to complicated features of the flow field. The increasing attention that such problems have received in recent years has therefore made unstructured grids of common use in computational fluid dynamics. A great effort has consequently been directed toward the development of unstructured mesh generation methods. Among the available techniques, those based on the Delaunay triangulation are particularly suited to an adaptive solution strategy since the Delaunay construction allows the addition of new points to an existing triangulation without the need for remeshing the whole domain.

In earlier applications of Delaunay triangulation to unstructured grid generation, the Delaunay construction has been used to connect a preassigned distribution of points, so as to obtain a triangulation satisfying certain geometrical properties. In the most common mesh generation methods of this class, in fact, the spatial distribution of the mesh points is determined by means of some appropriate technique in a step preceding that of establishing their connections (see, e.g., Baker [1], Jameson, Baker, and Weatherill [4], and Weatherill [10]).

In this work we describe a completely different approach, which takes full advantage of the sequential way in which the Delaunay triangulation is constructed. The distinctive characteristic of the new method is that point positions and connections are computed simultaneously. Two different mesh generation algorithms will be considered in the present work, the only difference between them being the way in which points are positioned in the domain. Both algorithms are of the greatest logical simplicity allowing the treatment of domains of arbitrary shape and a complete control of the size of the triangles over the computational domain. Moreover, the algorithms are computationally very efficient and are applicable to two- and three-dimensional problems.

The structure of the paper is organized as follows. In Section 2 we recall the well-known Delaunay triangulation and the associated Dirichlet tessellation. Section 3 describes how Bowyer–Watson algorithm, which is widely employed to construct the Delaunay triangulation, can be conveniently used in the context of mesh generation. In Section 4 we introduce an initial triangulation of the boundary points which represents the starting point of the two mesh generation methods to be described in detail in Sections 5 and 6, respectively. In Section 7 the potentialities offered by the proposed algorithms are investigated by applying them to some representative test cases. Section 8 is devoted to the conclusions.

2. DIRICHLET TESSELLATION AND DELAUNAY TRIANGULATION

A systematic way for generating a triangulation of a set of points can be obtained by considering a geometrical construction first introduced by Dirichlet in 1850. Consider an arbitrary set of points P_i, $i = 1, ..., N$. For any point P_i we define a convex polygon $\mathcal{V}_i$ characterized by the property that every point of $\mathcal{V}_i$ is nearer to P_i than to any other P_j. These convex polygons are called *Voronoi regions* and their union is called *Dirichlet tessellation.*

The boundary between two Voronoi regions $\mathcal{V}_i$ and $\mathcal{V}_j$ facing each other lies midway points P_i and P_j and is therefore a segment of the axis of the line which joins P_i with P_j. By connecting only those points whose Voronoi regions have a common edge, a triangulation known as *Delaunay triangulation* is obtained. An example of a Dirichlet tessellation with the associated Delaunay triangulation of the convex hull of a small set of points is shown in Fig. 1.

This geometrical construction can be easily extended to more dimensions. In the three-dimensional space, for example, the boundary between two facing Voronai regions is a plane polygon, and, by connecting only the points whose Voronai polyhedra have a polygon in common, a set of tetrahedra is obtained.

It should be noted that Delaunay triangulations and Dirichlet tessellations can be considered the geometrical dual of each other, in the sense that for every triangle $\mathcal{T}_i$ there exists a vertex C_i of the tessellation, and conversely, for every polygon $\mathcal{V}_j$ there exists a vertex P_j of the triangulation (see Fig. 1). In addition, for every edge of the triangulation there exists a corresponding *segment* of the Dirichlet tessellation. In the following, vertices and segments of the Dirichlet tessellation will be also called Voronoi vertices and Voronoi segments, respectively.

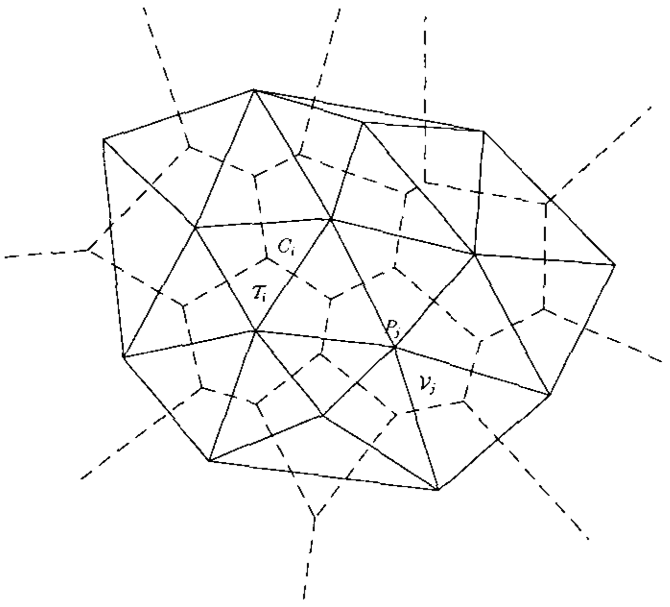

FIG. 1. Dirichlet tessellation (dashed) and Delaunay triangulation (continuous) of the convex hull of a small set of points.

3. BOWYER–WATSON ALGORITHM AND MESH GENERATION

The Delaunay triangulation recalled in the preceding section is a construction characterized by certain specific geometrical properties. Some of these properties are of a local nature and allow us to devise algorithms which compute the Delaunay triangulation of an arbitrary set of points only by means of a sequence of purely local operations.

Among the methods belonging to this class, the Bowyer [2] and/or Watson [9] algorithm is very convenient in a mesh generation procedure. The method is based on the so-called *circumcircle property* which guarantees that no point of a Delaunay triangulation can lie within the circle circumscribed to any triangle. The Bowyer–Watson algorithm is essentially a "reconnection" method, since it computes how an existing Delaunay triangulation is to be modified because of the insertion of a new point. A simple example showing the sequence of operations needed to insert a new point into an existing triangulation and to generate a new triangulation is given in Fig. 2. The algorithm removes from the existing grid all the triangles which violate the circumcircle property because of the insertion of the new point. It can be shown that (1) all these triangles are always contiguous, thus forming a connected cavity surrounding the newly inserted point, and that (2) by joining the vertices of this cavity with the internal new point, a Delaunay triangulation is always obtained [2, 9]. As a consequence, the Delaunay triangulation of an arbitrary set of points can be constructed in a purely sequential manner starting from a

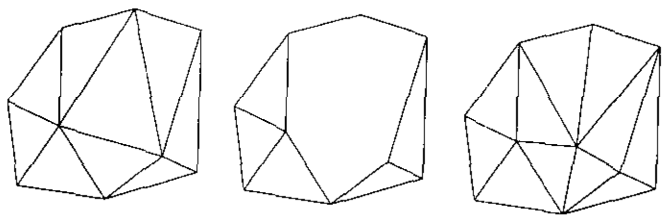
FIG. 2. Reconnection of an existing grid around a newly inserted point by means of the Bowyer–Watson algorithm.

very simple initial Delaunay triangulation enclosing all points to be triangulated (for example, that formed by two adjacent triangles) and adding one point after another until all points have been considered.

The algorithm efficiency depends on how quickly the search for the triangles to be deleted at each point insertion is performed and this is made much easier by the knowledge of the neighbouring triangles to each triangle. In fact, since all the triangles to be deleted are always contiguous, a tree search among neighbouring triangles can be used to find all the other triangles to be deleted after the first one. In the typical case, the number of triangles to be deleted at each point insertion does not depend on the number of all existing triangles. As a consequence, if the information pertaining to the neighbouring triangles is available and an $O(\log N)$ multidimensional search for the *first* triangle to be deleted is employed, the algorithm can compute the Delaunay triangulation of a set of N points in $O(N \log N)$ operations. In special cases, however, the number of triangles to be deleted at each point insertion can be very large. In the worst possible situation, when all existing triangles have to be deleted at each point insertion, the operation count of the Bowyer–Watson algorithm degrades to $O(N^2)$. It has, however, been shown that a proper ordering of the input points can cure this problem.

Unstructured mesh generation techniques which use the Bowyer–Watson algorithm to compute the connections of a preassigned distribution of points have been successfully employed for both two- and three-dimensional problems [1, 4, 10].

The sequential nature of the Bowyer–Watson algorithm can be further exploited to compute mesh point positions and connections simultaneously if, starting from an available triangulation with N points, the position of the $(N+1)$th point is chosen according to some suitable geometrical criterion which depends on the existing triangulation. Within such an approach, it is very important to choose carefully the position where to insert the new point in the existing mesh, to avoid a nonsmooth point distribution. Even if the Delaunay construction guarantees in advance that the points will be connected so as to produce triangles as equilateral as possible, a poor point distribution can eventually lead to an unsatisfactory triangulation. In this respect, the vertices and the segments of the Dirichlet tessellation are promising locations for placing a new point since they represent a geometrical locus which falls, by construction, midway between the triangulation points. In both cases, the new point is inserted in a position which avoids the formation of edges that are too short so that it is unlikely for the new triangles to have a very elongated shape. Two different mesh generation methods in which the new point is inserted at a Voronoi vertex or on a segment of the Dirichlet tessellation will be considered in Sections 5 and 6, respectively.

In the first method, which follows closely an idea originally introduced in the pioneering work of Holmes and Snyder [5], the new point is introduced in the existing triangulation at the Voronoi vertex (centre of the circumscribed circle) corresponding to the "worst" triangle, defined as the one having the most different circumscribed circle radius in relation to that required for the final grid. In this way, since the worst triangle is obviously eliminated from the triangulation, we improve the quality of the grid at every new point insertion. The mesh generation process is terminated when all "bad" triangles have been eliminated, leaving a grid formed only by suitable triangles.

In the second method here described, the new point is instead inserted in an attempt to generate one or possibly several new triangles having from the very beginning the size prescribed for the final grid. This result is accomplished by placing the new point along a segment of the Dirichlet tessellation, in a position carefully chosen to guarantee that the newly generated triangles have the sought for size. As will be shown in the examples reported in Section 7, it turns out that the grids generated by means of the second method are characterized by triangles of a generally better shape in comparison with those obtained by means of the first method. This is especially true for triangles located near the boundaries of the domain, a region where a regular spatial discretization is of crucial importance to perform accurate computations.

Both mesh generation algorithms offer complete control of the triangle sizes and allow an implementation with operation count $O(N \log N)$. This result is accomplished by means of a proper ordering of the triangle data. Sophisticated multidimensional efficient search routines are therefore not required by the proposed methods.

4. INITIAL TRIANGULATION OF THE BOUNDARY POINTS

Both mesh generation methods described in the paper choose the position of the $(N+1)$th mesh point on the basis of the grid connecting the already existing N points. It is therefore necessary to provide an initial grid to start the mesh generation process. To this aim, all the boundary points are triangulated by means of the Bowyer–Watson algorithm.

The initial *boundary triangulation* must be body conforming; i.e., all boundary edges must be included in the triangulation. Since the Delaunay triangulation of a given set of points is a unique construction, there is no guarantee that, for an arbitrary distribution of boundary points, the resulting triangulation will be boundary conforming. However, by performing a check on the initial triangulation, all possible missing boundary edges can be detected. It can be proved that, by repeated insertions of new mesh points at the midpoint of the missing boundary edges, a boundary conforming triangulation is always obtained; see, for instance, [11]. This initial check is a relatively cheap procedure, since the number of boundary points in the mesh is generally small compared to the total number of mesh points to be generated—asymptotically $O(\sqrt{N})$ in two dimensions.

Two different types of triangles can be identified at this stage—the triangles *internal* to the domain to be meshed and the triangles *external* to it. It is important to keep track of both *internal* and *external* triangles. Internal triangles must be considered in view of the fact that the new mesh points must be generated inside the domain. Moreover, to prevent the elimination of boundary edges during mesh generation, it is sufficient to check that no external triangle is ever deleted from the triangulation. In this way, the boundary conforming property possessed by the initial triangulation remains invariant during the entire mesh generation procedure.

Different strategies can be adopted to deal with a point that would cause an external triangle to be deleted. For example, an attempt could be made to find a different location for the new point. Alternatively, the boundary could be locally refined. In this work, a very simple strategy turned out to be effective—points which would cause an external triangle to be deleted are simply discarded. In principle, this procedure could lead to a grid which locally does not satisfy the length scale criterion. Numerical investigation seems, however, to indicate that this simple boundary treatment, when used in conjunction with the point generation methods described in the following sections, leads in practice to a fairly regular point distribution near to the boundaries.

In the following sections we will describe in detail the two aforementioned strategies for defining the position of the new mesh point inside the domain on the basis of an existing set of internal triangles.

5. VORONOI-VERTEX POINT INSERTION METHOD

As outlined in Section 3, a promising location to place a new point is the centre of the circle circumscribed to a triangle. With this choice, the new point is in fact inserted in a position where, by the circumcircle property, the grid is coarsest.

The choice of the "insertion triangle," namely the triangle where to insert the new point, can be done according to different criteria. For example, in the grid generation method of Holmes and Snyder, the new points are introduced to eliminate in turn triangles with large areas and triangles having bad "aspect ratios," the latter being defined as the ratio between of the radii the inscribed and circumscribed circles. In this way, all triangles that are large or far from equilateral are eliminated from the triangulation, which therefore tends to cause clustering according to the geometrical discretization of the boundary points. A disadvantage of the method is that the coarseness of the triangulation over the entire domain is controlled only by the graduation of the boundary points, so that the dimensions of the internal triangles far from the boundaries may turn out to be inappropriate for the considered problem.

A simple insertion criterion is suggested here which allows control of the grid coarseness over the entire domain. We begin by defining a function $f(\mathbf{x})$ which prescribes the value of a characteristic dimension of the triangles, say the radius of the circumscribed circle, as a function of the triangle position. We define for each triangle $\mathcal{T}_k$ with circumscribed circle radius ρ_k the ratio

$$\alpha_k = \frac{\rho_k}{f(\mathbf{x}_k)},$$

where $\mathbf{x}_k$ indicates the position of the centre of the circle circumscribed to the triangle $\mathcal{T}_k$. By inserting the new mesh point at the centre of the circumscribed circle having the largest value of α, it is possible to reach eventually a mesh in which $\max_k \alpha_k < 1$, which is exactly what we are looking for, assuming that $f(\mathbf{x})$ represents the value prescribed for the radius of the circumcircle with centre at $\mathbf{x}$.

The actual expression of the size function $f(\mathbf{x})$ can be given in a variety of ways. For instance, $f(\mathbf{x})$ can be constructed by considering a piecewise linear function obtained by interpolating prescribed nodal values over a convenient *background mesh*. In this way, very general size functions can be obtained, the only input data required being the backpoint coordinates and the corresponding size function values. The background mesh can in fact be constructed easily by means of the Bowyer–Watson algorithm, given the backnode coordinates, and, provided that a triangulation of the backnodes is available, the size function $f(\mathbf{x})$ can be computed by means of standard triangular linear finite elements.

Beside allowing complete control of the triangle size over the domain, the proposed method also bypasses the efficiency bottleneck of the Bowyer–Watson algorithm, caused by the need to find the first triangle to be deleted at every point insertion. In fact, this triangle can be chosen to be the one with the largest value of α. Then, an efficient method is obtained by regarding all the internal triangles as a heap list, ordered according to the value of α, and by deleting the triangle at the top of the list. All other triangles to be deleted at each point insertion can be found by means of an inexpensive $O(1)$ tree search looking at neighbouring triangles. Since any removal or insertion of new elements in the list can be achieved in $O(\log N)$ operations (see, e.g., [3]), we obtain an overall $O(N \log N)$ method. The first mesh generation method can be summarized therefore as follows:

1. Triangulate all the boundary points by means of the Bowyer–Watson algorithm.
2. Check if the triangulation is body conforming and recover all missing boundary edges.
3. Divide all triangles into internal or external with respect to the computational domain.
4. Define a non-dimensional ratio α and evaluate its value α_k for each internal triangle $\mathcal{T}_k$.
5. Order the internal triangles according to α_k.
6. Insert a new point at the centre of the circumcircle of the triangle at the top of the ordered list.
7. Regenerate the triangulation by means of the Bowyer–Watson algorithm, possibly discarding those points which would cause the deletion of an external triangle.
8. Insert the newly generated triangles into the list of the internal triangles.
9. If $\max_k \alpha_k > 1$ return to step 6, else stop.

6. VORONOI-SEGMENT POINT INSERTION METHOD

In this section, the second mesh generation method relying on the Bowyer–Watson algorithm is described. The basic difference between this method and that presented in the previous section is that the new mesh point is taken on a segment of the Dirichlet tessellation, instead of at a Voronoi vertex. Furthermore and more important, instead of eliminating the worst triangle as in the first method, the new point is here inserted in an attempt to generate immediately one or possibly more new triangles having the required size from the very beginning. It turns out that the quality of the meshes obtained by means of the second method appears to be generally superior to that of the meshes provided by the previous one.

6.1. *Point Insertion Strategy*

The starting point of the mesh generation is an initial boundary triangulation obtained by connecting all boundary points as described in Section 4. Since the new point is introduced on a segment of the Dirichlet tessellation and there is a one-to-one correspondence between the segments of the tessellation and the edges of the triangulation, the position of the new point will be established in a way which depends on suitable couples of neighbouring triangles and their common edge. To this aim, it is convenient to subdivide all triangles present in the initial boundary triangulation into several types as explained in the following.

In the first place, it is necessary to distinguish between triangles *external* and triangles *internal* to the domain, in the same way as described in Section 4. Then, the internal triangles must be further subdivided into two types—triangles already having and triangles not yet having the size requested by their position in the domain. These two types of internal triangles will be referred to as *accepted* and *nonaccepted* triangles, respectively. In most common cases, the internal triangles of the intial boundary triangulation are all nonaccepted triangles. Since points are inserted by considering nonaccepted triangles with a neighbour already having the required size, it is necessary to introduce a further subdivision of the nonaccepted triangles—namely, *active* and *waiting* triangles. An *active* triangle is defined as a nonaccepted triangle having at least one *accepted* or one external triangle among its neighbours. All the nonaccepted triangles not active are considered *waiting*. A waiting triangle is therefore a triangle surrounded only by active or waiting triangles. For the sake of clarity, the hierarchical structure describing all the types of triangles involved in the mesh generation process is depicted in Fig. 3.

The new point is always inserted by considering the Voronoi segment associated with a triangulation edge shared by the active triangle with largest circumscribed circle radius and one of its accepted and/or external neighbours. In this way, by virtue of the circumcircle property of the Delaunay triangulation, new points are automatically positioned in regions of the domain, where the mesh is coarsest. If more than one accepted and/or external neighbour exists, the segment associated with the shortest triangulation edge is chosen. This choice is purely empirical, and other possibilities might give good results as well. All test cases reported in Section 7 have, however, been obtained according to the above criterion.

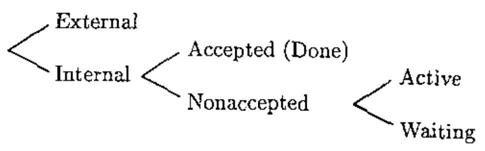

FIG. 3. Hierarchical structure of triangle types.

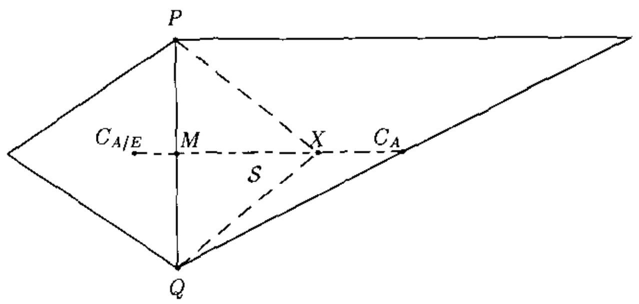

FIG. 4. Accepted or external triangle (left), active triangle (right), and the new point X.

6.2. *Positioning of the New Point*

The position of the new point along the Voronoi segment is chosen in an attempt to generate a new triangle having the size prescribed at its location. With reference to Fig. 4, let $\mathcal{S}$ be the segment of the Dirichlet tessellation associated with the triangulation edge PQ in common with the two aforementioned neighbouring triangles, the left being the accepted or external triangle and the right being the active one. The new point X is placed on $\mathcal{S}$ on the side of the active triangle in a position chosen so that the triangle formed by connecting X with P and Q has the size prescribed for the final grid. The position of X is computed as follows. Let C_A be the centre of the circle circumscribed to the active triangle (the right endpoint of segment $\mathcal{S}$ in Fig. 4), and let M be the point in common to edge PQ and segment $\mathcal{S}$ (midpoint of PQ). Furthermore, let

$$\rho_M = f(\mathbf{x}_M)$$

be the value of the circumscribed radius prescribed for the final triangulation at point $\mathbf{x}_M$ (obtained, as usual, by means of a background grid). Ideally, we would like to locate the new point X on segment $\mathcal{S}$ at the intersection of $\mathcal{S}$ with the circle passing through points P and Q and having circumscribed radius equal to ρ_M. However, it might happen that the prescribed value ρ_M is not appropriate, since any circle through P and Q has a radius $\rho \geqslant \frac{1}{2}|PQ|$. Furthermore, a real intersection point X exists only for circles having a radius smaller than that of the circle passing through P, Q, and C_A; i.e.,

$$\rho \leqslant \frac{p^2+q^2}{2q},$$

where $p = \frac{1}{2}|PQ|$ and $q = |C_A M|$ (see Fig. 4). For these reasons, a limited value of ρ_M is defined as

$$\hat{\rho}_M = \min\left[\max(\rho_M\, p), \frac{p^2+q^2}{2q}\right].$$

The position of the new point X is therefore defined by the relationship

$$\mathbf{x}_X = \mathbf{x}_M + d\mathbf{e},$$

where the distance $d = |XM|$ of X from M and the unit vector $\mathbf{e}$ are

$$d = \hat{\rho}_M + \sqrt{\hat{\rho}_M^2 - p^2},$$

$$\mathbf{e} = \frac{\mathbf{x}_{C_A} - \mathbf{x}_{C_{A/E}}}{|\mathbf{x}_{C_A} - \mathbf{x}_{C_{A/E}}|}.$$

Now, it is important to note that it is not assured that, by inserting X as explained above and then retriangulating, the triangle XPQ will be actually produced by the Bowyer–Watson algorithm. However, point X is positioned by construction inside the circle circumscribed to the active triangle, which is therefore always deleted from the triangulation. As a consequence, we are guaranteed that the new triangulation will contain two edges PX and QX, having by construction a length consistent with that required for the final triangulation. As a matter of fact, it is inessential whether the triangle XPQ is generated or not, since we are guaranteed that a Delaunay triangulation always connects points so as to produce triangles as equilateral as possible.

As a final remark, note also that, when $\mathcal{S}$ is too short with respect to $|PQ|$, the new point will be inserted at the Voronoi vertex of the active triangle. In this special situation, the point location is therefore coincident with that generated by the method described in Section 5.

6.3. *Assignment of the New Triangles*

As anticipated in Section 3, at each point insertion the Bowyer–Watson algorithm generates a new triangulation which differs from the previous one only locally around the newly inserted point. For this reason, and by virtue of the definitions employed to classify the triangles, a new grid in which *all* triangles are classified consistently with the definitions given in Section 6.1 can be constructed by considering *only* (1) triangles having the new point as one of their vertices (new triangles) and (2) triangles adjacent to the new ones (see Fig. 5). At each point insertion, these triangles

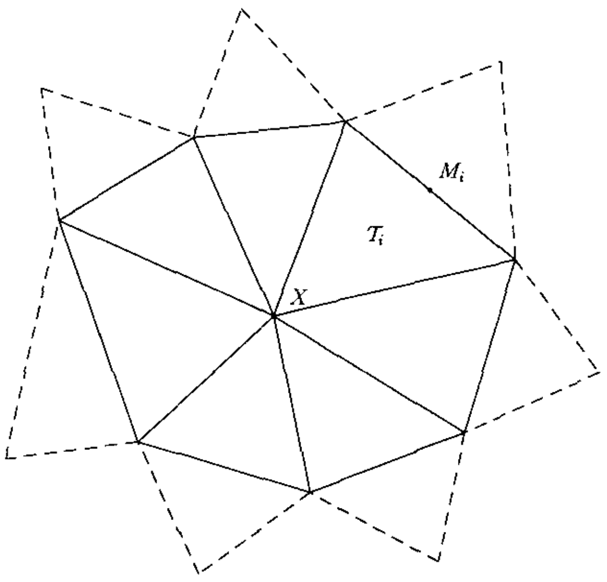

FIG. 5. The new triangles generated by the Bowyer–Watson algorithm because of insertion of point X (continuous) and their neighbours (dashed).

have to be reassigned to the different types described above according to their size and relative position. As a consequence, new accepted triangles will replace old nonaccepted ones starting from the boundary and moving to the interior of the domain, in a fashion similar to that characterizing the triangle propagation occurring in methods in which new points are generated along a propagation front, such as that originally proposed in [6] for two-dimensional applications and subsequently extended to three dimensions in [7].

There is no need to subdivide the new triangles into external and internal, since, by construction, no external triangle is ever deleted by the Bowyer–Watson algorithm and all the new triangles are therefore internal. However, it is necessary to subdivide the new triangles into accepted or nonaccepted and to further subdivide the new nonaccepted triangles and their nonaccepted neighbours into active or waiting.

The new triangles $\mathcal{T}_i$ are classified as accepted or nonaccepted depending on their circumcircle radii ρ_i. For every new triangle, consider the midpoint M_i of the edge opposite to the new point (see Fig. 5) and evaluate

$$\rho_{M_i} = f(\mathbf{x}_{M_i}).$$

If the ratio

$$\frac{\rho_{M_i}}{\rho_i} < \delta,$$

the triangle $\mathcal{T}_i$ is considered accepted, otherwise it is considered nonaccepted. The value of δ is purely empirical. Good results have been obtained choosing $\delta = 1.5$. The new nonaccepted triangles are then subdivided into active and waiting in a way that is analogous to that described in Section 6.1.

Since no external triangle is ever deleted, there is no need to subdivide triangles adjacent to the new ones into external and internal. In fact they will simply retain the external or internal attribute they had prior to the new point insertion. Moreover, since the insertion of the new point does not affect the geometry of the triangles adjacent to the new ones, it is not necessary to subdivide these triangles into accepted or nonaccepted. As a consequence, also the accepted or nonaccepted attribute will be retained. It is, however, necessary to perform a subdivision of the nonaccepted triangles adjacent to the new ones into active or waiting, according to the same rules used for the new triangles. The mesh generation method stops when there are no more active triangles left.

An example showing the triangle progression during the mesh generation process is given in Figs. 6 and 7 for the simple case of an elliptical domain, where a uniform triangle distribution has been required.

As explained above, the new point is inserted on a Voronoi segment relative to the active triangle having the largest circumcircle radius, and this implies that computational efficiency can be achieved by sorting active triangles according to the values of their radii. The second mesh generation algorithm can therefore be summarized as follows:

1. Triangulate all the boundary points by means of the Bowyer–Watson algorithm.

2. Check if the triangulation is body conforming and recover all missing boundary edges.

3. Divide all triangles into internal and external. Divide internal triangles into accepted and nonaccepted on the

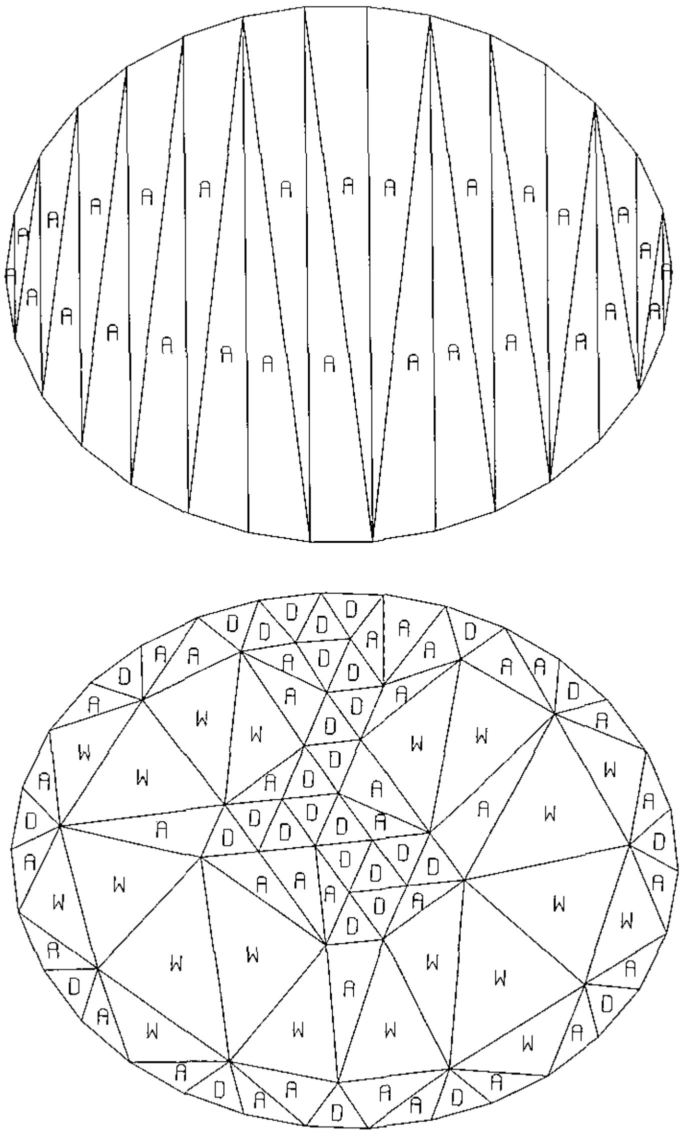

FIG. 6. Two successive steps of the mesh generation of an elliptical domain, "D" indicates accepted (done) triangles, "A" active triangles, and "W" waiting triangles. The first step shows the initial boundary triangulation.

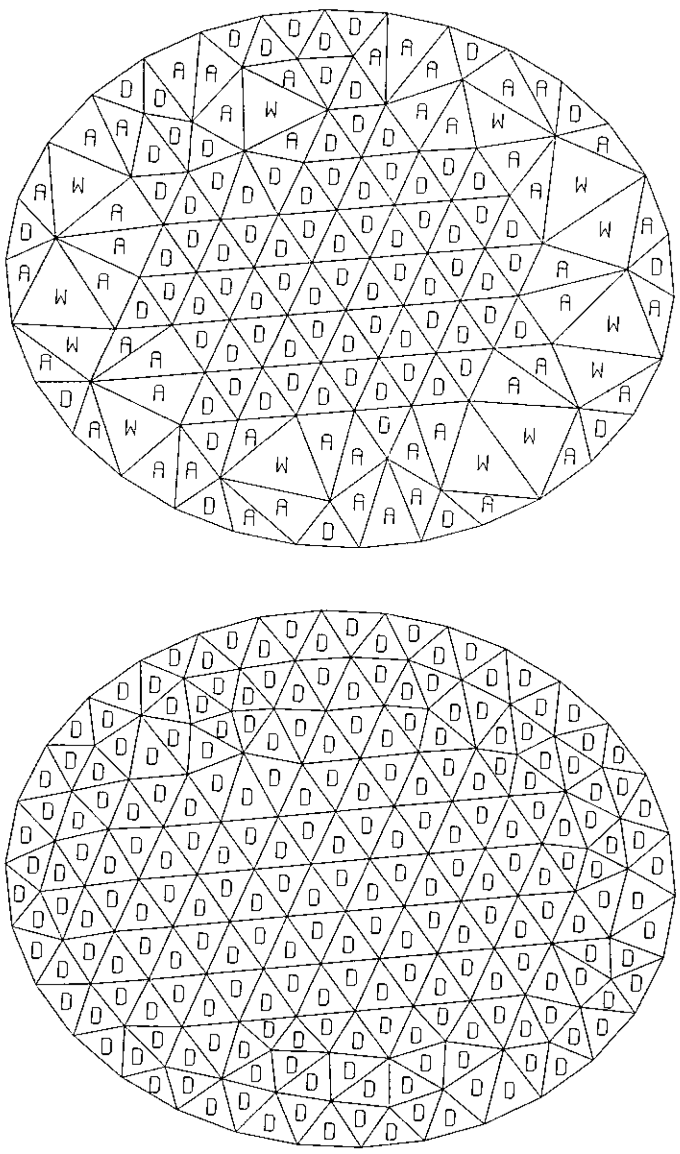

FIG. 7. Two successive steps of the mesh generation of an elliptical domain, "*D*" indicates accepted (done) triangles, "*A*" active triangles, and "*W*" waiting triangles. The last step shows the final triangulation.

basis of their circumcircle radii. Divide nonaccepted triangles into active and waiting.

4. Order the active triangles according to their circumscribed circle radii.

5. Consider the top element of the ordered list of the active triangles and insert the new point according to the Voronoi segment insertion criterion.

6. Regenerate the triangulation by means of the Bowyer–Watson algorithm.

7. Divide new triangles into accepted and nonaccepted on the basis of their circumcircle radii. Divide new nonaccepted triangles into active and waiting.

8. Divide nonaccepted triangles adjacent to the new ones into active and waiting.

If the active triangle list is empty stop, else go to step 5.

7. MESH GENERATION EXAMPLES

Several examples showing the potentialities offered by the proposed methods are given in this section. In the first place, we compare the relative merits of the two methods described so far. The performance of the second method is subsequently investigated by triangulating a domain of complex geometry. A more realistic example is then reported, showing a grid generated with the method described in Section 6 around a multiple airfoil configuration suitable for Euler computations. Finally, a simple example of three-dimensional grid generation for a cubical domain is displayed.

7.1. *Comparison of the Two Methods*

Four triangulations of a simple elliptical domain are here presented. A simple rectangular background grid has been employed to generate the four meshes. The differences between the two methods are investigated examining (1) a grid, where a uniform distribution of triangles over the entire domain has been required, and (2) a grid in which a clustering of the triangles along one of the diagonals of the background rectangle is obtained by imposing triangle dimensions at the vertices of the backgrid rectangle in a chequerboard fashion, the higher value being ten times the lowest.

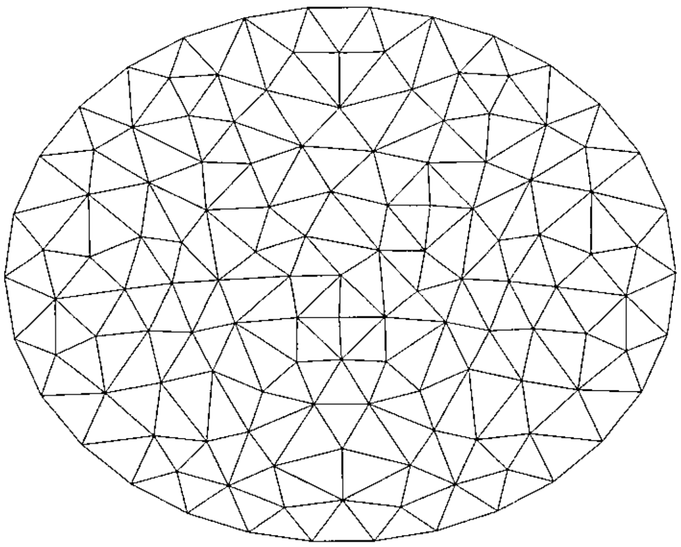

FIG. 8. Triangulation of an elliptical domain. Uniform triangulation generated by the first method.

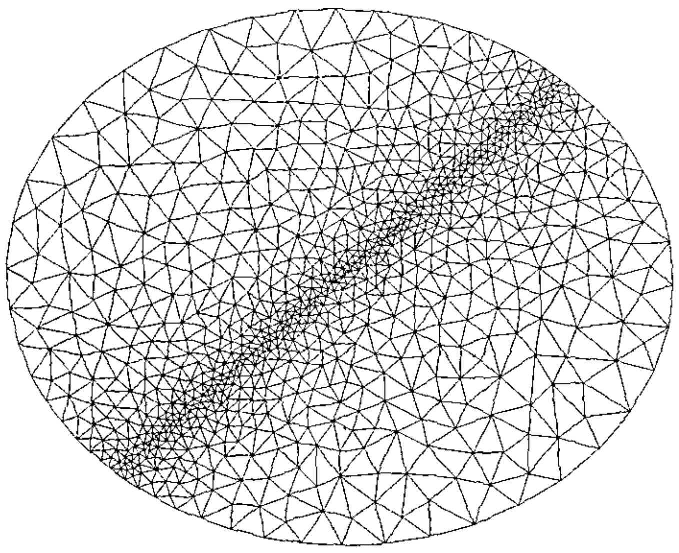
FIG. 9. Triangulation of an elliptical domain. Diagonally clustered triangulation generated by the first method.

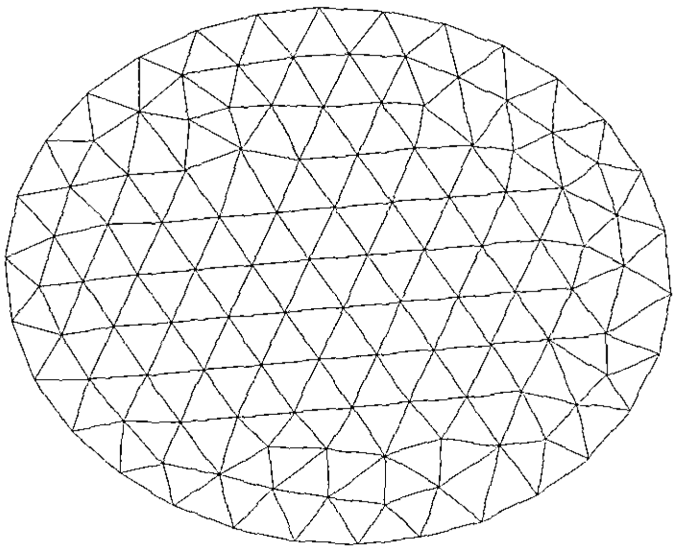
FIG. 11. Triangulation of an elliptical domain. Uniform triangulation generated by the second method.

Figures 8–10 show the results obtained by means of the first method, while Figures 11–13 show the results obtained with the second method. Figures 10 and 13 show the improvements obtainable for the mesh generated by means of the first and the second methods, respectively, by slightly repositioning all mesh points using a standard mesh smoothing (Laplacian filtering, see, e.g., [11]).

The two mesh generation methods can be compared quantitatively by defining an "aspect ratio" σ as the ratio between twice the inscribed and circumscribed circle radii. With this definition, σ ranges between 0 and 1. In fact, for a degenerate triangle having an angle with value approaching 0 or π the value of σ approaches 0, while for an equilateral triangle the aspect ratio value is equal to 1. Table I shows the fraction of triangles having aspect ratio σ falling between prescribed values σ_1 and σ_2 for the diagonally clustered triangulation computed with Method 1, Method 1 with smoothing, Method 2, and Method 2 with smoothing. For this problem, both mesh generation methods never generate triangles having an aspect ratio with value less than 0.35.

By comparing Fig. 8 with Fig. 11 it is apparent that the

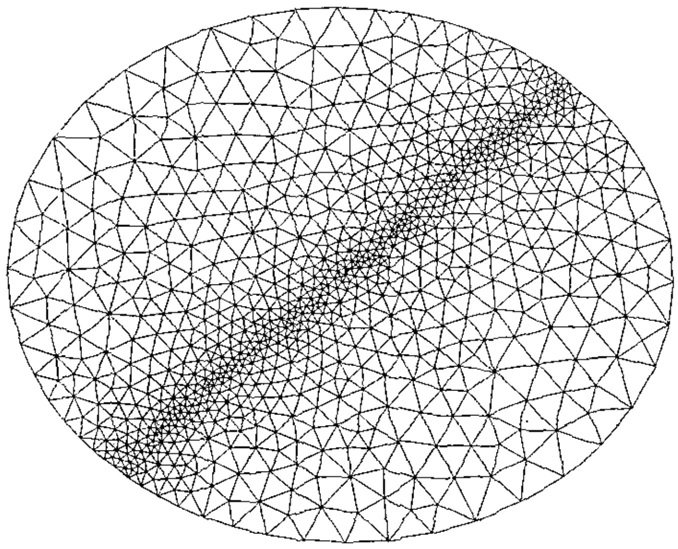
FIG. 10. Triangulation of an elliptical domain. Diagonally clustered, smoothed triangulation generated by the first method.

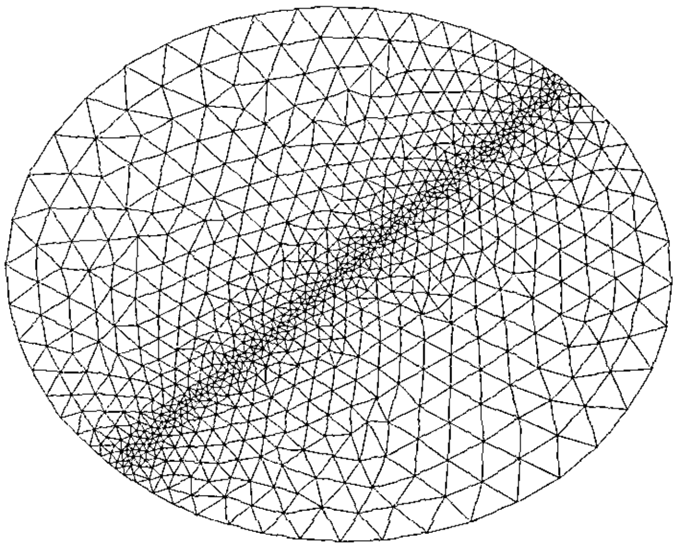
FIG. 12. Triangulation of an elliptical domain. Diagonally clustered triangulation generated by the second method.

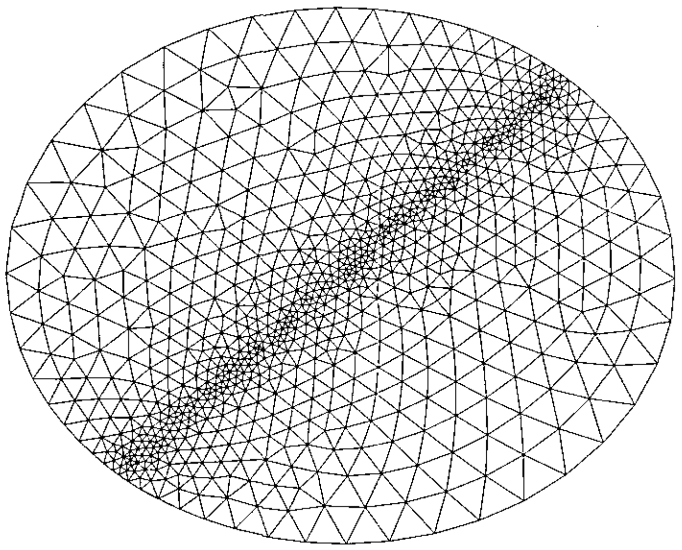

FIG. 13. Triangulation of an elliptical domain. Diagonally clustered, smoothed triangulation generated by the second method.

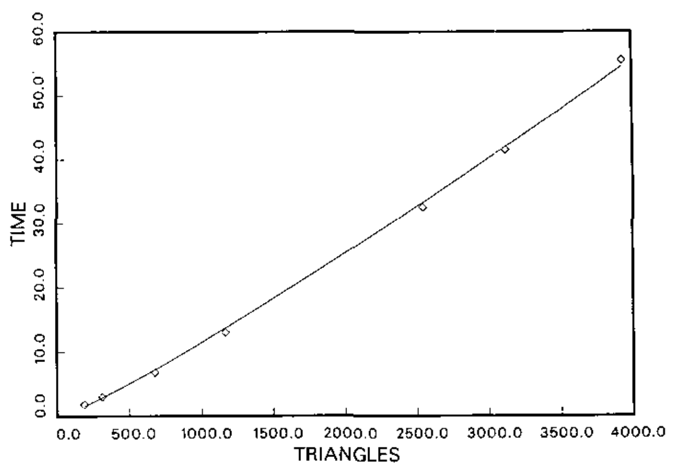

FIG. 14. Timing for the second mesh generation method.

second method is capable of generating a much more regular triangulation than the first one for uniform grids. Table I shows how this greater regularity is maintained also when a nonuniform grid distribution is prescribed. Note in particular that the triangles generated near the boundaries of the domain displayed in Figs. 12 and 13 have a much better "aspect ratio" in comparison with those shown in Figs. 9 and 10. Despite many efforts, no reasonable way to alleviate this problem has been found for the first method.

Figure 14 shows the CPU time versus the number of triangles for a uniform mesh computed by means of the second mesh generation method. The solid line represents the least squares approximation of the actual CPU time (diamonds in the figure) with the estimated $O(N \log N)$ time. This simple test case shows clearly the advantages offered by the second mesh generation method over the first. Due to this superiority, all the remaining examples have been limited to the investigation of the performances of the second method.

TABLE I

Aspect Ratio Distributions

Aspect ratio		Fraction of triangles			
		Method 1		Method 2	
σ_1	σ_2	Standard	Smoothed	Standard	Smoothed
0.35	0.40	0.00059	0.00000	0.00000	0.00000
0.40	0.45	0.00117	0.00117	0.00000	0.00000
0.45	0.50	0.00351	0.00000	0.00000	0.00000
0.50	0.55	0.00527	0.00059	0.00060	0.00000
0.55	0.60	0.01347	0.00000	0.00658	0.00060
0.60	0.65	0.01288	0.00000	0.00598	0.00120
0.65	0.70	0.02342	0.00351	0.00957	0.00060
0.70	0.75	0.02869	0.00761	0.02751	0.00120
0.75	0.80	0.06792	0.01288	0.04844	0.01316
0.80	0.85	0.15398	0.03396	0.07596	0.01316
0.85	0.90	0.18150	0.07436	0.10176	0.04785
0.90	0.95	0.20433	0.22834	0.16208	0.15550
0.95	1.00	0.30328	0.63759	0.56160	0.76675

Note. Diagonally clustered triangulation generated with Method 1, Method 1 with smoothing, Method 2, and Method 2 with smoothing.

7.2. *Triangulation of a Multiply Connected Domain*

The general applicability of the method is well demonstrated by this example which shows the capability of the

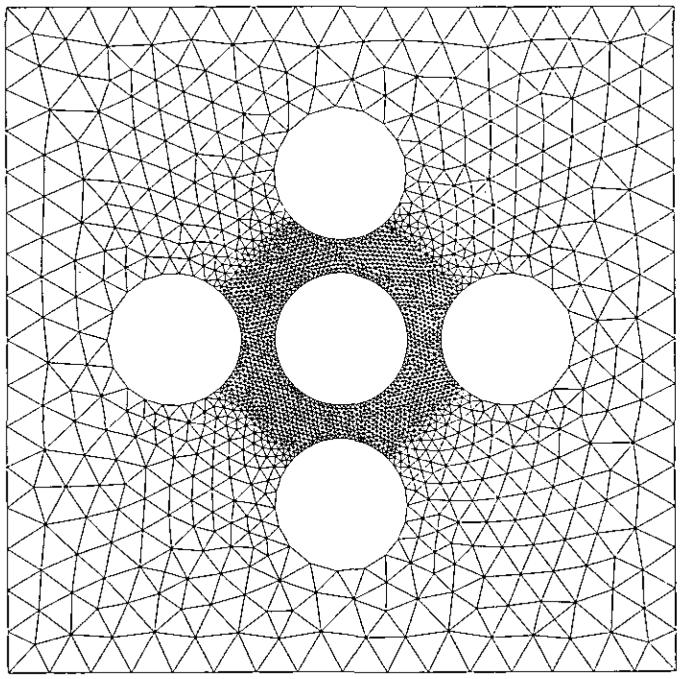

FIG. 15. Multiconnected domain. Triangulation of the entire domain.

method to treat domains of complex geometry as well as the possibility to prescribe a nonuniform mesh distribution in a completely arbitrary way.

Figure 15 shows the entire domain triangulated. A background grid imposing a constant value for the triangle dimension in the square defined by joining the centres of the five circles has been employed to generate the grid. Figure 16 shows an enlargement of the triangulation in the region laying among the circles. It is to be noted that the high quality displayed by the triangles obtained in this test case is a result due to the combined effect of Delaunay triangulation and the proposed point-generation strategy.

7.3. *Triangulation around a Multi-airfoil Configuration*

The last two-dimensional example shows the results provided by the second method for a realistic application, i.e., the mesh generated around a multiple element airfoil configuration [8]. The complete mesh extends for some 10 chords away from the airfoils. A small portion of the triangulation surrounding the airfoil configuration is shown in Fig. 17, while a detail of the triangulation in the channel between the main airfoil and the maneuver flap is shown in Fig. 18 and 19 without and with Laplacian smoothing, respectively. The possibility of prescribing the size of the triangular elements as a function of their position in the domain in a completely general way should be apparent from the above figures. It is to be noted once again the good quality possessed by triangles adjacent to the domain boundaries, an essential feature for a grid intended for accurate numerical computations.

7.4. *Three-Dimensional Mesh Generation*

A straightforward generalization of the second method has been implemented and applied to a simple but not trivial three-dimensional example, which shows the potentialities of the proposed method also for three-dimensional applications. A simple cubical domain has been tetrahedronized, prescribing a nonuniform distribution of the tetrahedra size.

The starting point for the three-dimensional mesh generation method is a boundary tetrahedronization obtained by connecting a surface distribution of boundary points constructed by means of the two-dimensional method. From this point on, by following the simple steps reported at the end of Section 6, new points are generated inside the domain, until all the internal tetrahedra have been classified accepted.

Figure 20 shows one of the six (equal) faces of the cubical

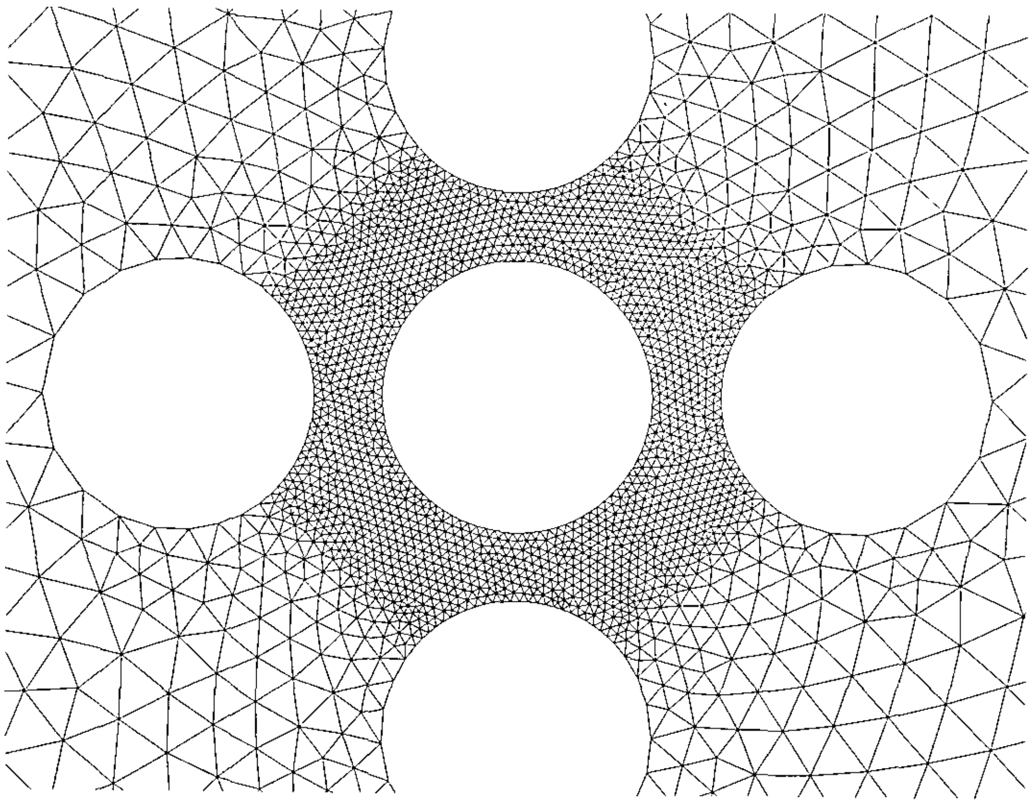

FIG. 16. Multiconnected domain. Enlargement of the triangulation of the region among the circles.

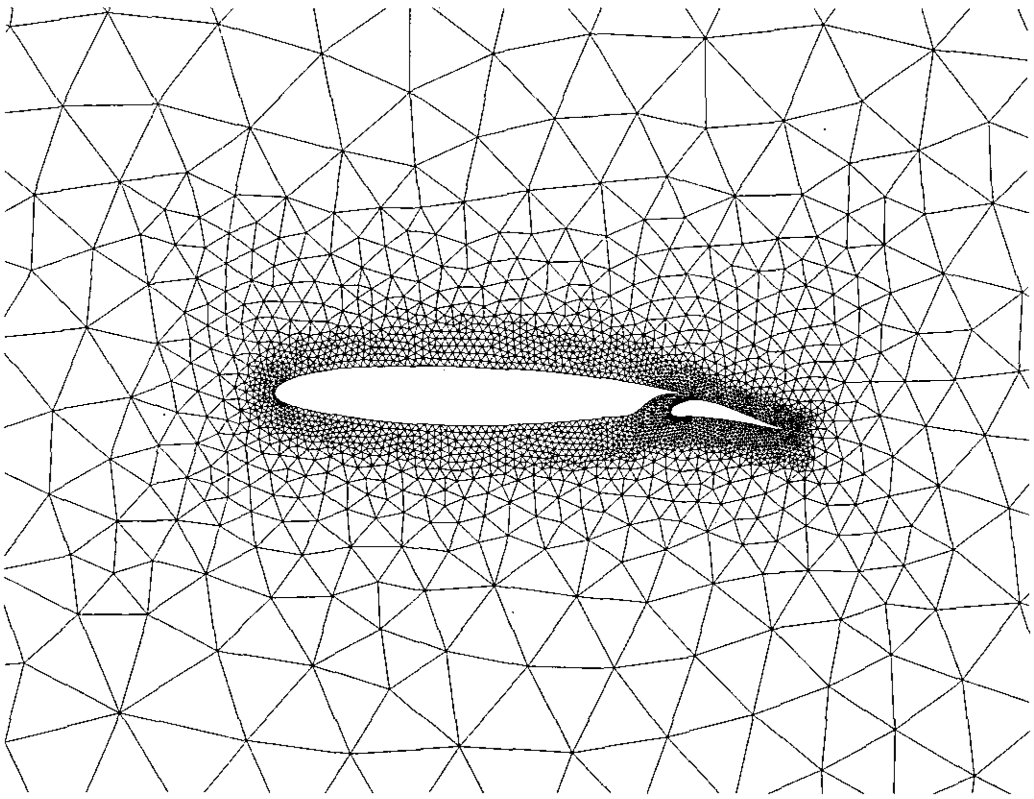

FIG. 17. Multi-airfoil. Triangulation of the region around the airfoil configuration.

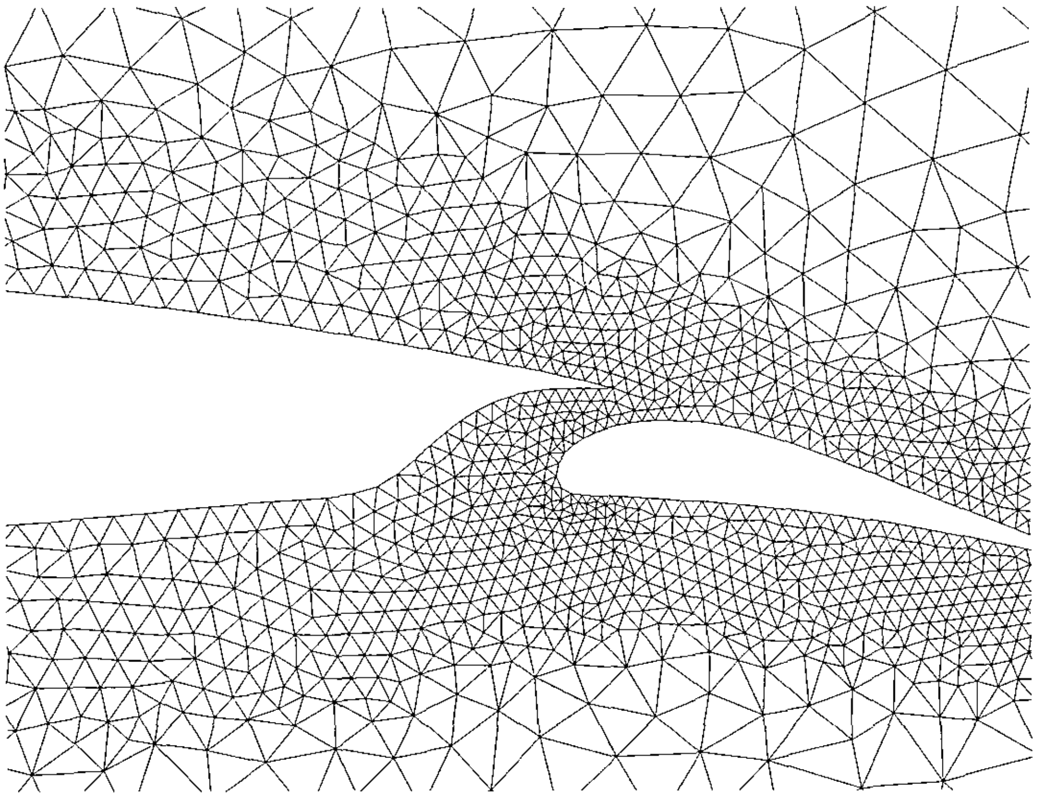

FIG. 18. Multi-airfoil. Enlargement of the triangulation of the channel between the main element and the flap.

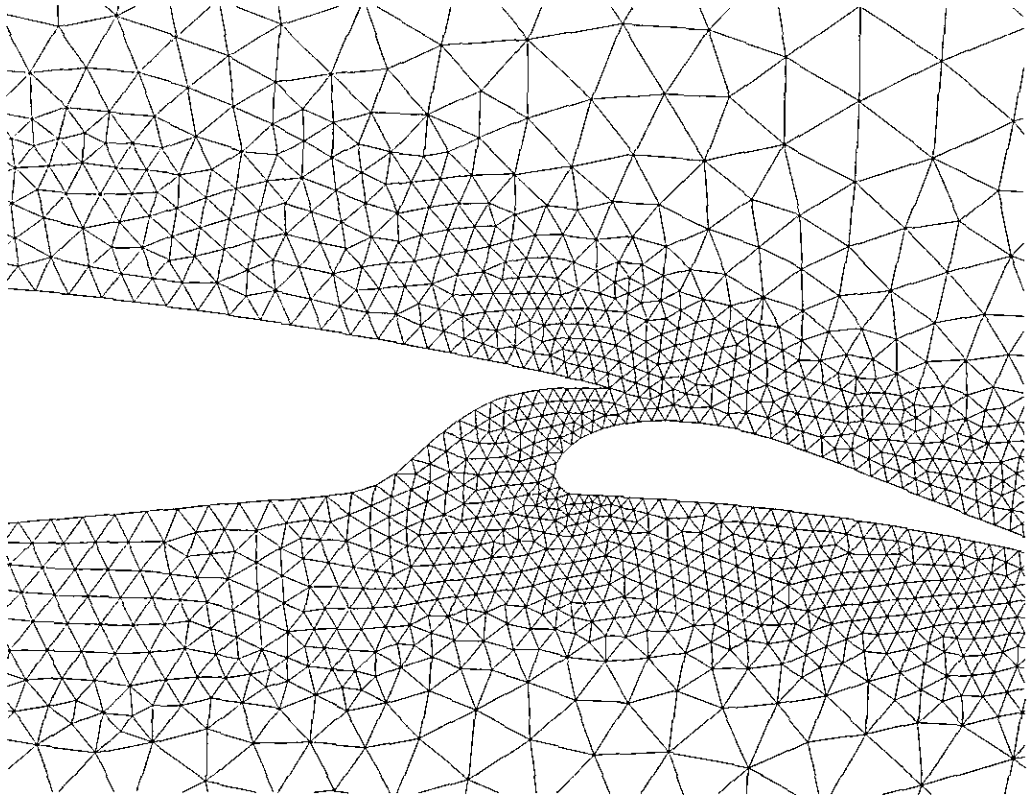

FIG. 19. Multi-airfoil. Enlargement of the smoothed triangulation of the channel between the main element and the flap.

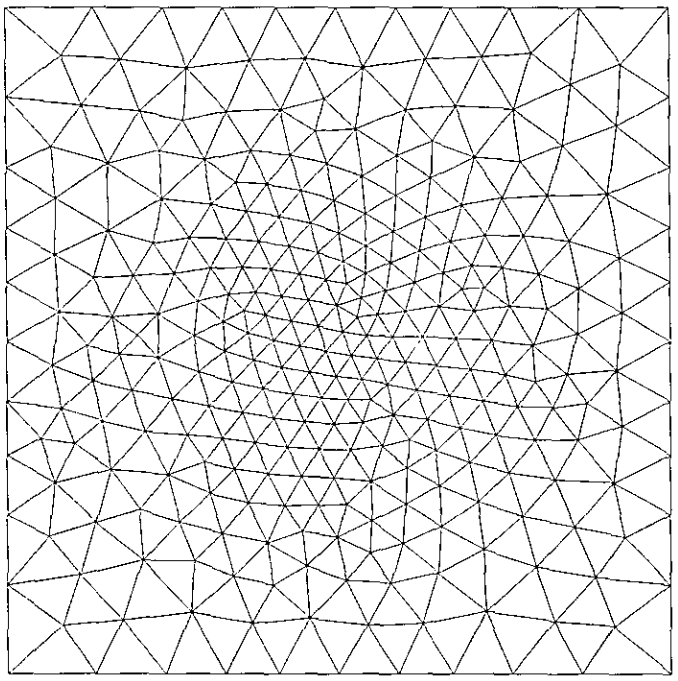

FIG. 20. Three-dimensional cubical domain. Surface triangulation of one of the six (equal) faces of the cube.

domain. Note the nonuniform triangle distribution over the cube faces, which is matched by a corresponding nonuniform distribution also in the interior of the cubical domain.

The entire grid contains about 50,000 tetrahedral elements and has been generated on a workstation HP835 in about half an hour of CPU time. Work is in progress to extend the proposed method to treat three-dimensional domains of complex geometry also.

8. CONCLUSIONS

Two mesh generation methods relying on the Bowyer–Watson algorithm and computing point positions and connections simultaneously have been presented. The methods distinguish themselves by the great simplicity and geometrical flexibility, as demonstrated in all the test cases presented in Section 7, and they allow an implementation with operation count $O(N \log N)$. By virtue of Bowyer–Watson algorithm, the proposed techniques can be easily extended to three dimensions.

The first method is just a minor modification of the method proposed by Holmes and Snyder in [5] and allows control of the triangle dimension distribution on the domain in a general and flexible way.

The second algorithm generalizes the ideas contained in the first method but includes an original strategy to generate mesh points in the domain. Although it bears some resemblance with ideas commonly considered in advancing front techniques, this method, due to the full exploitation of the geometrical properties of the Delaunay triangulation and the associated Dirichlet tessellation in conjunction with the local nature of the Bowyer–Watson algorithm, provides a completely new general procedure which eliminates the need for the complex topological validity checks usually employed by advancing front methods and avoids by construction the closure difficulties which are sometimes faced by this kind of method.

The second method appears to be capable of generating grids characterized by generally better shaped triangles in comparison with the first method. This is especially true near the boundaries of the domain. This property is not really surprising if we consider that the second method combines some aspects of the first method with some ideas of the advancing front techniques, the latter giving their best near the boundaries of the domain.

At present, it is possible to control the size of the triangular elements but not their shape. An extension of the point insertion strategy in order to generate anisotropic point distributions does not seem to be difficult. Nevertheless, the natural tendency of the Delaunay triangulation to generate triangles that are as equilateral as possible could prevent the construction of highly stretched meshes. Further investigations are required to address this issuc. More work is also needed to extend the method to treat three-dimensional domains of complex geometry for large-scale, real-life applications of industrial interest.

REFERENCES

1. T. J. Baker, "Three-Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets", Proceedings, AIAA 8th CFD Conference, Hawaii, June 1987.
2. A. Bowyer, *Comput. J.* **24**, No. 2, 162 (1981).
3. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, (MIT Press, Cambridge, MA McGraw–Hill, New York, 1990), p. 140.
4. A. Jameson, T. J. Baker, and N. P. Weatherill, "Calculation of Inviscid Transonic Flow over a Complete Aircraft," 24th AIAA Aerospaces Sciences Meeting, Reno; AIAA Paper 86-0103, 1986 (unpublished).
5. D. G. Holmes, and D. D. Snyder, "The Generation of Unstructured Triangular Meshes using Delaunay Triangulation," *Numerical Grid Generation in Computational Fluid Mechanics'88* (Pineridge Press, Swansea, UK, 1988).
6. J. Peraire, M. Vadhati, K. Morgan, and O. C. Zienkiewicz, *J. Comput. Phys.* **72**, 449 (1987).
7. J. Peraire, M. Peiro, L. Formaggia, K. Morgan, and O. C. Zienkiewicz, *Int. J. Numer. Methods Eng.* **26** (1988).
8. E. Stanewsky, and J. J. Thiebert, "Airfoil SKF 1.1 with Manoeuvre Flap," in *Experimental Data Base for Computer Program Assessment, AGARD-AR-138, May 1979.*
9. D. F. Watson, *Comput. J.* **24**, No. 2, 167 (1981).
10. N. P. Weatherill, Princeton University MAE Report No. 1715, July 1985 (unpublished).
11. N. P. Weatherill, Lecture Series 1990-06, VKI, 1990 (unpublished).