Computers Math. Applic. Vol. 24, No. 5/6, pp. 129–150, 1992
Printed in Great Britain. All rights reserved

0097-4943/92 $5.00 + 0.00
Copyright© 1992 Pergamon Press Ltd

DELAUNAY TRIANGULATION IN COMPUTATIONAL FLUID DYNAMICS

N. P. WEATHERILL
Institute for Numerical Methods in Engineering
University College of Swansea
Singleton Park, Swansea, U.K.

Abstract—A method, which utilises the Delaunay criterion, is described by which computational grids consisting of assemblies of triangles or tetrahedra can be constructed. An algorithm is briefly outlined to construct the triangulation and its dual, the Voronoi diagram. Issues related to how to construct boundary conforming grids from such a triangulation are addressed, and details are presented of how grid points within the domain can be generated automatically. The point generation algorithm utilises either the given boundary point distribution, or, for grid adaption, a background mesh. Computational aspects of constructing the triangulation in both 2 and 3 dimensions are covered. Examples of meshes and flow computations for a range of aerospace geometries are presented.

1. INTRODUCTION

For most practical problems in fluid dynamics it is necessary to solve the governing equations numerically with the assistance of computers. The formulation of the equations into a form for numerical solution generally requires the flow domain to be discretized using a set of points which can be connected to form cells or elements. Such techniques as the finite difference or finite element method require this underlying space discretization. The computational grid or net of points and elements replaces the continuum, and the governing partial differential equations are replaced by appropriately constructed discretized forms. The mesh must accurately model the geometrical boundaries and it is essential that grid points are available in regions of the domain where flow features develop. Further, the relative grid point spacing and point connectivities influence the accuracy of a solution.

With major advances in the numerical solution of the Euler and Reynolds-Averaged Navier-Stokes equations, the importance of the grid and its influence on accuracy has grown. Consequently, in recent times techniques for the automatic generation of computational grids have received much attention.

Many techniques have been explored [1,2]. A rather simple but useful classification is that of methods which generate structured meshes, which possess curvilinear coordinate networks of points and associated regular connectivities, and methods which generate unstructured meshes which, in general, are assemblies of triangles in 2 dimensions and tetrahedra in 3 dimensions. Meshes of the latter type require a connectivity matrix to be introduced to explicitly define the connectivities between points. This is not required for a structured mesh since the points map to a matrix, where it is assumed that neighbors in the matrix are neighbors in the physical space in which the mesh is constructed. Traditionally, finite difference methods have employed structured curvilinear meshes, whilst finite element methods have utilised unstructured meshes. The relative advantages and disadvantages, from the viewpoint of computational fluid dynamics, together with details of other methods, have been discussed elsewhere [3].

The author gratefully acknowledges the support of the Royal Aerospace Establishment, Farnborough, U.K. and the Aircraft Research Association, Bedford, U.K.

One of the major advantages of the unstructured approach is that it is applicable to very complex geometrical domains and amenable to mesh adaption to features in the flow. This is related to the fact that unstructured meshes do not possess any global property and elements and points can be added and deleted locally as the geometrical or flow features dictate. This is unlike structured meshes which possess a global curvilinear coordinate system. The approach adopted for structured grids to alleviate this constraint is to subdivide the domain into a set of sub-regions. Each sub-region is topologically equivalent to a cuboid within which a structured mesh can be generated. The sub-regions are defined appropriate to a given geometry and designed to give a suitable grid topology. This approach, commonly called multiblock or composite grid generation, has proved highly successful [4–6].

This paper will describe a method of generating unstructured meshes consisting of an assembly of triangles in 2 dimensions and tetrahedra in 3 dimensions. Traditionally, finite element techniques, applied to a wide range of engineering problems, have utilised unstructured meshes and the literature is extensive [7]. In fluid dynamics, the advancing front technique [8] has proved to be particularly flexible. In this approach the mesh is generated by advancing in from the boundary, by creating points and connections which produce valid non-intersecting elements. The position of the points can be created to ensure adequate resolution of the domain and flow features. The connection of points is performed according to a set of pre-defined rules. In contrast, in the approach to be discussed here, the point connections, from which the elements are derived, uses a classical geometrical construction which has been recorded in the literature for over a century.

Our approach to mesh generation separates the problem into two parts. First, given a set of points, can a valid triangulation be defined, and second, how are the positions of the points derived? These problems will be addressed before demonstrating the method with examples.

2. VORONOI NEIGHBORHOODS

Dirichlet [9], in 1850, first proposed a method whereby a given domain could be systematically decomposed into a set of packed convex polygons. Given two points in the plane, P_i and P_j, the perpendicular bisector of the line joining the two points subdivides the plane into two regions, V_i and V_j. The region V_i is the space closer to P_i than to P_j. Extending these ideas, it is clear that for a given set of points in the plane, the regions V_i are the territories which can be assigned to each point, such that V_i represents the space closer to P_i than to any other point in the set. This geometrical construction of tiles is known as the *Dirichlet tessellation*. This tessellation of a closed domain results in a set of non-overlapping convex polygons, called *Voronoi regions*, covering the entire domain. This definition readily extends to higher dimensions where, for three dimensions, the Voronoi regions are convex polyhedra.

A more formal definition can be stated. If a set of points is denoted by $\{P_i\}$, then the Voronoi region $\{V_i\}$ can be defined as

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

i.e., the Voronoi region $\{V_i\}$ is the set of all points of p that are closer to P_i than to any other point. The sum of all points p forms a Voronoi polygon.

From this definition, it is clear that, in two dimensions, the territorial boundary which forms a side of a Voronoi polygon must be midway between the two points which it separates, and is thus a segment of the perpendicular bisector of the line joining these two points. If all point pairs which have some segment of boundary in common are joined by straight lines, the result is a triangulation within the convex hull of the set of points $\{P_i\}$. This triangulation is known as the Delaunay triangulation [10]. An example of this geometrical construction is given in Figure 1.

The equivalent is also true for higher dimensions. In three dimensions, the territorial boundary which forms a face of a Voronoi polyhedron is equidistant between the two points which it separates. If all point pairs which have a common segment in the Voronoi construction are connected, then a set of tetrahedra is formed which covers the convex hull of the data points.

The Delaunay triangulation possesses some interesting properties [11]. One of these is the in-circle criterion.

A triangulation $T(P_i)$ satisfies the in-circle criterion if and only if no point of the set P_i is interior to the circumcircle of any triangle of $T(P_i)$. This criterion provides a mechanism from which the Delaunay triangulation of a given set of points can be obtained. To illustrate this geometrical property, consider the four points P_1, P_2, P_3, P_4, as shown in Figure 2. If points P_1, P_2, P_3 form one triangle and points P_2, P_3, P_4 the other, it is seen that the circumscribing circle through the points P_1, P_2, P_3 encloses the fourth point P_4. Hence, this connection of points does not satisfy the in-circle criterion. However, if points P_1, P_2, P_4 and points P_1, P_3, P_4 are connected, then it is immediately obvious that the two circumscribing circles satisfy the in-circle criterion. An arbitrary triangulation can be transformed into a Delaunay triangulation by the repeated application of the in-circle criterion to each edge.

As a consequence of the in-circle criterion the maximum-minimum criterion follows. If the diagonal of any strictly convex quadrilateral is replaced by the opposite one, the minimum of the six internal angles will not decrease. Hence, the Delaunay triangulation produces a triangulation which is 'optimally equivalent' (Figure 3).

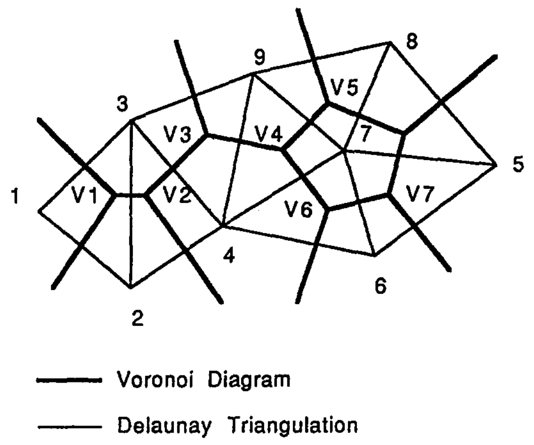


Figure 1. Voronoi regions and associated Delaunay triangulation.

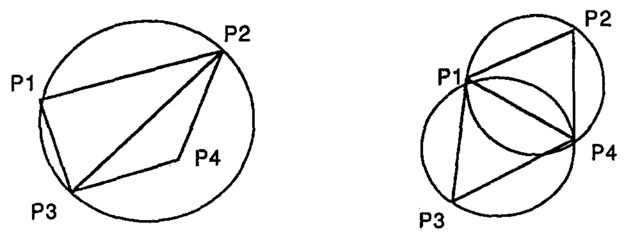


Figure 2. The in-circle criterion for 4 points.

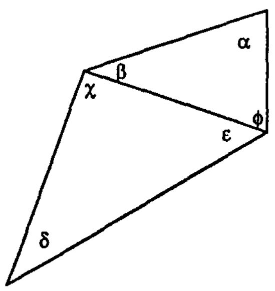


Figure 3. The diagonal of the convex quadrilateral maximises the minimum of the six internal angles.

Clearly, the in-circle criterion and the maximum-minimum criterion are desirable properties for the generation of regular unstretched meshes.

It is possible to completely describe the structure of the Voronoi diagram and Delaunay triangulation by constructing two lists for each Voronoi vertex; a list of the points which define a triangle for a given vertex of the Voronoi construction (so-called forming points), and a list of neighboring Voronoi vertices to a given Voronoi vertex. As an example, Table 1 contains the vertex structure for the construction shown in Figure 1.

Table 1. The data structure for the construction shown in Figure 1. (* signifies vertex not defined.)

Voronoi vertex	Forming points			Neighboring Voronoi vertices		
1	1	2	3	2	*	*
2	2	3	4	1	3	*
3	3	4	9	2	4	*
4	4	7	9	3	5	6
5	7	8	9	4	8	*
6	4	7	6	4	7	*
7	5	6	7	6	8	*
8	5	7	8	5	7	*

This data structure naturally extends to applications in three dimensions, where each Voronoi vertex has four forming points (tetrahedra of the Delaunay triangulation) and four neighboring Voronoi vertices.

This data structure provides the infra-structure to enable the Delaunay triangulation to be constructed. As will be shown, it can be used to implement fast search algorithms which are required for an efficient implementation of the Delaunay construction.

3. ALGORITHM TO CONSTRUCT THE DELAUNAY TRIANGULATION

The algorithm used to construct the Delaunay triangulation is a sequential process and follows the work of Bowyer [12]. Each point is introduced into an existing structure, which is broken and then reconnected to form a new Delaunay triangulation.

The algorithm applies to both two and three dimensions. In the presentation here, the terms between parentheses indicate the interpretation for three dimensions.

Step 1. Define the convex hull within which all points will lie. It is appropriate to specify 4 points (8 points) together with the associated Voronoi diagram structure.

Step 2. Introduce a new point anywhere within the convex hull.

Step 3. Determine all vertices of the Voronoi diagram to be deleted. A point which lies within the circle (sphere), centred at a vertex of the Voronoi diagram and which passes through its three (four) forming points, results in the deletion of that vertex. This follows from the 'in-circle' definition of the Voronoi construction.

Step 4. Find the forming points of all the deleted Voronoi vertices. These are the contiguous points to the new point.

Step 5. Determine the neighboring Voronoi vertices to the deleted vertices, which have not themselves been deleted. These data provide the necessary information to enable valid combinations of the contiguous points to be constructed.

Step 6. Determine the forming points of the new Voronoi vertices. The forming points of new vertices must include the new point, together with the two (three) points which are contiguous to the new point and form an edge (face) of a neighbor triangle (tetrahedron). (These are the possible combinations obtained from Step 5.)

Step 7. Determine the neighboring Voronoi vertices to the new Voronoi vertices. Following Step 6, the forming points of all new vertices have been computed. For each new vertex, perform a search through the forming points of the neighboring vertices, as found in Step 5, to identify common pairs of forming points. When a common combination occurs, then the two (three) associated vertices are neighbors of the Voronoi diagram.

Step 8. Reorder the Voronoi diagram data structure, overwriting the entries of the deleted vertices.

Step 9. Repeat Steps 2–8 for the next point.

The algorithm described here can connect an arbitrary set of points which lie within a convex hull. An example, in 2 dimensions, is given in Figure 4. The algorithm described above is the basis for connecting points. However, it must be augmented with other routines for use as a mesh generator.

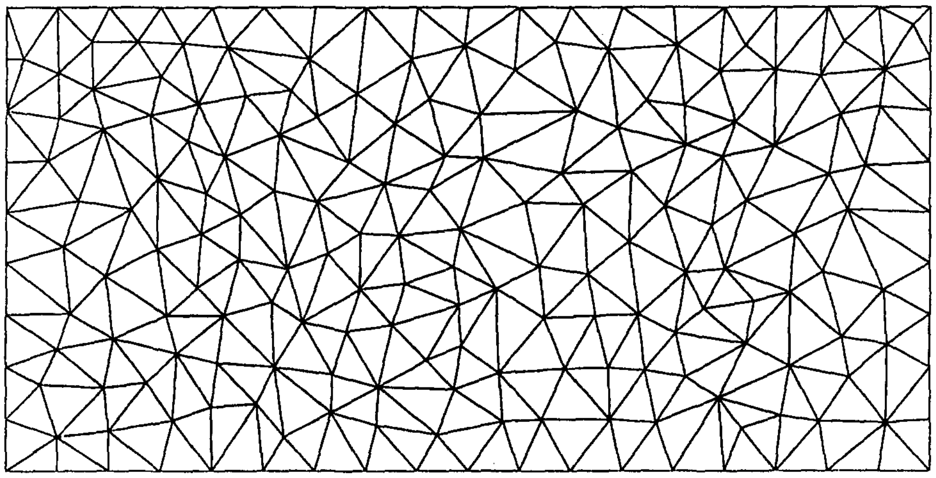


Figure 4. Delaunay triangulation of an arbitrary set of points.

4. APPLICATION TO MESH GENERATION

The Delaunay triangulation, its geometrical properties and how to construct it, have been widely known for a considerable time. However, the application of the construction to mesh generation in computational fluid dynamics has only recently been explored [13].

The previous sections have discussed how to create a triangulation from a given set of points. Three basic problems now remain in applying the technique to mesh generation. The Delaunay triangulation produces a triangulation of a geometrically convex domain. Triangles are produced inside and outside boundaries of the domain and, hence, it is necessary to identify these and delete them in the mesh generation procedure. Furthermore, a major requirement of a mesh generation procedure is to ensure that the mesh is boundary conforming, i.e., in two dimensions, edges in the triangulation form the boundaries and, in three dimensions, triangular faces of the assembly of tetrahedra conform to the boundary surface. Unfortunately, given a set of points $\{P_i\}$ and the corresponding triangulation $\{T_j\}$, obtained from the Delaunay triangulation of the points, there is no guarantee that the bounding edges will be contained within $\{T_j\}$. The triangulation routine described in Section 3 does not account for a requirement that certain edges must be recovered to ensure a boundary conforming mesh. Hence this problem must be addressed. Also, as mentioned in the introduction, the Delaunay criterion provides a mechanism to connect an arbitrary set of points to form a valid triangulation. It does not indicate how the points should be generated. However, for general flexibility of a mesh generator based upon the Delaunay approach, some technique is required to automatically define the points. Each of these problems will now be discussed.

4.1. Boundary Integrity

As an example of the boundary integrity problem, consider the boundary point distribution shown in Figure 5. The spatial distribution of points is such that it is not possible to ensure that edge AB is contained within a Delaunay triangulation, since it proves impossible to construct a circle through AB and one other point which does not include at least one other point in the set $\{P_i\}$.

A similar situation can be arise in three dimensions. Consider the triangular face in Figure 6a consisting of the edges $[P_1, P_2]$, $[P_2, P_3]$, and $[P_3, P_1]$. Although these edges may be contained within the three-dimensional triangulation this does not guarantee that the triangular face is present, since another tetrahedra could penetrate through the face (Figure 6b).

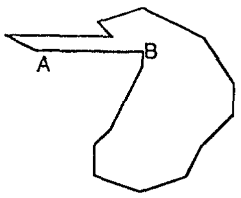
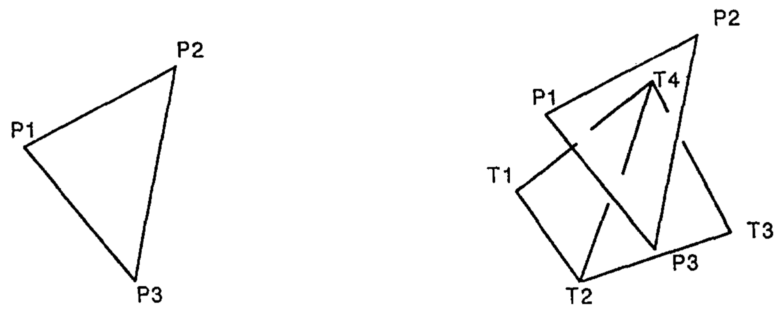


Figure 5. Boundary point distribution. Edge AB cannot be recovered in a Delaunay triangulation of these points.



(a) Edges of the triangular face. (b) Edges recovered but not the face.

Figure 6.

The boundary connectivity constraint, required of a mesh generator, is not naturally built into the Delaunay construction. The length scales and point distribution determine the point connectivities, irrespective of the boundary surfaces. Hence, it is necessary to ensure that the problem is well defined and that the boundary data is appropriately included in any Delaunay triangulation algorithm.

This problem is widely recognised and several approaches to its solution have been proposed [14,15]. Here two methods are presented which have proved successful. Both are applicable in two and three dimensions.

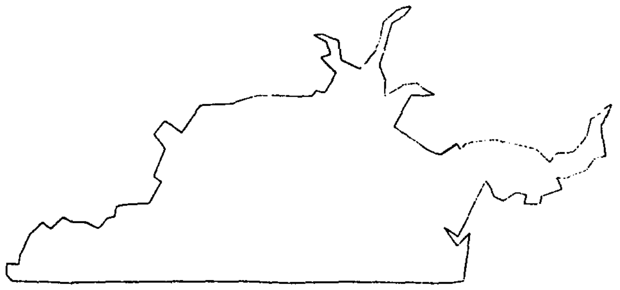
4.1.1. Boundary Integrity in Two Dimensions

Two approaches to ensure that a given set of edges can be recovered within a Delaunay triangulation will be presented. The first has been described in detail elsewhere [16] and only the outline of the method will be given. It is particularly interesting in that it contains aspects of the advancing front method and provides a way of combining this technique with the Delaunay triangulation.

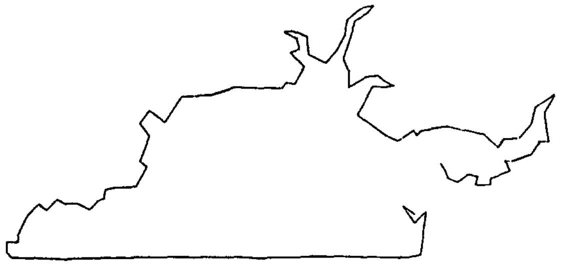
The Creation of a Point for Every Boundary Edge

The basis for this algorithm, which can ensure that arbitrary shapes can be made to conform to the Delaunay criterion, can be found in the work of Peraire [17] on the mesh generation procedure based on the concept of the advancing front. In this work, an algorithm was proposed whereby triangles were formed by connecting edges on the front to new points or existing points. The procedure ensures that no triangles can be formed which contain any other point or that edges intersect any other edge (except at the given nodes of the triangulation). It is clear that if this first criterion is replaced by one which ensures that no other points lie within the circumcircle through the three points which form the triangle, then the resulting triangulation will be Delaunay-satisfying.

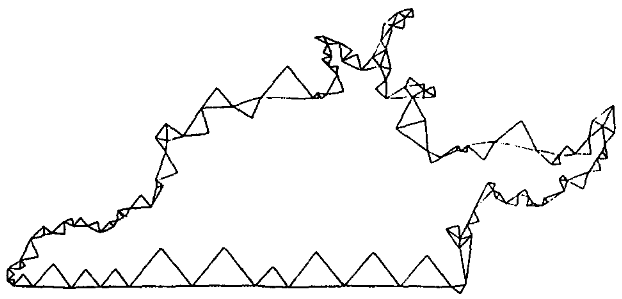
Weatherill [16] has adopted this basic idea and constructed an algorithm to ensure that given boundary edges are recovered in the Delaunay triangulation. An example of the application of this algorithm, at various stages, is shown in Figure 7.



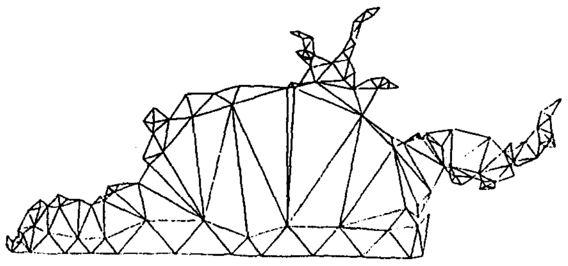
(a) Boundary of the estuary.



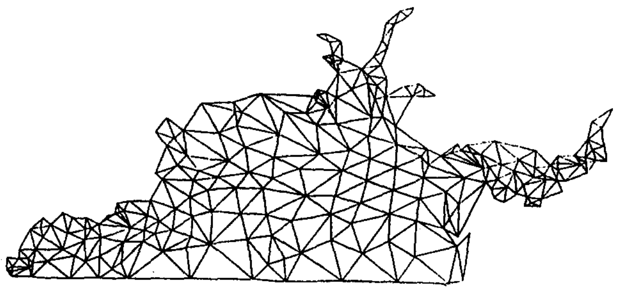
(b) Edges of the estuary, as obtained after the Delaunay triangulation.



(c) Point distribution derived from the algorithm.



(d) Delaunay triangulation.



(e) Final triangulation after addition of points and smoothing.

Figure 7. Application of the algorithm to the profile of an estuary (Carmarthen Bay, Seven Estuary, U.K.).

Edge Checking

The second method involves an *a posteriori* approach, in that the given boundary edges are checked after the Delaunay triangulation has been completed for a given set of points.

The following algorithm outlines the approach.

```
START

STEP 1  Input boundary points $\{P_i\}$, $i = 1, IP$ and boundary point
        connectivities which define the edges $\{E_j\}$, $j = 1, JP$.
STEP 2  Input field points $\{PP_k\}$, $k = 1, FP$.
STEP 3  PERFORM DELAUNAY TRIANGULATION OF $\{P_i\}$ and $\{PP_k\}$ to
        obtain triangulation $\{T_m\}$, $m = 1, NT$.
STEP 4  (i)   Are boundary edges $\{E_j\}$ contained in $\{T_m\}$.
        (ii)  If yes - go to STEP 5.
        (iii) If no  - add new point at mid-point of all missing edges.
                       New point set $\{PP_1\}$, $1 = 1, MP$ and new derived edges $\{EM_n\}$.
                       PERFORM DELAUNAY TRIANGULATION OF $\{PP_1\}$
                       Add $\{PP_1\}$ to $\{P_i\}$ and add $\{EM_n\}$ to $\{E_j\}$
                       Return to STEP 4(i).
STEP 5  Identify all triangles outside domain of interest $\{TO_m\}$, $m = 1, NP$
STEP 6  Delete triangles $\{TO_m\}$, $m = 1, NP$.

END
```

This procedure is different from the approached described in Method 1, in that it performs an *a posteriori* check on whether the boundary edges are recovered in the Delaunay triangulation. If an edge is missing, an additional point is inserted which acts to 'block' the breakthrough. This process has been called 'stitching' [18]. An example of this procedure is shown in Figure 8.

Comparing the two methods described, the second method is the simplest to implement and, in general, proves to be more computationally efficient in that a point is created only when required, rather than in the first approach where a point is assigned to every edge of the boundary. However, both techniques have proved successful for a variety of configurations.

4.1.2. Boundary Integrity in Three Dimensions

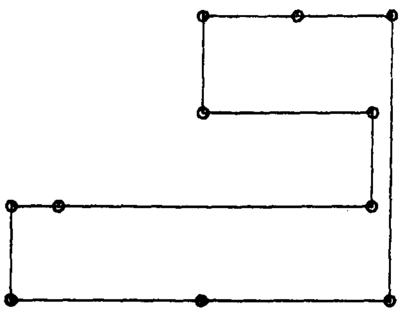
The two methods to ensure boundary integrity in two dimensions can be extended into three dimensions.

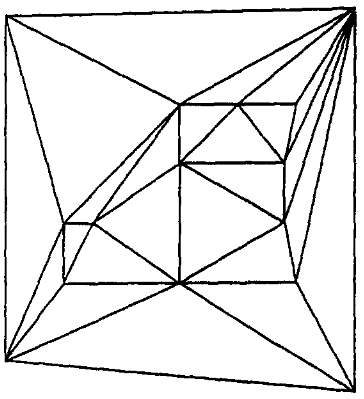
The Creation of a Point for Every Boundary Face

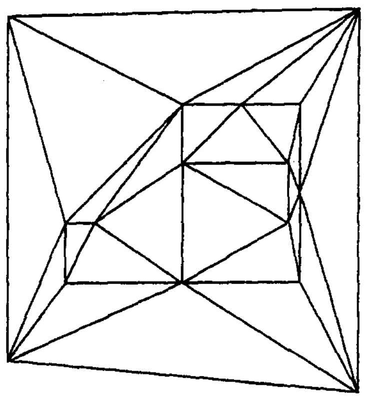
This approach is the three-dimensional extension of the algorithm described for two dimensions. A point is assigned to every boundary face ensuring that the sphere through the resulting four points does not include any other points. Details of this algorithm, together with examples, are given elsewhere [3]. Although the method works well, it is more computationally expensive than the *a posteriori* approach of the following method.

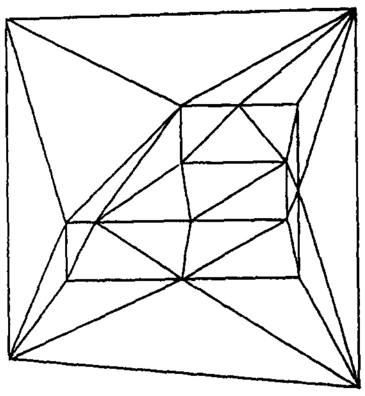
Edge and Face Checking

The previous method used an *a priori* approach, in which points were created before the triangulation process, to ensure that the resulting triangulation was boundary conforming. Of course, such an approach does not take into account the fact that many surface faces would naturally be recovered in the triangulation and, hence, the effort expended in creating a point


(a) The initial points which define the boundary. The edges shown must be recovered in the final triangulation.


(b) The Delaunay triangulation of the original set of points. Note, also shown are the 4 points which define the convex hull. All the original edges are not recovered in the triangulation.


(c) One point is added at the mid-point of one of the required edges which is missing. This then produces a triangulation from which one of the required edges can be recovered.


(d) Another point is added. This recovers the other required edge.

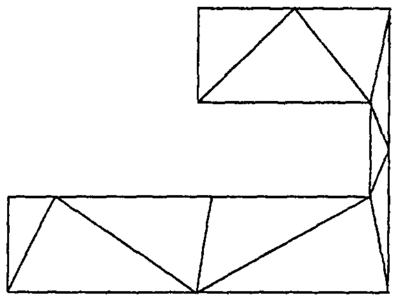

(e) All triangles outside the domain are deleted. Note the additional boundary points created to ensure boundary integrity.

Figure 8. An example showing the process of ensuring boundary integrity using the addition of boundary points.

for such faces is unnecessary. Alternatively, an *a posteriori* approach can be adopted whereby the triangulation is performed and then tested to determine the faces which are missing. Only at this stage are new points added to ensure boundary integrity. Such an *a posteriori* approach is now described.

The global procedure is outlined below.

```
START

STEP 1  Input boundary points $\{P_i\}$, $i = 1, IP$ and boundary point
        connectivities $\{C_j\}$, $j = 1, JP$.
STEP 2  Input field points $\{PP_k\}, k = 1, FP$.
STEP 3  Derive boundary edges $\{E_l\}, l = 1, JE$ from boundary
        connectivities $\{C_j\}$.
STEP 4  PERFORM DELAUNAY TRIANGULATION OF $\{P_i\}$ and $\{PP_k\}$ to
        obtain triangulation $\{T_m\}$, $m = 1, NT$.
STEP 5  (i)   Are boundary edges $\{E_l\}$ contained in $\{T_m\}$
        (ii)  If yes - go to STEP 6.
        (iii) If no - add new point at mid-point of all missing edges.
                  New point set $\{PP_n\}$ $n = 1, MP$ and new derived edges $\{EM_0\}$.
                  PERFORM DELAUNAY TRIANGULATION OF $\{PP_j\}$.
                  Add $\{PP_n\}$ to $\{P_i\}$ and add $\{EM_0\}$ to $\{E_l\}$.
                  Return to STEP 5(i).
STEP 6  (i)   Are boundary faces $\{C_j\}$ contained in $\{T_m\}$
        (ii)  If yes - go to STEP 7.
        (iii) If no - add new point at centroid of all missing boundary faces.
                  New point set $\{PS_r\}$, $r = 1, SP$ and new derived faces $\{SM_s\}$.
                  PERFORM DELAUNAY TRIANGULATION OF $\{PS_r\}$.
                  Add $\{PS_r\}$ to $\{P_i\}$ and add $\{SM_s\}$ to $\{C_j\}$.
                  Return to STEP 5(i).
STEP 7  Identify all tetrahedra outside domain of interest $\{TO_t\}$, $t = 1, NP$
STEP 8  Delete tetrahedra $\{TO_t\}, t = 1, NP$.

END
```

It is clear that the global procedure is iterative. Firstly, the boundary and field points are connected using the Delaunay triangulation. This is followed by a phase when the edges in the surface triangulation are recovered. Again, in general, this is also an iterative process whereby, for each edge which is missing, a new point is created which lies at the mid-point of the edge. All such points are then introduced to the Delaunay algorithm and the edges rechecked. This continues until all edges are recovered. This is a necessary step to ensure boundary conformity, but it is not sufficient since it is possible to have tetrahedra which penetrate the interior of a triangular face (see Figure 6). Hence, the next step is to check if the faces have been recovered. Due to the introduction of the additional edge points, some of the original triangles will now be formed by the sum of other smaller triangles. If a boundary triangle cannot be recovered, then a new point is added at the centroid of the surface region which is missing. These new points are then connected using the Delaunay triangulation, after which the edges are recovered followed by the surface faces.

As an example of this procedure, the algorithm has been applied to a double ellipsoid which has been considered extensively in the recent Hermes project. The surface mesh on the geometry and flow boundaries was obtained using a parametric representation of the surface geometry and use of the advancing front grid generator [19]. The field points were also obtained by the 'front'

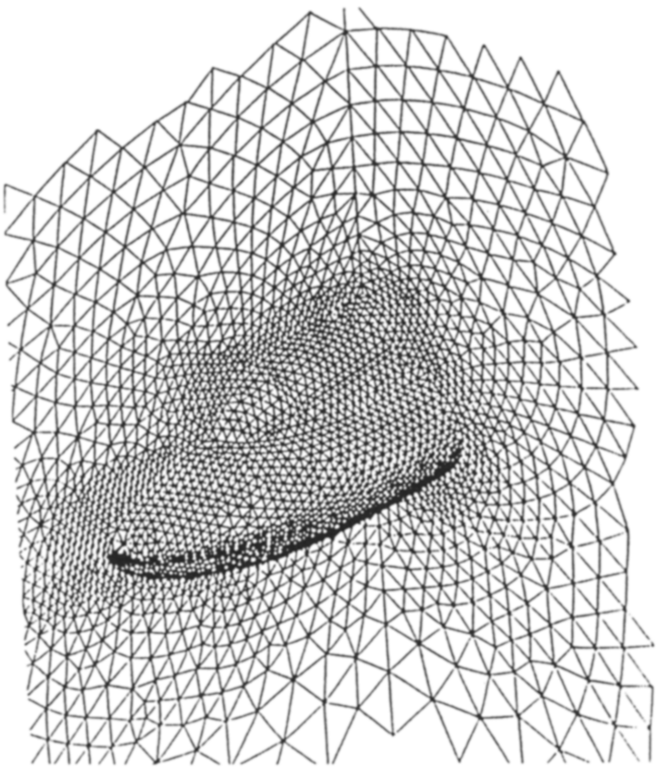


Figure 9. Surface triangulation of a double ellipsoid.

technique. A summary of the Delaunay triangulation and the boundary recovery history for this configuration is given in Table 2. A section of the surface triangulation is shown in Figure 9.

The general procedure for the Delaunay triangulation with boundary constraints indicates that all points should be added before the surface edge and face checking begins. A possible modification to this is to add the surface points, recover the surface faces, protect the tetrahedra outside the domain which contain the triangles which are on the surface, and then add the field points. Any field point which attempts to delete one of the protected tetrahedra is rejected. This procedure was implemented on this double ellipsoid geometry. However, it was found that the total number of rejected field points was 725, which is unacceptably high.

4.2. Rejection of Unwanted Triangles

The procedure outlined results in a triangulation covering the convex hull. It is necessary to delete all triangles which fall outside the domain of interest. This is achieved easily, since the boundaries between inside and outside of the domain are known and, following the procedures outlined in the Section 4.1, have been recovered in the Delaunay triangulation.

In two dimensions, a triangle is identified which lies interior to the domain. Using the data structure of the Voronoi diagram, a tree search is performed in which a neighbor triangle is visited if the common edge is not a boundary edge. Each triangle visited can be flagged as being inside the domain. When no triangle can be identified which has not already been visited, then all triangles not flagged are outside of the domain of interest and can be deleted. A similar procedure also applies in 3D, where one interior tetrahedron is found and a tree search is performed in which neighbor tetrahedra are visited provided this does not involve passing through a boundary face. Other techniques can be used which rely upon directionality of the boundary edges and faces. However, the procedure outlined proves to be very robust.

Table 2. Triangulation history for the double ellipsoid.

STATISTICAL SUMMARY FOR THE DOUBLE ELLIPSOID

INPUT DATA

No. of surface points:	3309
No. of field points:	22032
No. of surface faces:	6614
No. of surface edges:	9921
No. of tetrahedra obtained:	139436

After triangulation of all points: Edge recovery history

Iteration	Points	Tetrahedra	Surface Edges	Missing Edges
1	22040	139436	9969	48
2	22088	139728	10021	52
3	22140	140074	10065	44
4	22184	140344	10093	28
5	22212	140517	10109	16
6	22228	140614	10116	7
7	22235	140642	10121	5
8	22240	140669	10124	3
9	22243	140688	10125	1
10	22244	140689	10126	1
11	22245	140696	10126	0

To seal missing faces add: 0 points
All surface faces recovered

No. of tetrahedra outside domain:	9483
No. of tetrahedra inside domain:	131213

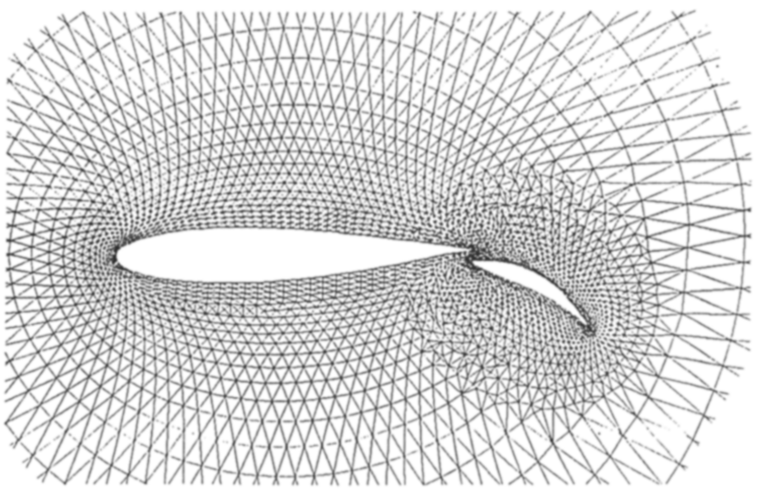
SUMMARY

No. of surface points added:	205
No. of surface edges added:	205
No. of additional surfaces faces:	410
No. of surfaces points:	3514
No. of field points:	22032
No. of surfaces faces:	7024
No. of surfaces edges:	10126
No. of tetrahedra:	131213

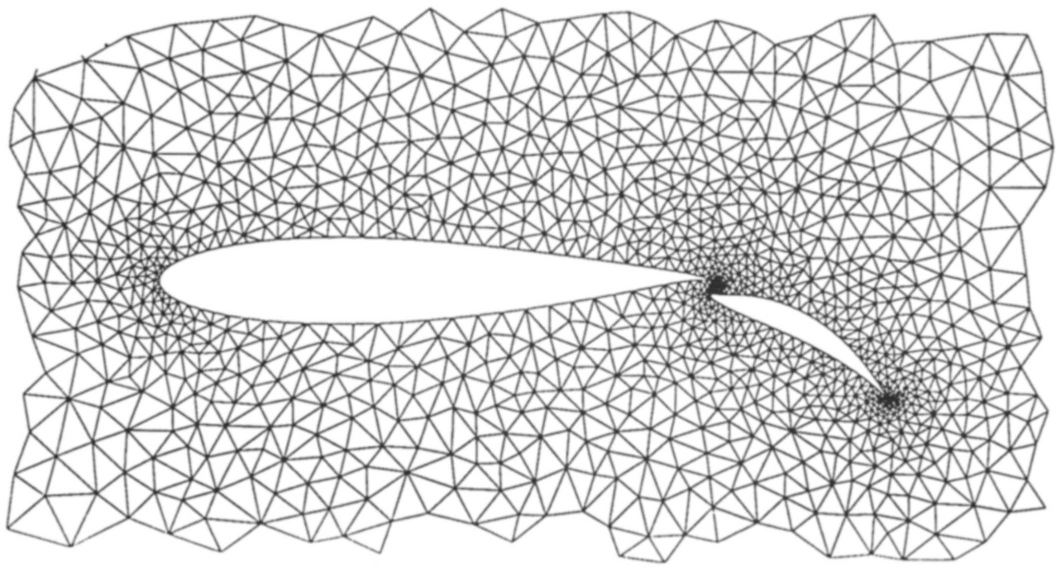
4.3. Point Generation

4.3.1. Points from an External Source

The points for connection by the Delaunay algorithm could be derived by a method external to the triangulation routine. Given a particular geometry, it may prove useful to adopt this approach. To illustrate this, consider a high lift multielement aerofoil configuration. It is relatively easy to generate a high quality mesh around each individual aerofoil component. The difficulty arises in connecting the component meshes. However, the Delaunay approach provides a suitable method of connecting points. Hence, a suitable solution method would be to generate component meshes, combine all the points and derive a connection of all these points. The triangulation process can be used to detect points which fall within an aerofoil component. This procedure results in a valid mesh as shown in Figure 10a. This approach was applied in both 2 and 3 dimensions in the early stages of our work [13,20,21].



(a) Triangulation obtained using points from a conformal mapping grid generator applied to each component.



(b) Triangulation using automatic point generation.

Figure 10. Delaunay triangulation for a two element aerofiol configuration.

This approach can lead to high quality meshes. However, it is clearly limited in its flexibility for arbitrary shapes, and hence it is necessary to devise a technique which can derive points for any shaped domain.

4.3.2. *Automatic Point Generation*

The computational domain is defined by the boundary points. It will be assumed that this point distribution reflects appropriate geometrical features, such as variation in curvature and gradient. Ideally, any method for automatic point generation should ensure that the boundary point distribution is extended into the domain in a smooth manner. A procedure, which has proved to be most successful in creating a smooth point distribution consistent with boundary point spacing and which naturally extends to 3D, is as follows.

(1) Compute the point distribution function for each boundary point x_i, i.e., for point i,

$$dp_i = \frac{\sqrt{(x_{i+1} - x_i)^2} + \sqrt{(x_i - x_{i-1})^2}}{2},$$

where it is assumed that 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, for the prospective point by interpolating the point distribution function from the nodes of the triangle.
   - (c) Compute the distances d_m, $m = 1,2,3$, from the prospective node to each of the nodes of the triangle.
     If $d_m < \alpha\, dp_m$, then reject the point. Return to the beginning of Step (3).
     If $d_m > \alpha\, dp_m$, then insert the point using the Delaunay triangulation algorithm.
   - (d) Assign the interpolated value of the point distribution function to the new node.
   - (e) Next triangle.

(4) Smooth the mesh.

The coefficient α is effectively the parameter which controls the grid point density.

An application of this algorithm, at various stages of point generation, is shown in Figure 11. As can be seen, an appropriate grid point distribution is obtained. The automatic point placement applied to the 2 component aerofoil produces a mesh of the kind shown in Figure 10b.

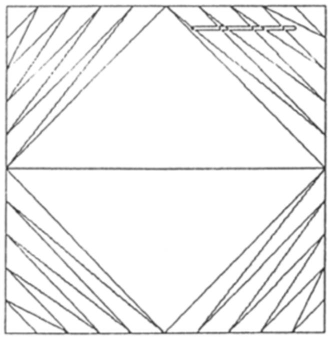
The procedure outlined above is appropriate when the only known quantity is the boundary point distribution. However, in many problems information is known about features in the domain, for example, if a flow simulation has been performed on a coarse mesh, the position of flow features which require further point resolution may be known. In such cases, it would be advantageous if the above simple algorithm could be amended to include such information. It proves possible to do this by using the concept of a 'background mesh.' This idea has been explored and fully exploited in other unstructured mesh methods, such as in the work of Peraire *et al.* [8]. Here, its application to the Delaunay approach will be outlined.

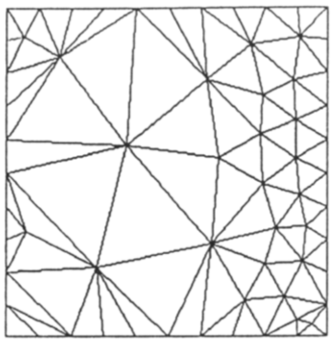
In the algorithm just presented, it is clear that the point spacing is controlled by Step (3b). The boundary point distribution is interpolated linearly throughout the field. If information is available from a flow calculation which indicates that refinement is required in a particular region of the field, it is necessary to convey that information to Step (3b). The mechanics of this are achieved by the use of the background mesh.

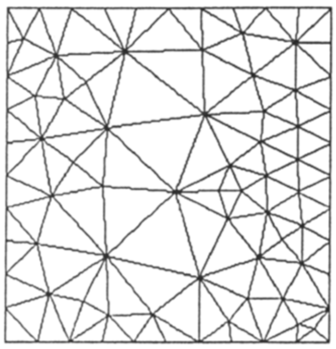
As a simple test case, take the rectangular domain shown in Figure 12a, in which mesh point clustering is required along the diagonal BD. An underlying mesh covering the domain is created, which at each node has a point distribution function specified. On creation of a prospective new node, the point distribution function for that node is obtained by interpolation of the distribution function from the background mesh. For the case in Figure 12, two triangles are sufficient to generate a mesh with the required point spacing. The distribution function at points B and D is defined to be small compared with those defined at points A and C. In this case, if a prospective point is generated close to the line BD, then the resulting interpolated value of the point distribution function will be small, resulting in a concentration of points along the diagonal BD. Hence, to implement the background mesh procedure, Step (3b) in the above algorithm is replaced by an interpolation of the point distribution function from the background mesh. The final result of the method for the rectangle is shown in Figure 12c.

In practice, the background mesh will be the previous mesh used for a flow simulation in which the features of the flow have been converted, by an appropriate transformation, to spacings in the physical space. The ideas expressed here can be also applied in three dimensions.

To show the flexibility of the approach, Figure 13a shows a background mesh in which the nodes which describe the letters 'CFD' have been assigned a small mesh point spacing. The resulting unstructured mesh derived using the automatic point creating algorithm is shown in Figure 13b.


(a)


(b)


(c)

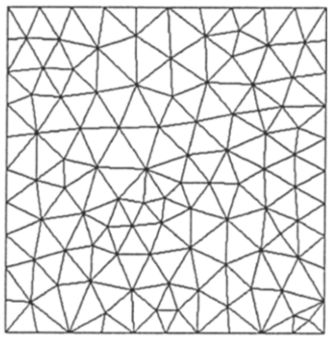
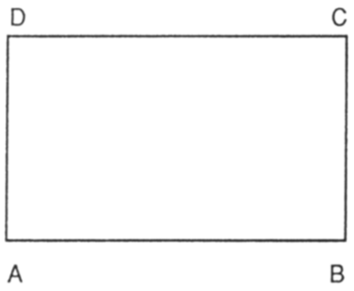
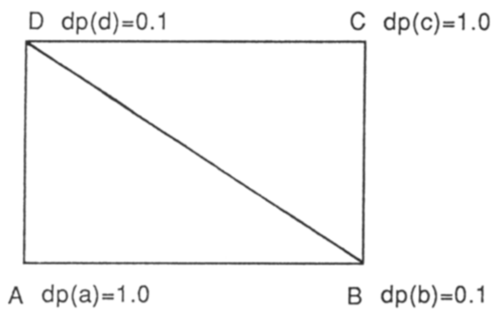

(d)

Figure 11. Automatic point generation. Shown is a sequence of grids during the point generation process.



(a) Rectangular domain ABCD.



(b) Background mesh consisting of two triangles. Point distribution function is specified at the nodes.

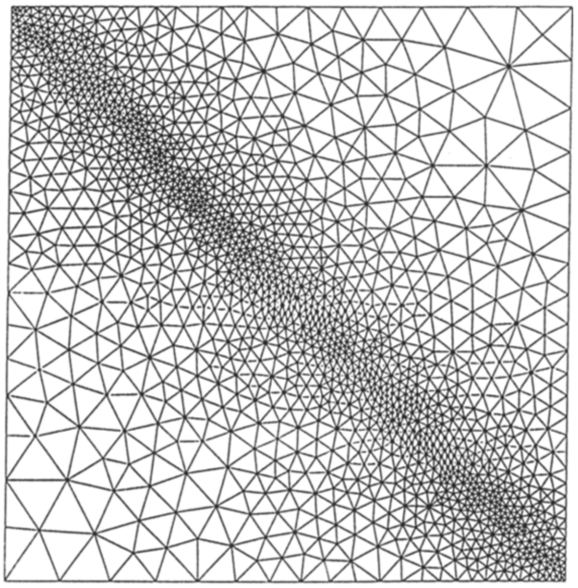
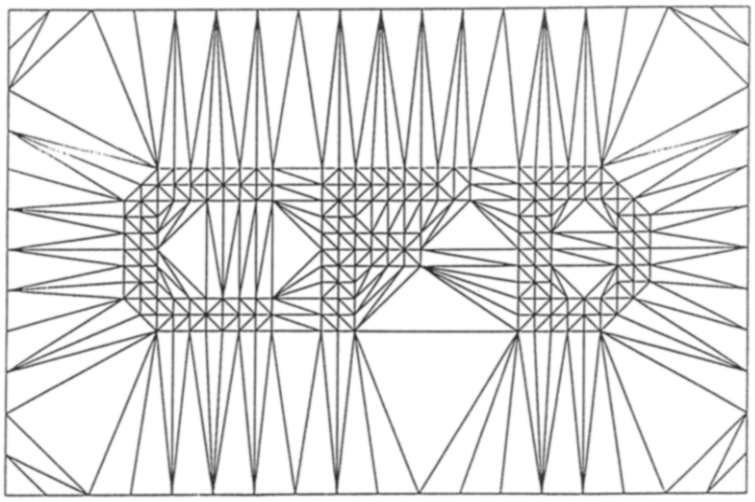

(c)

Figure 12. Generation of an adapted mesh using a background mesh.

Whichever method is used to generate the points, it is useful to smooth the mesh using a Laplacian filter. Using the data structure of the completed Voronoi diagram, it is possible to determine the points which define the polygon which encloses each point. Then each point is


(a)

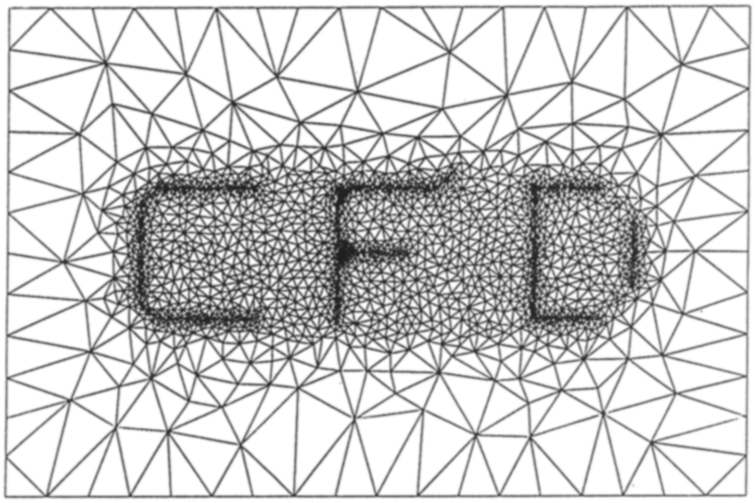

(b)

Figure 13. An example of the application of the background mesh.

smoothed N times according to

$$X_0^{n+1} = X_0^n + \omega \sum_{i=1}^{M} \frac{X_0 - X_i}{M},$$

where X_0^{n+1} is the new position of the point X_0^n, M is the number of neighboring points with coordinates X_i, and ω the relaxation parameter. The number of iterations should be within the range 20 to 50 and the relaxation parameter 0.05 to 0.25.

5. ASPECTS OF THE IMPLEMENTATION OF THE DELAUNAY TRIANGULATION

5.1. *The Convex Hull of Points*

The convex hull can be defined in a number of ways. It is only essential to define the hull such that it contains all the points which are to be triangulated. The way adopted here is to define 4 points (8 points, in 3 dimensions) approximating a quadrilateral (cuboid). This is subdivided into 2 triangles (5 tetrahedra) which results in 6 Voronoi vertices (17 Voronoi vertices), of which 4 (12) are null vertices since they lie outside the convex hull.

5.2. Point Insertion and Floating Point Accuracy

Apart from computing the radius of a circle (sphere), Step 3 of the Delaunay algorithm involves one of the few floating point operations in the algorithm to construct the Delaunay triangulation. In this step, it is necessary to find all the circles (spheres) within which the new point is within. Since the coordinates of the centre of every circle (sphere) in the Voronoi construction are stored in memory this would appear to be a simple computation. A point positioned at (x_p, y_p) is within a circle of radius R_i, centred at the Voronoi vertex V_i with coordinates (x_i, y_i), if

$$D_p = \left[(x_p - x_i)^2 + (y_p - y_i)^2\right] < R_i^2. \tag{5.1}$$

A similar formula applies in three dimensions. Normally, such a computation can be performed without problems, but if $|D_p - R_i^2|$ is the order of the round-off error of the computer, it is likely that an error can arise. Such an error may lead to an incorrect rejection or acceptance of a point without making a compensatory error in the immediate environment. This leads to a structural error in the Voronoi data, such that, on the introduction of another point in the vicinity, the hole created is disjoint. Such an error can be tested for in the Delaunay construction. However, once an error of this type has been committed it is difficult to correct and the triangulation process breaks down. A number of authors have analysed this problem and Baker [15] has determined a condition which must be satisfied on the magnitude of the round off error in the computer for a given typical length scale associated with the data points. In practice, it is worthwhile to perform the floating point operations in double precision.

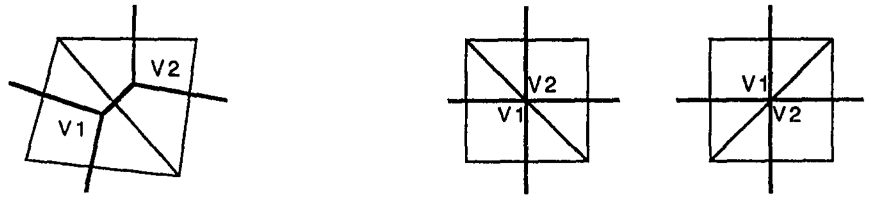
It is a very simple procedure to test this type of error within a Delaunay triangulation algorithm. If the test (5.1) is implemented as

$$|D_p - R_i^2| < \text{Err},$$

where Err is a variable defined in the computer program and represents the tolerance of the machine accuracy, then setting Err too large implies that this test is not sufficiently well defined and inconsistencies will frequently arise. Reducing Err to an appropriate value, consistent with the machine accuracy in double precision will largely alleviate this problem. From experience on a 32-bit word machine, for general point distributions, no problems have been encountered.

5.3. Degeneracies

Degeneracies arise in the Delaunay triangulation. One of the most common forms is when points are distributed in a regular manner. For example, in two dimensions, a degeneracy in the triangulation occurs if four points lie on a circle. In this case, the two Voronoi vertices are coincident and the triangulation is not unique, since the connections of the interior diagonal could be either way (Figure 14).



(a) Voronoi vertices V_1 and V_2 distinct.

(b) Voronoi vertices V_1 and V_2 coincident. Triangulation not unique.

Figure 14. Degeneracy and non-uniqueness of Delaunay triangulation for a regular distribution of points.

A similar degeneracy occurs in higher dimensions. In three dimensions this occurs if five or more points are on the surface of a sphere.

This degeneracy does not require any special treatment within the algorithm. The implication of the degeneracy is, however, more related to the floating point accuracy which has previously been discussed. Clearly, for a degenerate case, if a point is added, it is essential that the decision as to whether the point is inside or outside the circle (sphere) must be consistent for the two or more coincident Voronoi vertices.

One other form of numerical degeneracy which can occur is worthy of note. In the triangulation process, it is necessary to determine the centre of the circle (sphere) which passes through the three (four) points which form the associated triangle (tetrahedra). This calculation can be performed in a number of ways, but these usually involve the solution of two (three) linear equations for the unknown coordinates of the centre of the circle (sphere). If the three (four) points of a triangle (tetrahedra) are colinear, then this system of equations is singular and the effective centre of the circle (sphere) is at infinity. (If Cramer's rule were to be used to solve the two (three) equations, the singular nature of the system would be immediately obvious because the determinant of the coefficient matrix would be computed to be zero.) If this is the case, it has been found expedient to temporarily reject the point and save it for insertion at a later stage of the triangulation. Such degeneracies arise because of the particular spatial distribution of points, the order of insertion and the accuracy of the computer. However, the save and retry method works well.

5.4. Search Routines

The efficiency of the Delaunay algorithm described is dependent upon the use of fast search algorithms. In particular, routines which can efficiently perform the following operations.

(a) Within the boundary integrity procedure for three dimensions:
- (i) Derive boundary edges from the boundary surface triangulation.
- (ii) Check that boundary edges and faces are contained in the Delaunay triangulation.

(b) Within the Delaunay algorithm:
- (i) Determine the circles (spheres) within which an added point lies.

The key aspect of ensuring that such operations can be performed efficiently is the use of the inherent data structure, which has already been described, of the Delaunay construction. Of particular value is the information relating to the neighboring vertices of each Voronoi vertex. Furthermore, the points which define a triangle (tetrahedra) associated with a Voronoi vertex are also known. Hence, using this data structure it is possible to 'walk' through the Voronoi construction identifying for example, neighboring nodes and triangles (tetrahedra), etc. Figure 15 shows a Voronoi diagram for a two-dimensional construction and the equivalent local data structure.

The speed at which the boundary edges and surface faces are checked can be increased by firstly defining a vector which, for each node, assigns a Voronoi vertex of which one of its forming points is the node in question. In this way, when an edge is checked no search is required to find the region in the domain which contains the point and subsequent edges. Having once found a vertex, it is then necessary to perform a local tree search using the data structure inherent to the Delaunay construction, and illustrated in Figure 15.

The efficiency with which the Voronoi vertices to be deleted can be found, on the introduction of a new point, is largely dependent upon the time taken to find any vertex to be deleted. This time can be minimised if the points are firstly ordered such that contiguous points in the list are neighbors in physical space. If this is the case, then it is appropriate, when finding a circle (sphere) within which a point lies, to test the circles (spheres) created with the addition of the previous point. Once a circle (sphere), together with its Voronoi vertex has been found, then it is possible to utilise the tree data structure of the Voronoi construction.

On finding a vertex to be deleted, a list is created of its neighboring vertices. These are then tested and, if one is to be deleted, then the neighboring vertices of that vertex are added to the list of vertices to be checked. If the point is outside the circle (sphere) associated with a vertex, then its neighbors are not added to the list. When the list is empty, then all relevant circles (spheres) have been checked. The faces of the empty region created by the deletion of all triangles (tetrahedra) associated with the deleted vertices form the empty polygon (polyhedron) which contains the new point. The new Voronoi structure and Delaunay triangulation is then formed by connecting the new point to all the points on the edge (surface) of the polygon (polyhedron).

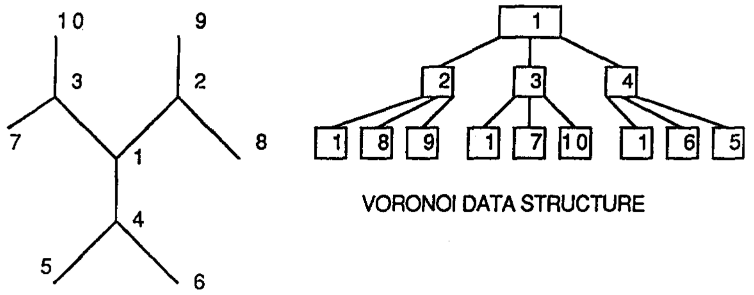


Figure 15. Voronoi vertices and data structure illustrating a local 'walk' around vertex 1.

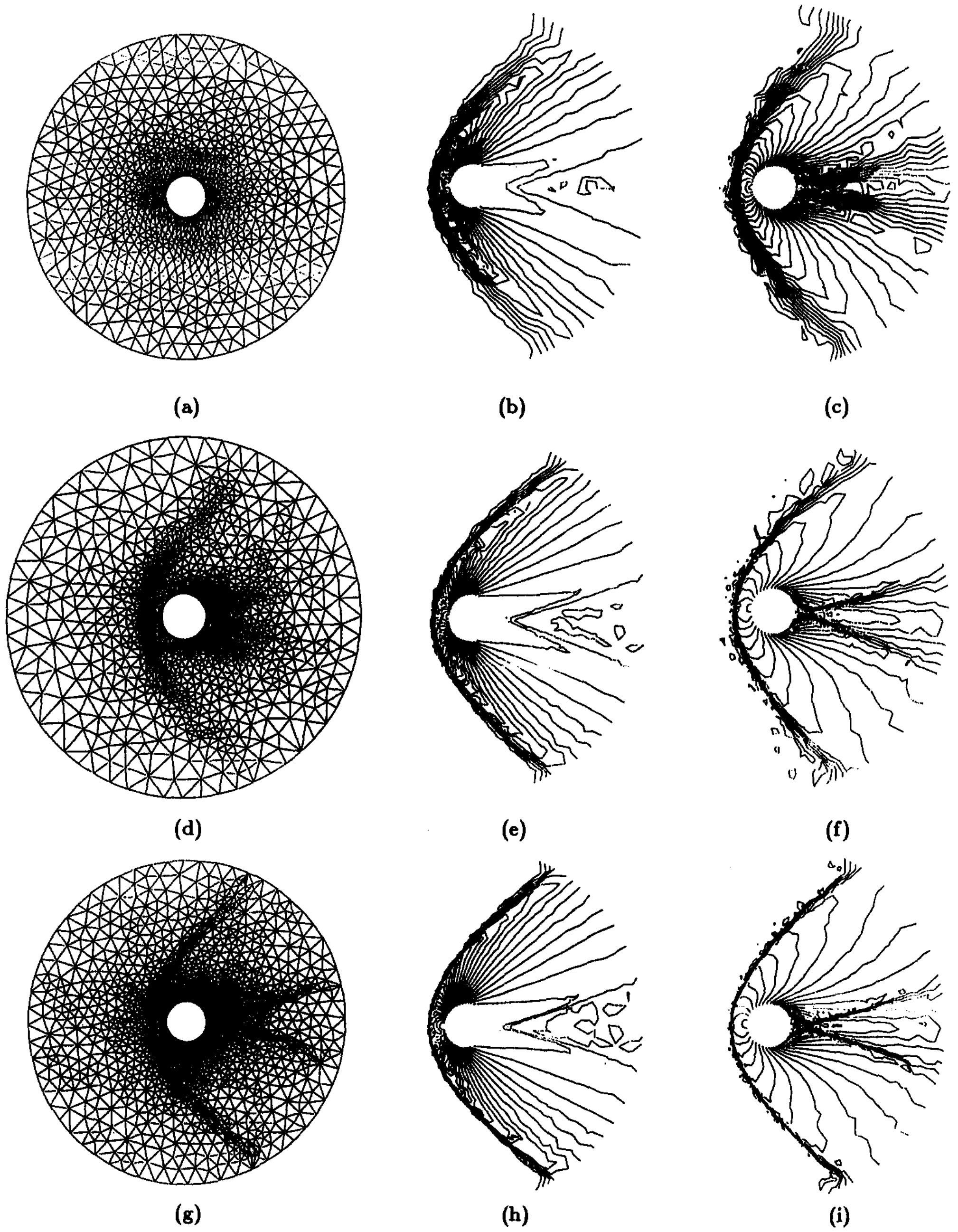


Figure 16. Mach 3 flow around a cylinder. Shown is the initial grid and flow solution together with subsequent grids and flow solutions using mesh adaption.

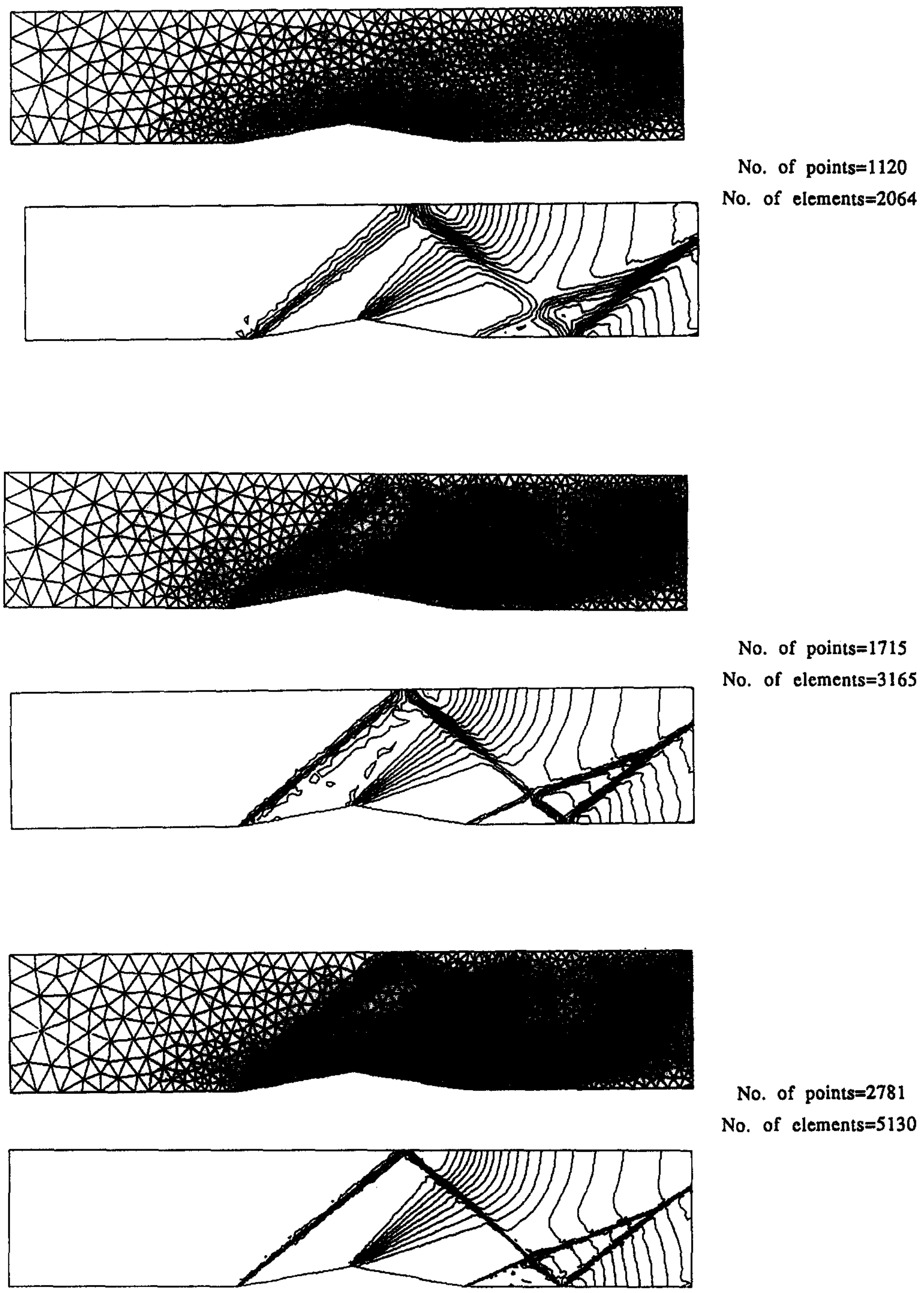


Figure 17. Mach 2 flow over a wedge. Shown is the initial grid and flow solution together with subsequent grids and flow solutions using mesh adaption.

6. APPLICATIONS

To illustrate the use of these ideas, some examples of the use of unstructured grids in computational fluid dynamics are given for realistic problems. In every case, the flowfield has been simulated using a multistage time stepping algorithm for the solution of the Euler equations [21].

Figures 16 and 17 show a sequence of grids and flow solutions for the Mach 3 flow around a cylinder, and the Mach 2 flow over a wedge using the Euler equations. In these examples, a combination of the use of a background mesh and local point enrichment has been used together with an adaptivity criterion based upon pressure to adapt the mesh to obtain a better resolution of the flowfield.

Finally, Figure 18 shows the Mach number contours on the surface of the double ellipsoid (Figure 9). This is an example which illustrates the use of the ideas described to three-dimensional flow problems.

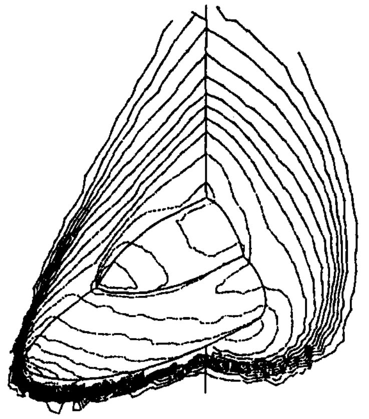


Figure 18. Mach number contours for the hypersonic flow around a double ellipsoid.

7. CONCLUSIONS

A method of constructing unstructured meshes from an arbitrary set of points in both two and three dimensions has been described. The geometrical criterion on which the method is based is due to Delaunay. To apply this approach to construct unstructured meshes which conform to a given boundary, it is necessary to implement algorithms which ensure boundary integrity of the mesh. Two methods have been discussed, each of which is applicable in two and three dimensions. Point generation strategies have also been discussed. These methods can be used to generate points consistent with local boundary point spacing, or to be consistent with a background mesh which could have been derived from a flow simulation.

REFERENCES

1. J. Hauser and C. Taylor (Editors), *Numerical Grid Generation in CFD,* Pineridge Press, Swansea, (1986).
2. J. Hauser, S. Sengupta, P. Eiseman and J.F. Thompson (Editors), *Numerical Grid Generation in CFD,* Pineridge Press, Swansea, (1988).
3. N.P. Weatherill, Grid generation in CFD, In *VKI Lecture Series on Grid Generation,* Von Karman Institute, Brussels, (June 1990).
4. N.P. Weatherill and C.R. Forsey, Grid generation and flow calculation for aircraft geometries, *J. of Aircraft* **22** (10), 855–860 (October 1985).
5. J.A. Shaw, J.M. Georgala and N.P. Weatherill, The construction of component adaptive grids for aerodynamic geometries, In *Numerical Grid Generation in CFD,* (Edited by J. Hauser, S. Sengupta, P. Eiseman and J.F. Thompson), Pineridge Press, Swansea, (1988).
6. J.F. Thompson, A general 3-D elliptic grid generation system on a composite block structure, *Computer Methods in Applied Mechanics and Engineering* **64**, 377–411 (1987).
7. M.S. Sheppard, Approaches to the automatic generation and control of finite element meshes, *Appl. Mech. Rev.* 41 (4), 169–185 (April 1988).
8. J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, Adaptive remeshing for compressible flow computations, *J. Comp. Physics* **72** (2), 449–466 (October 1987).
9. G.L. Dirichlet, Uber die Reduction der positiven quadratschen Formen mit drei undestimmten ganzen Zahlen, *Z. Reine Angew. Math.* **40** (3), 209–227 (1850).
10. B. Delaunay, Sur la sphere vide, *Izvestia Akademii Navk SSSR, Mathematical and Natural Sciences Division,* (6), 793–800 (1934).
11. F.P. Preparata and M.I. Shamos, *Computational Geometry,* Texts and Monographs in Computer Science, Springer-Verlag, Secaucus, NJ.
12. A. Bowyer, Computing Dirichlet tessellations, *The Computer Journal* **24** (2), 162–166 (1981).
13. N.P. Weatherill, The generation of unstructured grids using Dirichlet tessellations, Princeton University, MAE Report No. 1715 (July 1985).
14. P.L. George, F. Hecht and E. Saltel, Constraint of the boundary and automatic mesh generation, In *Numerical Grid Generation in CFD,* (Edited by J. Hauser, S. Sengupta, P. Eiseman and J.F. Thompson), Pineridge Press, Swansea, (1988).

15. T.J. Baker, 3-dimensional mesh generation by triangulation of arbitrary point sets, In *Proc. of the AIAA 8th CFD Conference*, Hawaii, (June 1987).
16. N.P. Weatherill, The integrity of geometrical boundaries in 2 dimensional Delaunay triangulation, *Comm. in Applied Numerical Methods* **6**, 101–109 (1990).
17. J. Peraire, A finite element method for convection dominated flows, Ph.D. Thesis, University of Wales, (1986).
18. W.J. Schroeder and M.S. Shephard, Geometry-based fully automatic mesh generation and the Delaunay triangulation, *Int. J. Numerical Methods in Engineering* **26**, 2503–2515 (1988).
19. J. Peraire, K. Morgan and J. Peiro, Unstructured finite element mesh generation and adaptive procedures for CFD, In *AGARD Conference, Proceedings No. 464, Applications of Mesh Generation to Complex 3-D Configurations*, AGARD-CP-464, (May 1989).
20. N.P. Weatherill, A method for generating irregular computational grids in multiply connected planar domains, *Int. J. Num. Methods in Fluids* **8**, 181–197 (1988).
21. A. Jameson, T.J. Baker and N.P. Weatherill, Calculation of inviscid transonic flow over a complete aircraft, Presented at the *24th AIAA Aerospace Sciences Meeting*, Reno, NV, AIAA Paper 86-0103, (1986).