INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 8, 181–197 (1988)

A METHOD FOR GENERATING IRREGULAR COMPUTATIONAL GRIDS IN MULTIPLY CONNECTED PLANAR DOMAINS

N. P. WEATHERILL*

Aircraft Research Association, Manton Lane, Bedford, U.K.

SUMMARY

A method for generating irregular triangular computational grids in two-dimensional multiply connected domains is described. A set of points around each body is defined using a simple grid generation technique appropriate to the geometry of each body. The Voronoi regions associated with the resulting global point distribution are constructed from which the Delaunay triangulation of the set of points is thus obtained. The definition of Voronoi regions ensures that the triangulation produces triangles of reasonable aspect ratios given a grid point distribution. The approach readily accommodates local clustering of grid points to facilitate variable resolution of the domain. The technique is generally applicable and has been used with success in computing triangular grids in multiply connected planar domains. The suitability of such grids for flow calculations is demonstrated using a finite element method for solution of the inviscid transonic flow over two-dimensional high-lift aerofoil configurations.

KEY WORDS Grid generation Triangles Unstructured

INTRODUCTION

Many problems in computational physics involve the numerical solution of a set of equations in a complicated shaped domain. The solution of such problems requires the domain to be discretized to produce a set of points on which the numerical algorithm can be based. For some problems, the generation of a suitable grid can be as demanding as the effort required to perform the computations for which the grid was intended. In recent times, considerable attention has been focused on the discretization process, which is commonly called grid generation. Many grid generation algorithms of varying degrees of automation and sophistication have appeared in the literature.[1,2] However, the complexity of many of these algorithms results in a continued search for more simple and, in some cases, more elegant grid generation techniques.

For the ease and accuracy of implementing boundary conditions, body conforming grids have received widespread use. Thacker[3] reviews some of the many different techniques which have been developed. For simple shapes, it is possible to generate rectilinear meshes without too much difficulty. For complicated shapes, however, it becomes increasingly difficult to produce a structured mesh that is aligned with all solid boundaries. Multiblock methods, in which the computational domain is subdivided into a set of blocks whereupon the arrangement of the blocks defines the grid topology which is consistent with the boundary shape, have been proposed for

* Present address: Institute for Numerical Methods in Engineering, University College of Swansea, Singleton Park, Swansea SA2 8PP, U.K.

0271–2091/88/020181–17$08.50

Received 27 October 1986
Revised 13 April 1987

dealing with such regions.[4] For very complex shapes, for example, a transport aircraft with closely coupled nacelles, it is questionable whether it is possible to automatically generate a structured grid and maintain low cell skewness, a smooth point distribution and adequate grid resolution of important regions.

An alternative grid generation procedure is to use tetrahedral or triangular cells leading to an unstructured grid which can be adapted to irregular boundaries. A smooth distribution of triangle size can be achieved without distortion of the elements. The unstructured triangle approach readily accommodates local clustering of grid points to facilitate variable resolution of a domain.

Finite element methods have been traditionally based on triangular cells in two dimensions and tetrahedral cells in three dimensions. It follows, therefore, that numerous algorithms have been developed to generate triangular grids within a given domain. Lo[5] categorizes and references some of these methods.

In this paper, a new method will be described for generating triangular grids which connects an arbitrary cluster of points by a systematic procedure based on the Delaunay criterion. This is dual to the Voronoi diagram (or Dirichlet tessellation) that results from a division of the domain into polygonal neighbourhoods, each consisting of the subdomain of points nearer to a given grid point than any other grid point. Although the Delaunay triangulation and associated Voronoi diagram have been exploited by others as a natural setting for calculations involving irregular spaced points,[6,7] we believe that the use of the Delaunay criterion as an explicit method of generating grids for complex shapes is a new departure. We will show how these ideas can be applied to multiply connected domains in two dimensions with an emphasis on aeronautical geometries. The suitability of the triangular grids for flow calculations will be demonstrated by computing the inviscid transonic flow over a two-dimensional high-lift aerofoil configuration using a new finite element algorithm for the Euler equations developed by Jameson.[8] Although the discussions will be restricted to two dimensions, the ideas extend to three dimensions and have been applied with success to compute the inviscid transonic flow over a complete aircraft.[9]

VORONOI NEIGHBOURHOODS AND DELAUNAY TRIANGULATION

Dirichlet[10] in 1850 proposed a method whereby a given domain could be systematically decomposed into a set of packed convex polygons. Given a set of points in the plane, the Dirichlet tessellation is the construct which assigns to each point a territory that is the area of the plane closer to that point than to any other point in the set. 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.

This basic idea, although first described by Dirichlet, has been subsequently rediscovered by various workers in many fields and the Dirichlet regions are described under a variety of names: Theissen regions (meteorology), Wigner–Seitz cells (solid state physics) and Voronoi diagrams (computational geometry). An extensive discussion on the application of Voronoi diagrams is given in Green and Sibson.[11]

If a set of points is denoted by $\{p_i\}$, then the Voronoi region $\{V_i\}$ can be defined as

$$\{V_i\} = \{p: \|p - p_i\| < \|p - p_j\|, \quad \forall j \neq i\};$$

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

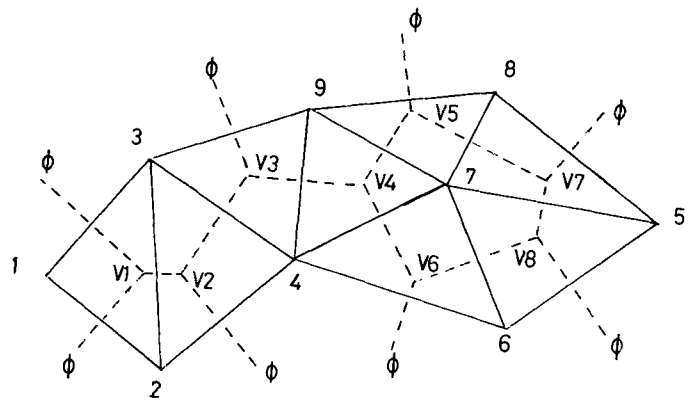


Figure 1. Dirichlet regions and associated Delaunay triangles of the set of points i, $i = 1$–9

pairs which have some segment of boundary in common are joined by straight lines, the result is a triangulation of the convex hull of the data points. This triangulation is known as the Delaunay triangulation of the set of points $\{p_i\}$ and forms the basis of our grid generation procedure. It is apparent that for each triangle there is an associated vertex of the Voronoi diagram which is at the circumcentre of the three points which form the triangle. In other words, each Delaunay triangle has a unique vertex of the Voronoi diagram and no other vertex within the Voronoi structure lies within the circle centred at this vertex. Figure 1 shows a schematic of a Voronoi diagram and associated Delaunay triangulation of a small set of points. In three dimensions, the territory of each data point is a convex polyhedron and each vertex of the Voronoi structure is at the circumcentre of a sphere defined by four points which describe the Delaunay tetrahedron.

A survey of computational methods to construct the Voronoi diagram and Delaunay triangulation is given in the paper by Green and Sibson.[11] Recent noteworthy papers which describe efficient algorithms are those of Watson,[12] Boywer[13] and Fortune.[14] The algorithm used here is based on the work of Boywer.[13]

DELAUNAY TRIANGULATION ALGORITHM

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 forming points of a Voronoi vertex (the forming points define a Delaunay triangle) and a list of the neighbouring Voronoi vertices. In n dimensions, each vertex has $n + 1$ forming points and $n + 1$ neighbouring vertices. As an example, Table I contains the vertex structure for the points shown in Figure 1.

The Delaunay algorithm is a sequential process; each new point is introduced into the existing structure, which is broken and then reconnected to form a new Delaunay triangulation.

The details of the algorithm will be presented in a step-by-step manner for a triangulation in two dimensions.

Step 1

Define the convex hull within which all points will lie. Specify four points together with the associated Voronoi diagram structure. Figure 2 indicates a typical start-up layout and Table II the corresponding data structure.

Table I

Vertex	Forming points	Neighbouring vertices
1	1 2 3	2 ϕ ϕ
2	2 3 4	1 3 ϕ
3	3 4 9	2 4 ϕ
4	4 7 9	3 5 6
5	7 8 9	4 7 ϕ
6	4 7 6	4 8 ϕ
7	5 7 8	5 8 ϕ
8	5 6 7	6 7 ϕ

Table II. Data structure for the triangulation shown in Figure 2

Vertex Voronoi diagram	Forming points	Neighbouring vertices
1	−10 1 3	−10 −10 5
2	−10 3 4	−10 −10 6
3	−10 2 4	−10 −10 6
4	−10 1 2	−10 −10 5
5	1 2 3	1 4 6
6	4 2 3	5 2 3

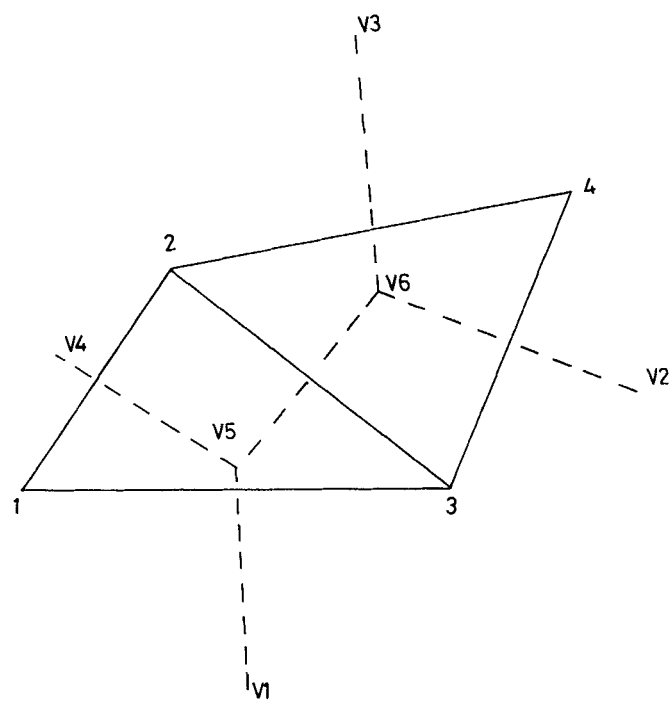


Figure 2. Initial points which define convex hull with associated vertices

Vertices 1, 2, 3, 4 are not strictly defined, since they lie outside the convex hull and therefore do not possess three forming points and do not have three neighbouring vertices. Default values of -10 for the missing forming points and neighbouring vertices have been used to indicate that these are null vertices.

Step 2

Introduce the new point anywhere within the convex hull $\{1, 2, 3, 4\}$.

Step 3

Determine all vertices of the Voronoi diagram to be deleted. A point which lies within the circle, centred at a vertex of the Voronoi diagram and which passes through its three forming points, results in the deletion of that vertex. This follows from the definition of a Voronoi polygon.

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 neighbouring Voronoi vertices to the deleted vertices which have not themselves been deleted. These data provide the necessary information to enable valid combinations of 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 two other points which are contiguous to the new point and form an edge of a neighbouring triangle (these are the possible combinations obtained from Step 5).

Step 7

Determine the neighbouring 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 neighbouring vertices as found in Step 5 to identify common pairs of forming points. When a common combination occurs, then the two associated vertices are neighbours 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 new point.

With the exception of Step 3, all the operations described are of a local nature and can be carried out in an operation count independent of the total number of points within the current structure. A naive search for vertices of the Voronoi diagram to be deleted would amount to $O(N)$ operations per point and thus to $O(N^2)$ operations for the total workload. An obvious improvement is to identify any one vertex that is to be deleted and then perform a tree search through the neighbouring Voronoi diagram data structure checking for other vertices to be deleted; i.e., (a) find a vertex to be deleted; (b) test each of that vertex's neighbouring vertices for deletion; (c) when all neighbouring vertices of deleted vertices have been checked, then all vertices to be deleted must have been identified. The effectiveness of this approach depends on the efficiency with which the first vertex to be deleted can be found. If the list of points is ordered in a systematic manner, then a reasonable starting vertex is the last vertex in the structure. This procedure can lead to a negligible workload for Step 3.

It is apparent that two floating point calculations are required in this method; the co-ordinates of the circumcentre (vertex) and the squared radius associated with the circle that passes through the three forming points of that vertex. The remaining operations are of a purely logical nature involving the manipulation of integer variables.

The specification of four initial points which define the convex hull within which all points lie is not, in practice, a restriction. These points can be chosen to ensure a sufficiently large hull so that no restrictions on the grid generators will occur.

It is apparent from the definition of a Voronoi polygon that problems can arise in the triangulation procedure when certain degeneracies occur in the data. Common degeneracies in two dimensions arise when

(1) two points are coincident,
(2) three points of a potential triangle lie on a straight line,
(3) four or more points are cyclic.

The first degeneracy is easily avoided by requiring that each new point be some small distance from its nearest neighbours before it is admitted into Step 2. The second degeneracy implies that the circumcentre of the three forming points is at infinity. This is readily detected and in such a case the new point which has caused such a degeneracy can be rejected or moved a small distance and treated as a new point.

The third degeneracy is of a more subtle nature than the preceding two. In this case, two vertices are coincident and give rise to a point at which more than three Voronoi polygons meet. In such a situation, the resulting triangulation is not unique, but this in itself is not a problem. The triangulation procedure is still valid for such cases. However, this degeneracy may manifest itself at a later stage of the triangulation when a point is introduced which results in the deletion of one of the vertices of the Voronoi diagram but not the other. An inconsistency thereby arises in the vertex structure which can result in the breakdown of the non-overlapping triangulation procedure. This problem occurs because of round-off error within the computer. It proves advantageous to check for cyclic points so that this subtle degeneracy cannot arise. If such points do occur, then one point can be rejected or moved a small distance. Apart from points which define the aerodynamic geometry, the actual position of the grid points is not critical and it has been found that degeneracies can be avoided by moving points a very small distance from their original position.

Computationally, the usual compromises between storage space and execution time have been observed. In addition to the forming points and neighbouring vertices of each vertex of the Voronoi diagram, the co-ordinates of each vertex and each point are stored together with the squared distance of the vertex from its forming points. Figure 3 shows how CPU time varies

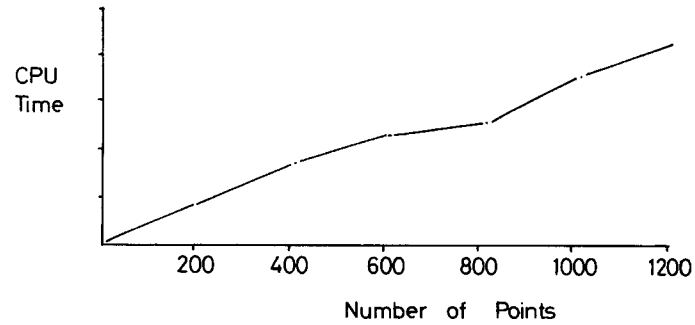


Figure 3. Variation of CPU time with number of points for the two-dimensional Delaunay triangulation

with the number of points for the two-dimensional Delaunay triangulation algorithm.

APPLICATION TO MULTIPLY CONNECTED DOMAINS

The Delaunay triangulation provides a systematic way of constructing a triangular covering of a given set of points. The technique does not generate the position of grid points. The major applications of the Voronoi diagram, as reported in the literature, have been to the fields of social statistics and meteorology, where the data points have been obtained from an experiment or observation. Our application of this technique to grid generation is somewhat different. Here we wish to discretize a finite domain and ensure that the resulting triangular structure is body conforming, a sufficient number of cells are obtained in regions where a high resolution of the domain is required and the aspect ratio of cells is not too large.

To illustrate the application of the Delaunay triangulation concept to grid generation, we will consider the generation of a grid around an ellipse within an elliptical domain.

First, the outer boundary points followed by the points which define the elliptical section are introduced into the Delaunay algorithm. The resulting triangulation of these points gives rise to edges of triangles which define the ellipse and outer boundary. Figure 4 shows a typical case. Now suppose that other grid points from an elliptic distribution are added to this structure. It may arise that a point which lies close to the inner ellipse would involve a restructuring of the triangles, resulting in the edges of the triangles no longer forming a body conforming structure. Clearly this cannot be allowed to occur. To prevent this situation, a check is made following the introduction of the outer and inner boundary co-ordinates to find all vertices of the Voronoi diagram with all three forming points from the set of points which define the inner elliptical section. Such vertices are protected throughout the remaining triangulation. Any additional point which would result in the deletion of such a vertex of the Voronoi diagram and hence break the body conforming triangular structure is rejected. This criterion, which defines protected triangles, is applicable to convex shapes. For concave segments, however, a triangle could be formed which contains all body nodes but lies outside the body. It is necessary, therefore, to check for such an occurrence by computing the scalar product of the boundary normals at each of the nodes which define the triangle with the vectors formed between the nodes of the triangle and the circumcentre of the triangle (i.e., the unique vertex of the Voronoi region associated with the triangle). If any of the scalar products are positive, then the triangle lies exterior to the body and is not protected.

At the end of the triangulation process, when all grid points have been introduced, a clean-up procedure is required. The vertices of the Voronoi diagram which have been protected are deleted, thus removing triangles interior to the elliptical section. In addition, it is necessary to delete vertices

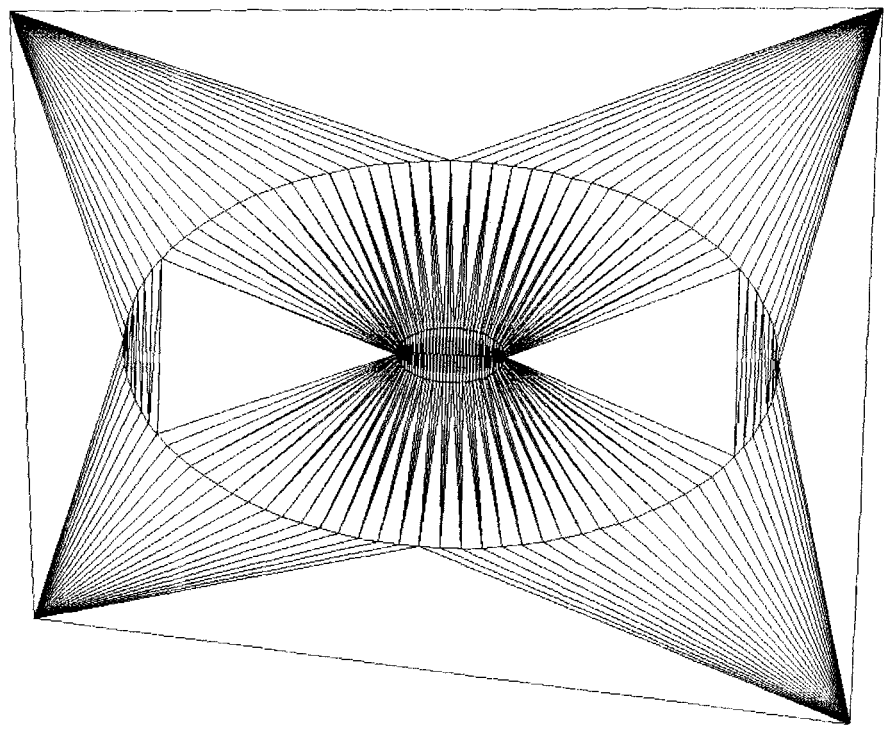


Figure 4. Triangulation of convex hull, outer boundary and inner boundary points

which have, as a forming point, any of the points which define the convex hull. The convex hull was defined for convenience and should not in any way define the outer boundary for computations. Figure 5 shows the Delaunay triangulation of a set of points which includes the convex hull, outer boundary, inner ellipse and interior grid points and Figure 6 this structure after the clean-up procedure. Figure 7 shows the Voronoi diagram superimposed onto the Delaunay triangulation.

The procedure outlines forms the basis for our grid generation technique. If the input points define a rectilinear grid, as was the case in the previous example, a more direct triangulation method could be employed, making the Delaunay approach excessively complicated. However, the generality of the Delaunay approach to grid generation can be readily extended by utilizing the concept of the protected vertices within the Voronoi diagram. Suppose a point which lies within the inner ellipse was added to the set of points shown in Figure 6. Clearly such a point would not be required, since the resulting triangulation about the point would lead to triangles interior to the geometry and ideally the point should be rejected by the algorithm. However, if such a point were introduced, it would result in an attempt to delete one or more of the protected vertices of the Voronoi diagram. But any such point is to be automatically rejected on the grounds that it may affect the body conforming structure of the triangles. This, therefore, is a method of detecting points which fall within the body section.

Strictly, using this criterion to reject points, a point is not accepted if it lies within the circle centred at a vertex whose forming points all lie on the inner section. This could also include points which lie outside the inner ellipse. However, if such a point was accepted, it would, in general, result in a number of highly skewed triangles close to the surface of the ellipse which would not be advantageous for the solution of the equation on the grid and hence it would seem reasonable to also reject such points.

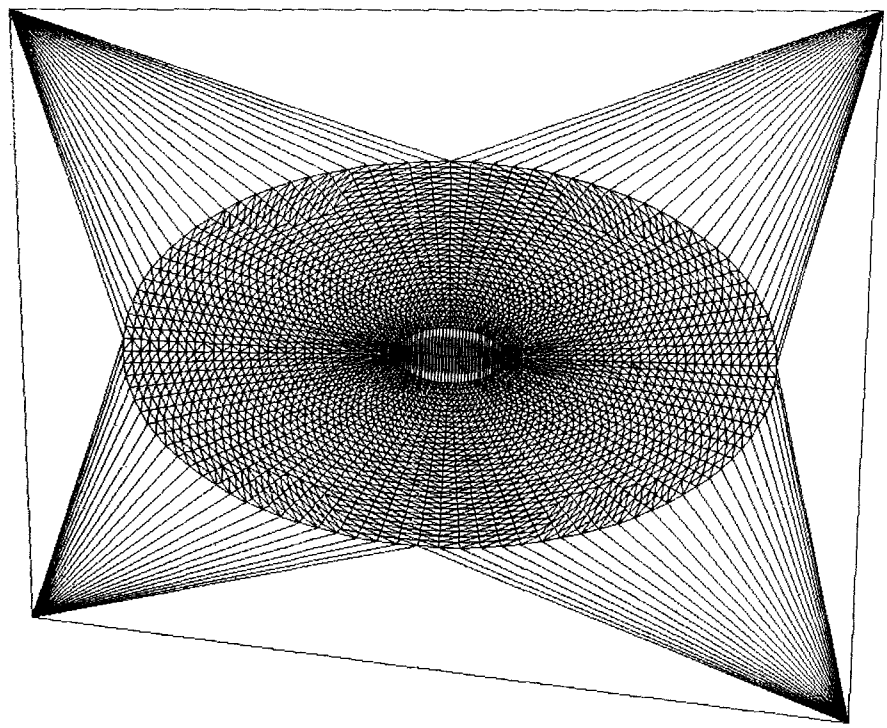


Figure 5. Triangulation of all data points

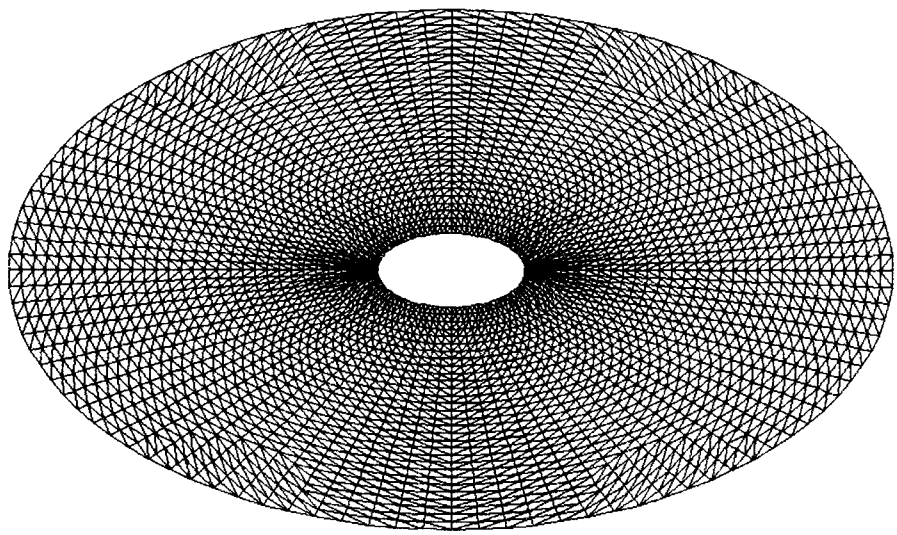
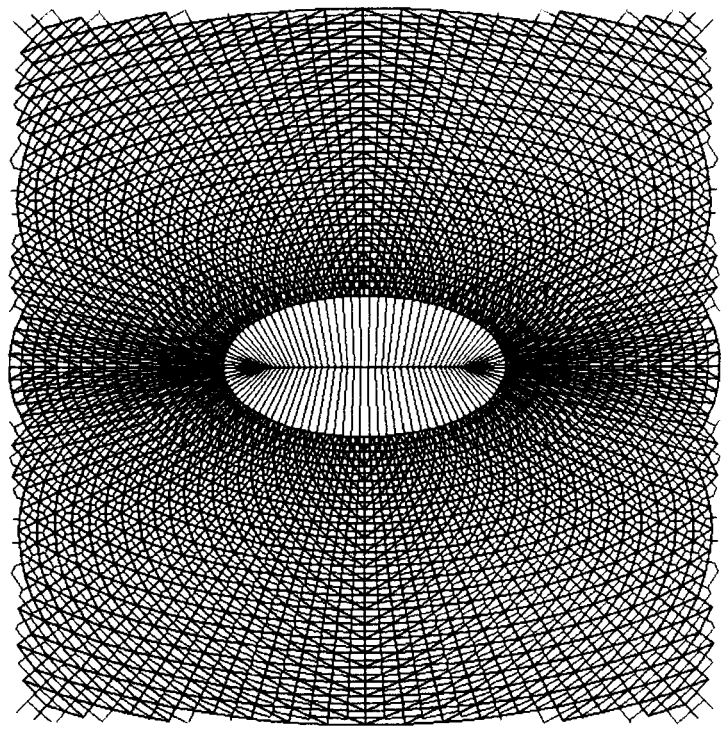


Figure 6. Triangulation after deletion of unwanted triangles



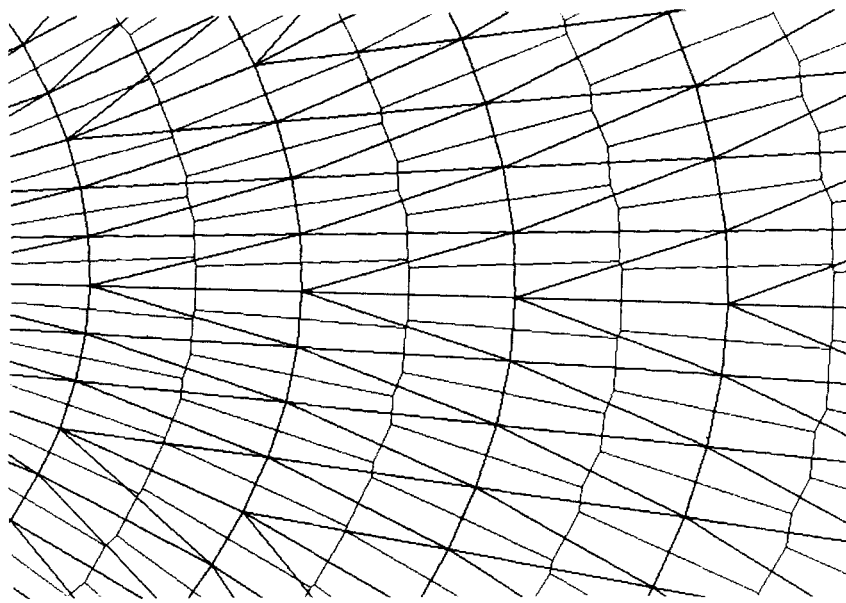


Figure 7. Delaunay triangulation and the associated Voronoi regions

There are situations, however, when it is desirable to generate points very close to the interior body sections. For example, a grid for a Navier–Stokes flow calculation would require points very close to the body. In such circumstances, a geometrical area check can be performed to determine whether a point is interior to a given body section. If a point attempts to delete a protected triangle, then the area of the protected triangle is compared with the sum of the areas of the three triangles formed by joining the point to each of the nodes of the protected triangle. If the summed area is greater than the area of the protected triangle, the point lies outside the triangle and is not rejected.

The method outlined to identify points interior to a given shape increases the applicability of our Delaunay approach to grid generation. The grid point generator does not have to produce a body conforming set of points which lie outside a given boundary. Any suitable set of points can be triangulated, with points interior to the inner boundary detected and rejected. In addition, the algorithm can be applied to more than one interior body. Following the approach already described, each set of points which defines an interior boundary is introduced, triangulated and the protected vertices for each closed boundary are noted. Any grid point which attempts to delete any protected vertex of the Voronoi diagram is rejected. As an example of these techniques, Figure 8 shows the resulting triangulation when a Cartesian grid is used to discretize the region around two aerodynamic vehicles.

Often the generation of a grid suitable for a particular body shape is fairly straightforward; a polar grid around a circle, an elliptical grid around an ellipse. It is possible to use these techniques to form the basis of a grid point generator for the Delaunay triangulation which results in high-quality triangular grids. For example, using a sequence of transformations, a grid around an aerofoil can be generated. Such a grid is shown in Figure 9 and the corresponding Delaunay triangulation is detailed in Figure 10. The grid shown in Figure 9 provides an excellent basic grid generator for multi-element aerofoil configurations. Using this '*O*' mesh generator, a grid is generated around one of the bodies in the configuration, the grid points covering the entire domain. For the other bodies, the analytic generator is used to produce points local to each section. The grid

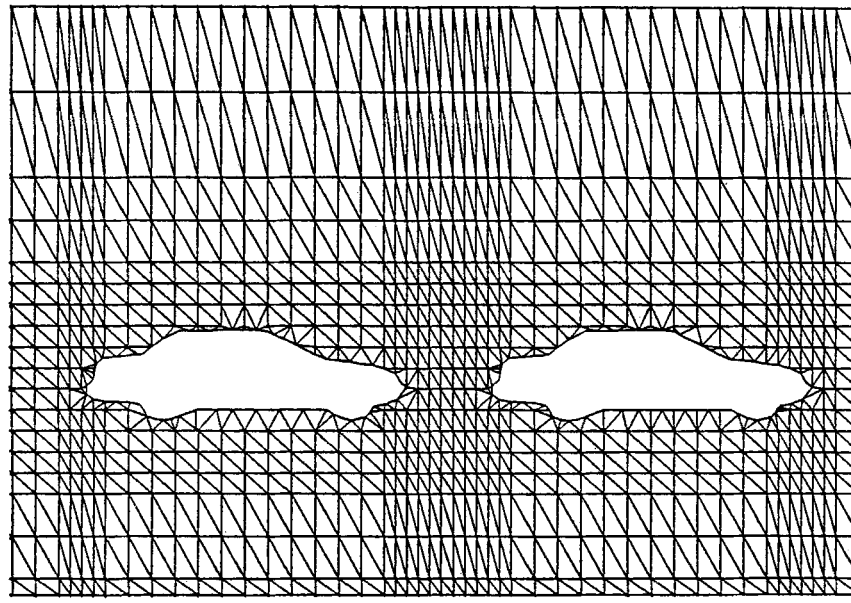


Figure 8. Unstructured triangular mesh for two automobiles

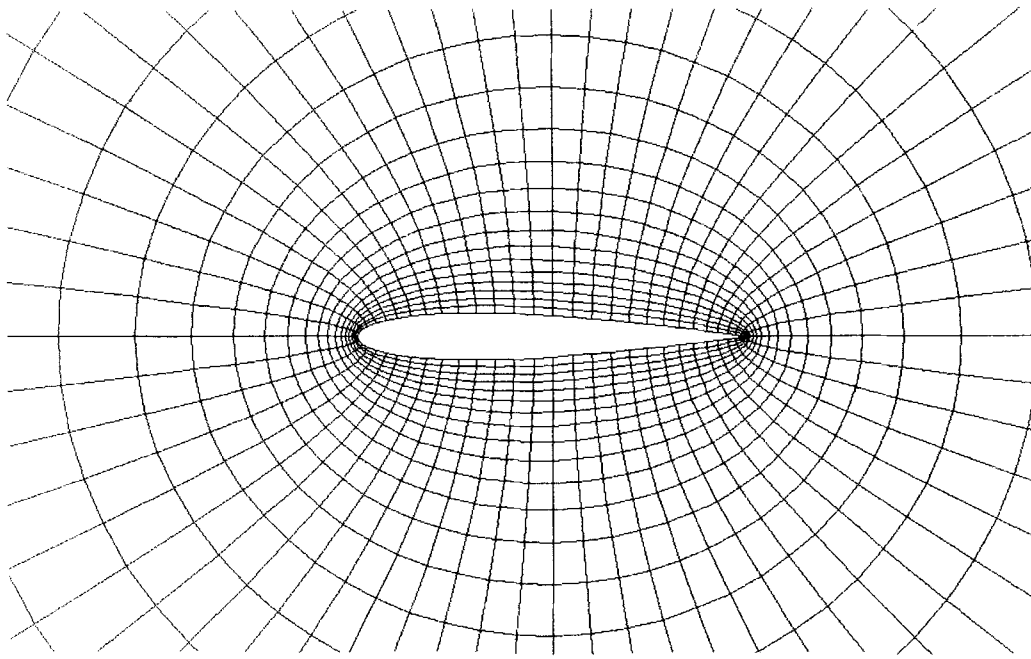


Figure 9. Regular quadrilateral mesh

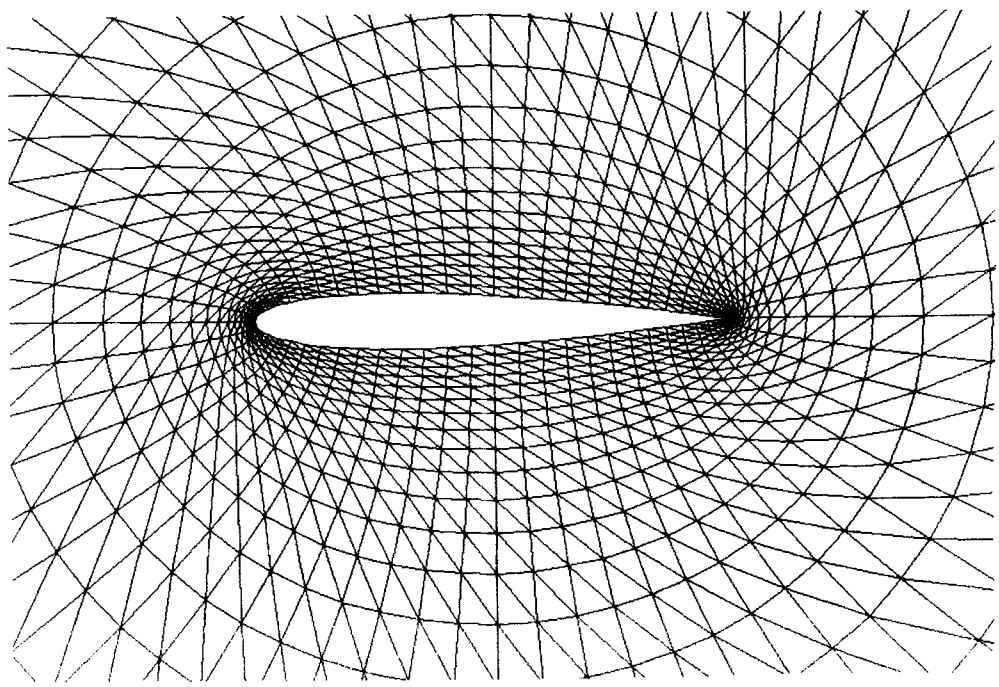


Figure 10. Regular triangulation of the grid points shown in Figure 9

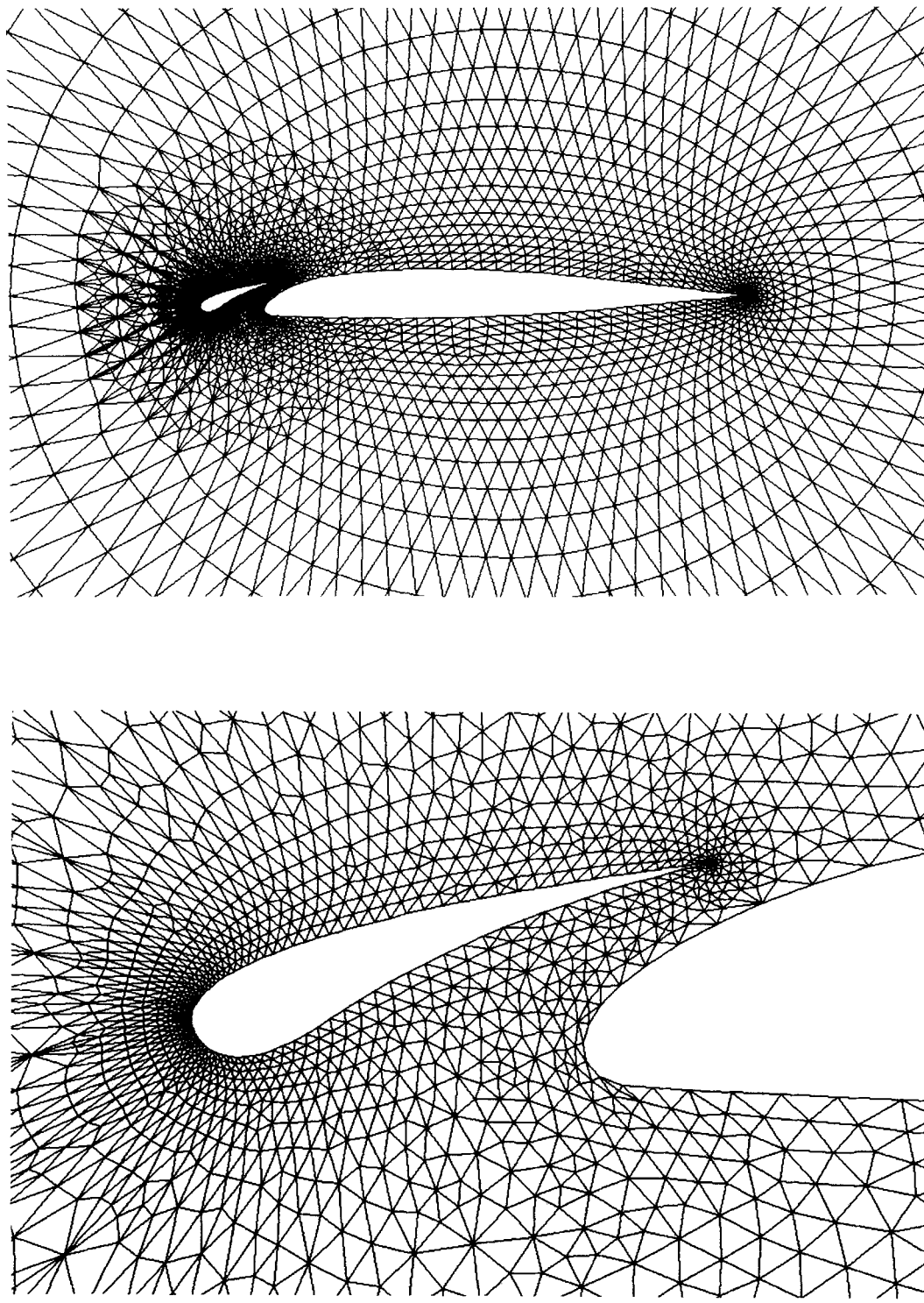


Figure 11. Triangular grid for a two-element high-lift configuration

points from one body which fall inside another body are rejected using the techniques already discussed. The resulting grid has an '*O*'-like structure local to each body, but in regions where the grids overlap the global grid is unstructured.

To decrease grid skewness in the overlapping regions, the triangulated grid can be smoothed 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. Each interior grid point is then smoothed N times according to

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

where X_0^{n+1} is the new position of the point X_0, m is the number of neighbouring points and ω is the relaxation parameter. The grid is thereby smooth, with desirable spacing local to each body. This approach is applicable to an unlimited number of bodies and provides an ideal grid for numerical calculations.

As an illustration of these techniques, a variety of examples are presented of triangular grids about complex two-dimensional configurations. Figure 11 shows a grid around an aerofoil with leading edge slat. The smoothing operator has removed any trace of grid structure within the slat region and regular-shaped triangles have resulted. Figure 12 shows an unstructured triangular grid around a four-element aerofoil configuration and Figure 13 the corresponding Voronoi diagram.

The Delaunay triangulation approach to grid generation is well suited for use with an interactive graphics workstation. The grid can be viewed after the introduction of any number of points. If for some reason additional points need to be added in a particular region, the user would be free to specify appropriate points which could then be incorporated into the grid without the need to restart the triangulation procedure.

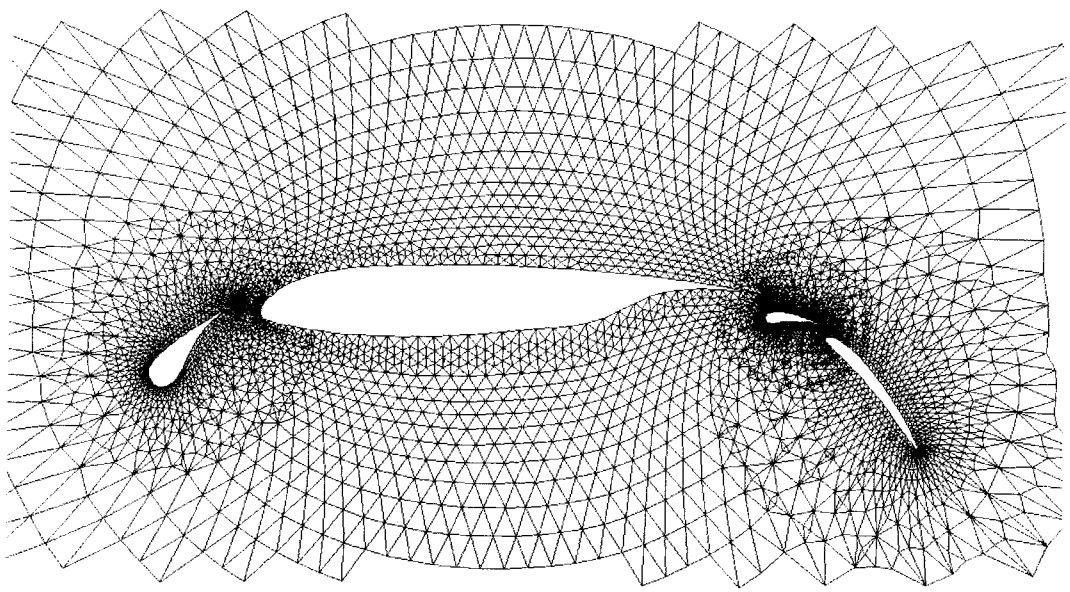


Figure 12. Triangular grid for a four-element high-lift configuration

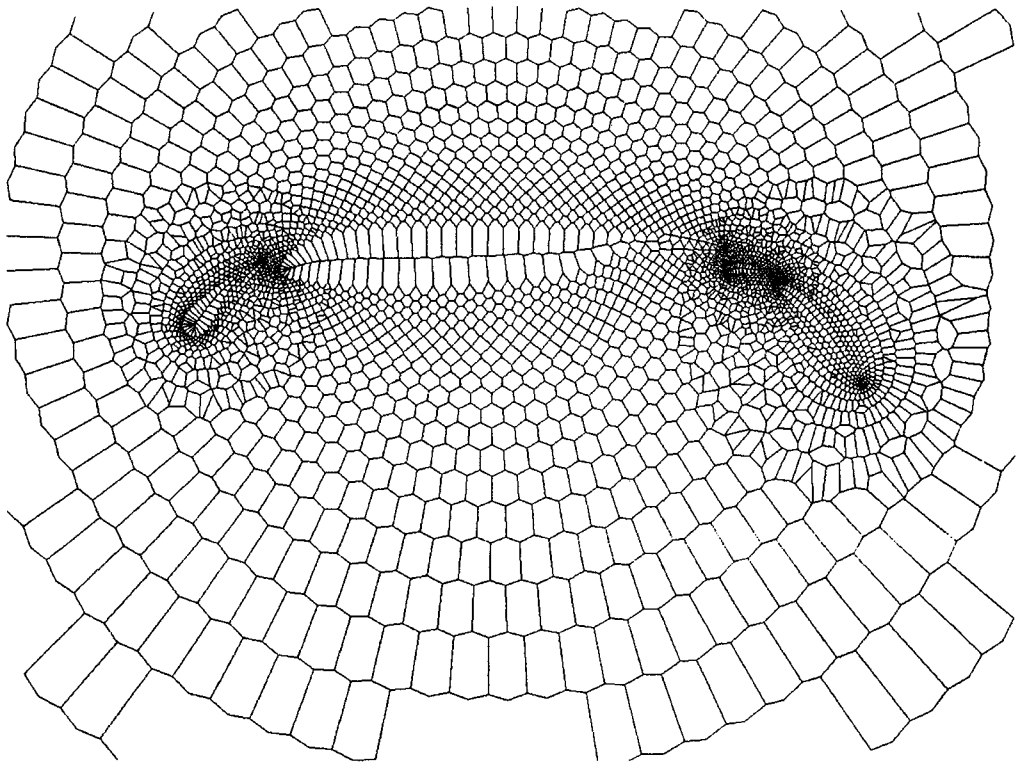


Figure 13. Voronoi diagram corresponding to the triangulation shown in Figure 12

Mesh enrichment is a natural technique to incorporate into this grid generation method. Points obtained by interpolation of the base grid are used to provide additional resolution of complex geometrical features. However, unlike mesh enrichment on a regular grid, the unstructured grid approach does not lead to any artificial interfaces where continuity of grid lines is lost.

It is evident from the foregoing comments that the Delaunay approach to grid generation could provide an effective technique for use with an adaptive mesh/flow algorithm. Some work in this area has already been reported.[15]

The finite element approximation to the Euler equations, as formulated by Jameson,[8] has been used with the Delaunay triangulation algorithm to compute the transonic flow over high-lift aerofoil geometries. The finite element formulation is obtained by directly approximating the integral equations for the balance of mass, momentum and energy in polygonal control volumes which are formed by the union of triangles meeting at a common node. The flux balancing can be broken down into contributions of fluxes across edges. This novel decomposition reduces the evaluation of the Euler equations to a single main loop over the edges of all triangles. It can be shown that the scheme can be reformulated so that it is essentially equivalent to the Galerkin method, but the present formulation leads to a substantial reduction of the computational effort.

Pressure distributions over the four-element high-lift configuration shown in Figure 12 are presented in Figure 14.

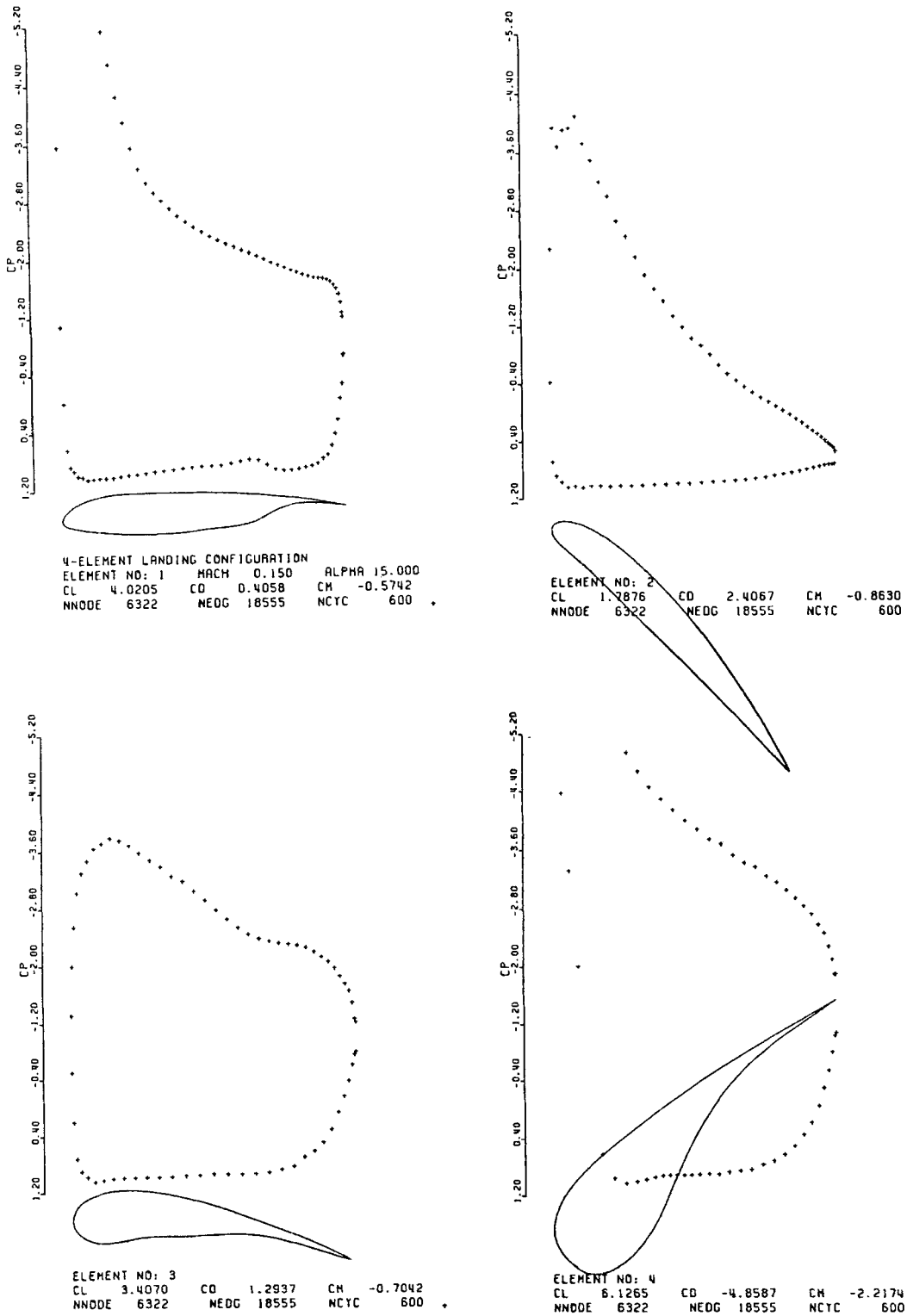


Figure 14. Pressure distribution computed using the Euler equations on the grid shown in Figure 12

CONCLUSIONS

A method of generating Voronoi regions and corresponding Delaunay triangles has been exploited to provide a means of connecting a set of points to produce a consistent triangular covering of a given domain. The set of points is generated using an approach consistent with the body geometries to ensure high grid quality. Examples given for complex two-dimensional geometries illustrate the power of this method. The generality of the approach is clearly shown in its ability to cope with a four-element aerofoil system.

ACKNOWLEDGEMENTS

The author would like to express his gratitude to Professor A. Jameson and Dr T. J. Baker of Princeton University and Mr A. B. Haines of the Aircraft Research Association for their encouragement and support of this work. The results presented in Figure 14 were kindly supplied by D. Mavriplis of Princeton University.

REFERENCES

1. J. Thompson (ed.), *Numerical Grid Generation*, North-Holland, Amsterdam, 1982.
2. J. Hauser and C. Taylor, 'Numerical grid generation in computational fluid dynamics', *Proc. Int. Conf. on Numerical Grid Generation is Computational Fluid Dynamics*, Landshut, West Germany, July 1986, Pineridge Press, Swansea.
3. W. C. Thacker, 'A brief review of techniques for generating irregular computational grids', *Int. j. numer. methods eng.*, **15**, 1335–1341 (1980).
4. N. P. Weatherill and C. R. Forsey, 'Grid generation and flow calculations for aircraft geometries', *J. Aircraft*, **22** (10), 855–860 (1985).
5. S. H. Lo, 'A new mesh generation scheme for arbitrary planar domains', *Int. j. numer methods eng.*, **21**, 1403–1426 (1985).
6. J. M. Augenbaum, 'A Langrangian method for the shallow water equations based on a Voronoi mesh—one dimensional results', *J. Comput. Phys.*, pp. 240, **53** (2), (1984).
7. B. McCartin, 'Discretisation of the semiconductor device equations', in *New Problems and New Solutions for Device and Process Modelling*, Boole Press, 1985, pp. 72–82.
8. A. Jameson, 'Algorithms for compressible flow calculations on triangles and tetrahedral meshes', *Princeton University MAE Report No. 1732*, December 1985.
9. A. Jameson, T. J. Baker and N. P. Weatherill, 'Calculation of inviscid transonic flow over a complete aircraft', *AIAA Paper 86-0103, AIAA 24th Aerospace Sciences Meeting*, Reno, Nevada, January 1986.
10. G. L. Dirichlet, 'Uber die Reduction der Positiven Quadratischen Formen mit drei Unbestimmten Ganzen Zahlen', *Z. Reine Angew. Math.*, **40** (3), 209–227 (1850).
11. P. J. Green and R. Sibson, 'Computing Dirichlet tessellations in the plane', *Comput. J.*, **21** (2), 168–173 (1978).
12. D. F. Watson, 'Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes', *Comput. J.*, **24** (2), 167–172 (1981).
13. A. Bowyer, 'Computing Dirichlet tessellations', *Comput. J.*, **24** (2), 162–166 (1981).
14. S. Fortune, 'A sweepline algorithm for Voronoi diagrams', *AT & T Bell Laboratories Report*, Murray Hill, NJ. (1985).
15. D. G. Holmes and S. H. Lamson, 'Adaptive triangular meshes for compressible flow solutions', *Proc. Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics*, Landshut, West Germany, July 1986, Pineridge Press, Swansea.