INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 17, 241–255 (1993)

A FRONTAL APPROACH FOR INTERNAL NODE GENERATION IN DELAUNAY TRIANGULATIONS

J.-D. MÜLLER

Department of Aerospace Engineering, The University of Michigan, Ann Arbor, MI 48109-2140, U.S.A. and Von Karman Institute, 72 Chaussée de Waterloo, B-1640 Rhode-Saint-Genèse, Belgium

P. L. ROE

Department of Aerospace Engineering, The University of Michigan, Ann Arbor, MI 48109-2140, U.S.A.

AND

H. DECONINCK

Von Karman Institute for Fluid Dynamics, 72, Chaussée de Waterloo, B-1640 Rhode-Saint-Genèse, Belgium

SUMMARY

The past decade has known an increasing interest in the solution of the Euler equations on unstructured grids due to the simplicity with which an unstructured grid can be tailored around very complex geometries and be adapted to the solution. It is desirable that the mesh can be generated with minimum input from the user, ideally, just specifying the boundary geometry and, perhaps, a function to prescribe some desired mesh size. The internal nodes should then be found automatically by the grid generation code. The approach we propose here combines the Delaunay triangulation with ideas from the advancing front method of Peraire *et al.* and Löhner *et al.* Both methods are briefly reviewed in Section 1. Our method uses a background grid to interpolate local mesh size parameters that is taken from the triangulation of the given boundary nodes. Geometric criteria are used to find a set of nodes in a frontal manner. This set is subsequently introduced into the existing mesh, thus providing an updated Delaunay triangulation. The procedure is repeated until no more improvement of the grid can be achieved by inserting new nodes.

KEY WORDS Unstructured grids Delaunay triangulation Advancing front Internal node generation

1. REVIEW OF PREVIOUS METHODS

1.1. The advancing front method

In the advancing front method,[1,2] an initial list of frontal faces between boundary nodes that represents the boundaries of the domain is established. The smallest face from all fronts is taken as the base of a triangle to be formed. An ideal third node to close the triangle is constructed, in accordance with parameters interpolated on a background mesh that the user specifies. All the other nodes of the existing triangulation that are within a certain radius from the new node are placed in a list, sorted by distance. The ideal node and a few auxiliary nodes are appended to the list of possible nodes. The new triangle is formed with the first node in the list for which the newly inserted faces would not intersect any already existing ones. The front is then updated and the process repeated until all fronts have collapsed, leaving no gaps to be filled.

0271–2091/93/150241–15$12.50
© 1993 by John Wiley & Sons, Ltd.
Received 29 April 1992
Revised 10 February 1993

As can be seen, a large amount of sorting and searching is needed. Especially, the check for intersecting faces is inherently an operation of $O(N^2)$, though the cost can be kept in check with the use of a sophisticated data structure. Numerical experiments in three dimensions[3] show that the computational cost is proportional to $N \log N$ with N tetrahedra in the grid; thus, the method is asymptotically optimal. The meshes created exhibit a very high degree of regularity after smoothing with a Laplacian filter, although the nodes generated may not be connected in an optimal way. A connection that is optimal, in several senses, can be guaranteed by using a Delaunay triangulation, which we discuss next.

1.2. The Delaunay triangulation

A Delaunay triangulation[4] uses a node cloud that is already given and a dissection of the domain into Voronoi regions. Each node is surrounded by its Voronoi region that comprises that part of the plane which is closer to this node than to any other node. The set of boundaries between Voronoi regions is called the Dirichlet tesselation and consists of straight line segments that are equidistant from the two nodes that are closest to each other across that line. A unique triangulation is obtained by connecting the nodes whose Voronoi regions share a common boundary. A closer look reveals that this procedure forms a triangle with the three nodes that are closest to each other. Each triangle has an associated vertex of the Dirichlet tesselation which is the centre of the circumcircle around that triangle. It follows from the construction that this circle does not contain any other node of the triangulation.

It can be shown[5] that a Delaunay triangulation is the optimal triangulation for a given cloud, in the sense that the minimum angle is maximized in any triangle for all possible choices of diagonals between four convex nodes. This leads to some interesting properties of discrete solutions on Delaunay triangulations. Rees[6] and Barth[7] found independently that solving a Laplacian on a Delaunay triangulation using a Galerkin finite element method observes a maximum principle. Rippa[8] proved that interpolating an arbitrary set of data linearly on a Delaunay triangulation minimizes the roughness of the solution. Baker[9] showed that Delaunay triangulations satisfy the uniformity principle of finite element methods, provided a smooth node cloud is used, thus minimizing the truncation error. Of course, this reasoning applies to a set of nodes given *a priori*. A non-optimal triangulation on a better distributed point cloud might have larger minimum angles and, hence, give a better solution. The idea pursued with our method is to use a Delaunay triangulation with a well-distributed node cloud.

Watson[10] gave an algorithm to establish a Delaunay triangulation that reaches the asymptotically optimal count of operations for a triangulation, $O(N \log N)$. This algorithm, commonly referred to as Bowyer's algorithm,[11] consists of a recursive incorporation of all grid points using the circle criterion into an already existing Delaunay triangulation. Initially, this can be a convex hull consisting of one or two triangles that fully cover the domain. Before introducing a new node, all triangles that contain this node in their circumcircle are found and they define the region to be retriangulated. In a Delaunay triangulation this cavity is simply connected, and all faces are visible from the node to be inserted. One obtains a new Delaunay triangulation connecting all faces of the cavity to the new node. After all nodes have been introduced, unwanted triangles outside the domain are deleted. Compared to the intersection check required in the advancing front method, the Watson/Bowyer algorithm needs only checking for distances. Moreover, the generation process is cellwise in the advancing front method and nodewise in Watson/Bowyer, which leads to computational gains with the Delaunay approach, as one finds roughly two cells per node in two dimensions and about six cells per node in three dimensions.

Weatherill[12] proposed a simple way to ensure that the triangulation does not violate any required boundary. His *a posteriori* procedure consists of inserting nodes on the required

boundary segments between two neighbouring boundary nodes that are not present in the final triangulation until all missing segments have been recovered. This method provides the least intrusion of additional nodes into the grid and minimizes the workload as the test is done only once after all nodes have been introduced. Costly checking throughout the triangulation process can be avoided. Alternatively, a simpler edge swapping procedure to recover required boundary edges can be applied in two dimensions that leads to a local violation of the Delaunay criterion.

The insertion algorithm and the boundary enforcement are facilitated if, in addition to the forming nodes of a triangle, the three neighbouring triangles also are stored. We will also make use of this extra information while searching on the grid or introducing new nodes.

1.3. Node placement for Delaunay triangulations

In itself, a Delaunay triangulation does not provide an interior point cloud. A popular approach to overcome this deficiency has been outlined by Holmes and Snyder.[13] A Delaunay triangulation of the boundary nodes is taken as an initial grid. Figure 1 gives such a triangulation for the three element aerofoil configuration shown in Figure 11. The initial triangulation consists of large, very skewed triangles that are found to exceed certain thresholds of maximum area or maximum skewness. Holmes and Snyder propose to measure skewness as the ratio of the radius of the circumscribed circle over twice the radius of the inscribed circle.

Once such a bad triangle is detected, it will be refined by the insertion of a new node at the circumcentre of that triangle. Refinement is performed on the largest triangle in the grid until all

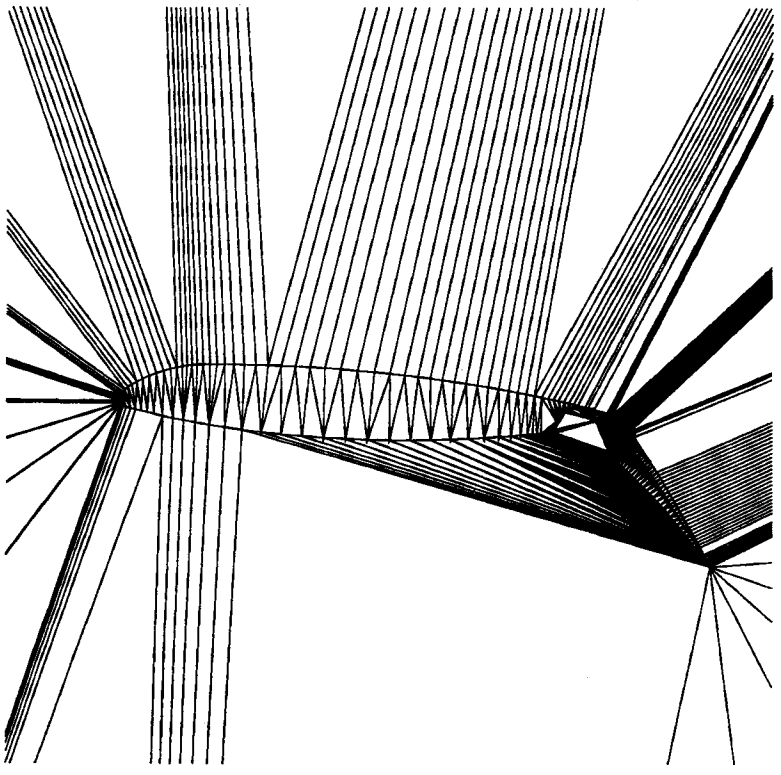

Figure 1. Triangulation of boundary nodes, closeup view. The nodes on the boundary of a three element aerofoil are connected to nodes on the outer boundary (not shown), nodes on another boundary or nodes on the same boundary

triangles are smaller than a first threshold value. Then refinement continues on the skewed triangles starting with the one having the largest circumcircle. The final grid is obtained after all skewed triangles are smaller than a second area threshold. Refining on skewness acts like an implicit mechanism that increases the node density in the close vicinity of boundaries with a finer node spacing.

However, searching for the largest cell or a skewed cell with the largest circumcircle for each new node is a rather costly procedure and the skewness criterion is expensive to evaluate since it involves three square roots during the calculation of the circle ratio. Also, the grids produced by this approach are not as regular as the ones given by the advancing front method. This is due to the fact that the refinement process is much more random compared to an ordered frontal advance.

2. FRONTAL NODE GENERATION

We introduce here a technique that combines the ideas of refining a Delaunay triangulation of boundary nodes with the ideas of frontal advance. In our method, the front will take the form of a boundary between a 'nicely' triangulated region and a 'badly' triangulated region. The method will be described for two dimensions, but we believe there will be few obstacles to implementing it in three dimensions.

If we look at a Delaunay triangulation of a set of boundary nodes (as in Figure 1), we can observe that almost every boundary face is either the short face of a triangle with one very acute angle, or else one of the two short faces in a triangle with one very obtuse angle. In accordance with our idea of a front that separates nice triangles from bad triangles, we take the boundary to define the initial position of such a front. To begin with, we have no nice triangles, but we will introduce a layer of well-positioned nodes that will allow the front to advance.

The construction of the new nodes is an easy task. We simply form an equilateral triangle with the frontal face and either stretch or compress it to match better the spacing requirements of the background mesh; this gives ideal spacing with the two nodes that form the face. A further check is required to make sure the new node is sufficiently distant from the remaining nodes in the grid and from the other new nodes. The desired distance is interpolated on a background mesh that uniquely specifies the distance between nodes everywhere in the domain. Nodes that exhibit bad spacing are either merged with other nodes or discarded. With these new nodes in place, the Delaunay algorithm is re-run and will readily accept the proposed, nice triangles as it resents skewness in its triangulation. We will name these triangles 'explicit' in the following. Also, nice triangles between the new nodes will be formed as they have been tested for sufficient spacing as well, the 'implicit' triangles. In the rest of the domain, Delaunay still has to construct acute cells, though with slightly improved shape.

Again, the short faces of these acute cells denote a frontier between the region with nice cells and the region still waiting to be refined (Figure 2), and the process can be repeated until all bad triangles have vanished. Hence, the algorithm can be cast into the following steps:

1. detect all bad triangles in the grid and find their short faces,
2. find a set of nodes to form nice triangles with the short faces,
3. check whether the new nodes are not too close to any other node already introduced into the structure,
4. check whether the new nodes are not too close to any other new node,
5. retriangulate with the set of new nodes.

The steps are repeated until no more improvement by node insertion can be achieved.

2.1. *Front detection*

The front consists of the interface between the region of properly refined triangles and the unrefined region. A refinement should only take place on a face that has a refinable triangle on one side and an unrefinable one on the other. If refining was merely based on side ratios, an obtuse triangle in the front would lead to the introduction of three nodes. Figure 3 shows the two nodes that would be formed from the two short faces in the front of the obtuse triangle and the node from the face of the acute triangle that neighbours the obtuse one. Not that this extra node is badly placed, but this node should be formed only in the next stage. The introduction of this third node would lead to an irregular front with scattered faces that might not be connected, and the subsequent refining would have much of the randomness of Holmes and Snyder's method.[13]

A triangle is unrefinable if either it is not skewed or it is skewed but node spacing around the cell does not allow further insertion. Checking is simplified by keeping a status flag for each triangle to examine each cell only once. It is to be emphasized that contrary to the usual advancing front method[1,2] no explicit tracking of the front and, thus, no expensive overhead is required.

2.2. *Node construction*

The ideal node to be placed in the mesh would satisfy the distance criterion with all neighbouring nodes, i.e. the distance to all nodes that it will be connected to equals the background spacing evaluated at the midpoint between these two. Clearly, this is an ill-posed

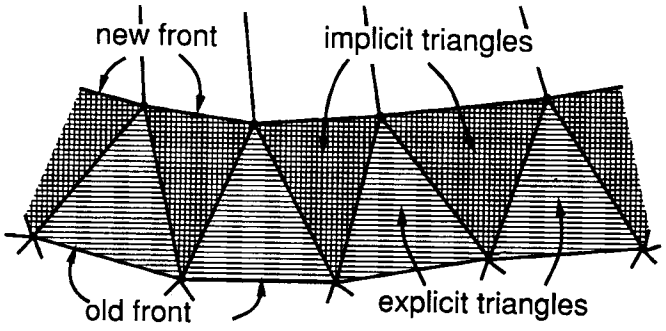

Figure 2. Explicit triangles (striped) and implicit triangles (squared) that are formed along the old front and build the new front

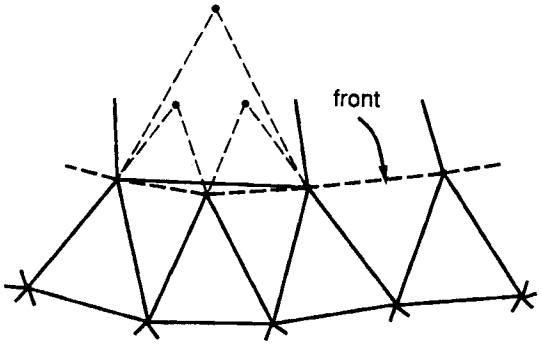

Figure 3. Obtuse triangle along the front with three new nodes formed

problem. But even trying to satisfy the distance condition with the two nodes of the frontal face leads to a system of two quadratic equations. The task will become more amenable with the restriction to isoceles triangles. We will carry out the geometrical construction in an approximate manner leading to only one linear equation.

We approximate the length of the sides l_1 and l_2 opposite to nodes 1 and 2 by $2/\sqrt{3}l$ where the altitude l is as found in an equilateral triangle. Requiring that this approximated sidelength equals the desired spacing h_4 evaluated midway between node 3 and the midpoint of the base, M (figure 4), we find

$$h_4 = \frac{2}{\sqrt{3}}|\mathbf{x}_3 - \mathbf{x}_M| = \left(\frac{1}{2}(\mathbf{x}_3 - \mathbf{x}_M)\nabla h + h_M\right),$$

where h_M denotes the desired spacing at M, ∇h is the local gradient of the background spacing and $\mathbf{x}_3$, $\mathbf{x}_M$ are the position vectors of node 3 and point M. As we place the new node along the median, we can write

$$\mathbf{x}_3 - \mathbf{x}_M = l\frac{\mathbf{x}_3 - \mathbf{x}_M}{|\mathbf{x}_3 - \mathbf{x}_M|} = l\mathbf{n}_3,$$

where $\mathbf{n}_3$ is the unit normal on the base pointing towards the triangle to be refined. The altitude for a triangle with counterclockwise sense is thus

$$l = \frac{h_M}{\frac{2}{\sqrt{3}} - \frac{1}{2}\mathbf{n}_3\nabla h}.$$

Note that in the given form the altitude of the explicit triangle is independent of the length of the base. This preserves the thickness of the layer of cells introduced even if the length of the faces varies strongly (Figure 9).

2.3. *Searching*

The efficiency of unstructured grid generation methods is very dependent on the way specific nodes or cells are found in the grid. For example Watson's/Bowyer's algorithm requires the search for a circumcircle that contains a new node and interpolation on a background mesh involves

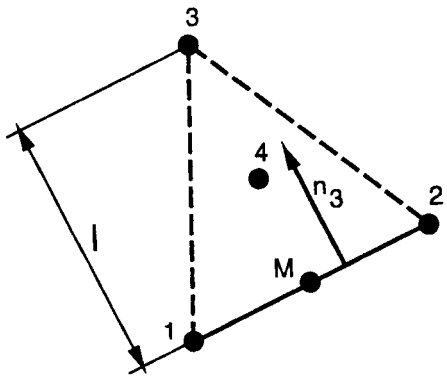

Figure 4. Short face and construct 713 node. The new node 3 and the spacing reference point 4 lie on the median of the face 24

finding the background cell that contains the point of interest. As already stated, an implementation of Watson's/Bowyer's algorithm usually comes with the storage of the neighbours to each triangle and the position of its circumcentre.

Hence, a straightforward way to locate a position in a Delaunay triangulation would be to walk along the Dirichlet tesselation from circumcentre to circumcentre closer towards the target. But this method does not necessarily converge as a Voronoi vertex can lie outside its associated triangle.

A method of similar computational cost is to walk from cell to cell in the direction of the target. As we progress at each step a finite distance towards our point of destiny we must reach our target; so this search procedure always converges.

The direction to turn to is given by the maximum scalar product of the normal on the face and the vector from the midpoint of that face to the target (Figure 5). Of course, only two directions have to be tested once the search is on its way, also it would be rather wasteful to use unit normals. The search is finished when all scalar products are non-positive, indicating that the target lies either in the cell or on a face of the cell.

The cost of this search is $O(\sqrt{N})$ on a mesh with N nodes. However, once the foreground and background cells associated with the new node are determined, all of the triangles in the vicinity, where most of the remaining operations take place, are found in a few steps. Still, this search procedure could also be applied within a specific bin of a tree data structure.

2.4. Background mesh

The Delaunay triangulation of all boundary nodes is computed as an initial triangulation to begin the node generation process. This triangulation provides at no extra cost a suitable background mesh to provide a local value of desired distance between nodes at any point within the convex hull. It will be assumed here that this desirable distance is a piecewise linear function of position, interpolated between the nodal values of a triangle in the background grid. The spacing value at each node is computed as the average distance to its two neighbouring nodes on the boundary.

A linear variation between the fine spacing on a body and the coarse spacing on a far-field boundary is obtained when the background triangle connects directly from the interior to the exterior boundary. However, along concave contours, it may happen that Delaunay connects between finely spaced interior boundaries and the background grid will specify a too large area of

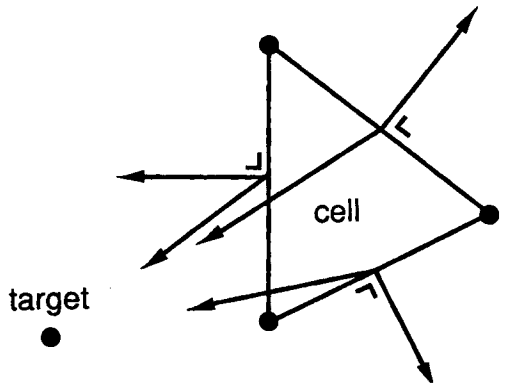

Figure 5. Scalar product criterion to walk from a cell towards a target

fine spacing. Figure 1 gives an example of such an ill-connected background mesh. It shows a close-up of a multi-element aerofoil, obscured by the triangles formed inside the elements. A clearer view of the configuration can be seen in Figure 9. The triangles leaving the frame are connected to the outer boundary. However, the initial triangulation also connects the finely discretized trailing edge of the main flap with the lower side of the main aerofoil and implies an undesirable high node density in the entire ill-connected area between the two elements.

The problem can easily be circumvented by the introduction of extra nodes into the background mesh. To be consistent with the philosophy of minimal user input, one should have the program introduce the necessary nodes and merely ask the user which boundaries he does not want to have connected. The procedure will be to detect an illicit liaison and place a background node in between. During a subsequent retriangulation, most, if not all, of the triangles shared between the two bodies will be broken and few extra nodes will suffice.

The remaining question is what spacing to apply to these new nodes. One would like the spacing to rise smoothly from every boundary node into the domain to match finally the spacing of the outer boundary. This corresponds to extrapolating the spacing with an average gradient from every boundary node towards the automatically inserted node and take the minimum of all these values—an unreasonably costly procedure.

Fortunately, the Delaunay properties make the task at hand a lot more amenable. If we place the new node to break an unwanted triangle at the Voronoi vertex of that triangle, we know that

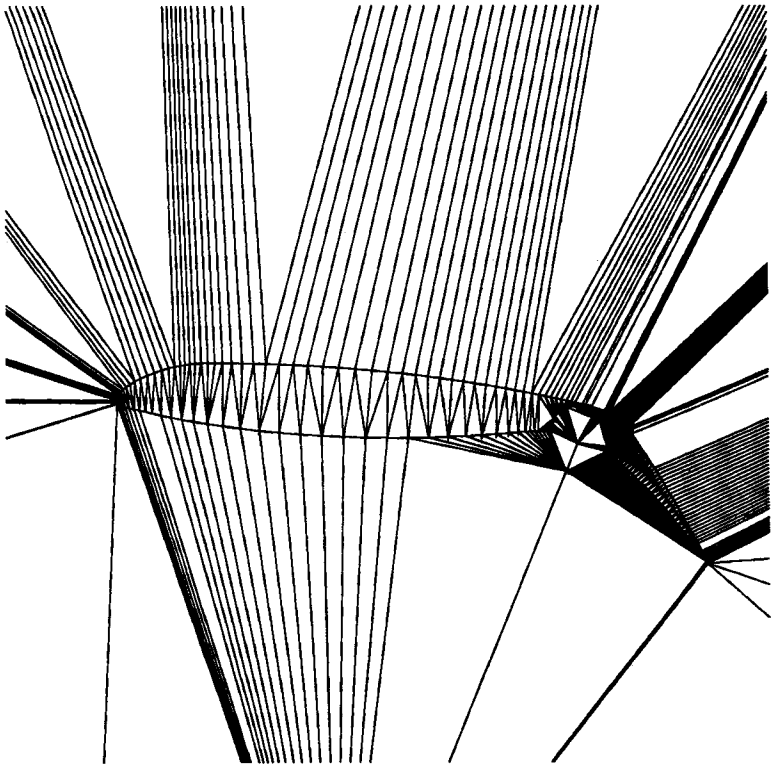

Figure 6. Background grid automatically modified by the insertion of four nodes to break up unwanted connectivity. Two of these nodes are shown between the main aerofoil and the second flap

there is no other node closer to the new node than the three nodes forming that triangle. Moreover, the new node is equidistant from both ill-connected boundaries. We then extrapolate from the spacing of the more finely discretized boundary using the average gradient of the initial triangulation. Figure 6 shows the background mesh modified by automatic insertion. Four nodes have been introduced from triangles in the area between the main aerofoil and the main flap.

2.5. Skewness threshold

So far, the term 'bad' has been used for long skinny triangles, without specifying on what we base this label. From the previous discussion it follows that a criterion is needed that is easy to evaluate and that discriminates the faces to be used in the front. An obvious and inexpensive choice for quantifying the proportions of a triangle is to look at ratios of the squared sidelength over the squared maximum sidelength. A triangle will be called 'bad' once any of its three side ratios drops below a threshold. Considering the fact that a triangle is formed by placing a node somewhere along the perpendicular bisector of the base, one can estimate threshold values for the side ratios. A first estimate can be derived for an acute angled triangle on a zero-gradient background with the length of the face matching the background spacing. In this case the Delaunay triangulation will always form an equilateral explicit triangle with the base, and an implicit triangle with the new node and the distant node of the previous bad triangle.

Figure 7(a) shows the geometry in question. The worst 'implicit' triangle is produced when the distant node of the acute triangle also lies on the perpendicular bisector. If we let the dashed triangle become less and less acute, α will grow and β will become smaller. Both angles will be equal if $b \approx 0{\cdot}64\delta$, so that refining an acute triangle with $b < 0{\cdot}64\delta$ will make the grid worse. Hence, for acute triangles a good threshold is the side ratio of a triangle with $b = 0{\cdot}64\delta$ or

$$\left(\frac{S_{\min}}{S_{\max}}\right)^2 = 0{\cdot}366.$$

Similar reasoning applies to the obtuse triangle in Figure 7(b). Here the two nodes formed perpendicular to the two short faces will be merged subsequently as they are too close to each other. Hence, we have to consider the new node to be placed on the angular bisector. The smallest angle in the old triangulation was α; in the new triangulation it is β. Since $2(\alpha + \beta) = \pi$, matters only improve if $\alpha < \pi/4$. That is, we should only refine obtuse angled triangles for which the side ratio is

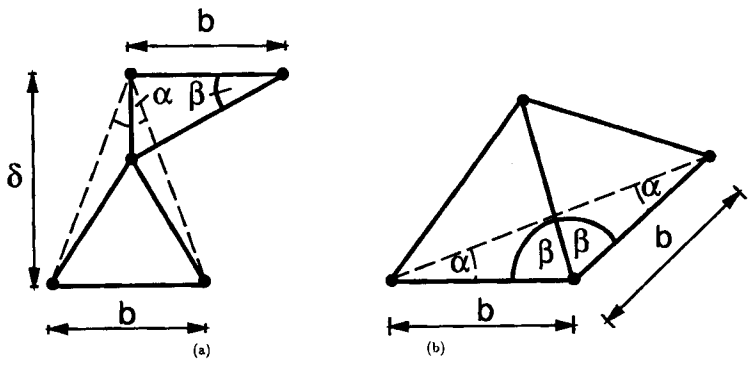

Figure 7. (a) Refining of an acute triangle (dashed) into an equilateral triangle (bottom) and an acute 'implicit' triangle (top); (b) Refining of an obtuse triangle (dashed) into two triangles (full)

less than

$$\left(\frac{S_{\min}}{S_{\max}}\right)^2 = 0{\cdot}5$$

It turns out that the quality of the triangulation is somewhat insensitive to the choice of the side ratio threshold and any value in the range of the two estimates will give good results. This allows one to use the same threshold for both obtuse and acute triangles. The triangulation will change with a different threshold but the minimum angles found will remain virtually unchanged. Strong gradients in the background grid might lead to the formation of explicit triangles that exceed the threshold in the side ratio. Therefore, the altitude of the triangle to be formed will be bounded by the altitude of an obtuse angled triangle with the threshold sidelength ratio and the altitude of its acute counterpart.

$$\sqrt{\left[\left(\frac{S_{\min}}{S_{\max}}\right)^2 - \frac{1}{4}\right]} \leq \frac{l}{S_3} \leq \sqrt{\left[\left(\frac{S_{\max}}{S_{\min}}\right)^2 - \frac{1}{4}\right]}.$$

2.6. Spacing check

The spacing check balances the mechanism of node introduction due to excessive skewness by rejecting or merging nodes; in this way, grid quality is assured for the implicit triangles. We might want to reject a new node, because it falls too close to some existing node, or we might want to merge two new nodes because they are too close to each other. The two cases have to be dealt with separately.

Looking for close nodes that are already introduced, we can make use of the fact that a Delaunay mesh constructed with Watson's/Bowyer's algorithm covers the entire convex hull. We can construct an enlarged circle around each triangle which is the circumcircle plus an added rim of the required distance for the new node. If the new node does not fall within that enlarged circle, the distance between the new node and the nodes of the triangle is at least the required spacing (Figure 8).

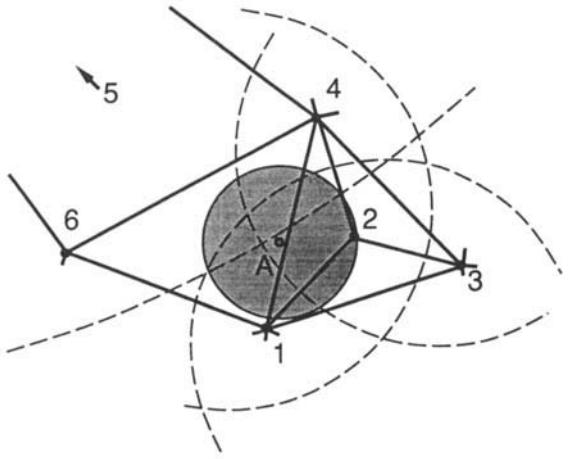

Figure 8. A new node (A) is contained in the triangle 146, but closest to the node 2. The enlarged circles 124, 132, 342 do contain A and require testing. The enlarged circle around 456 does not contain A and excludes nodes 4, 5 and 6 from testing

In a way similar to the tree search during the insertion procedure, the simply connected region can be determined where nodes that are already introduced might be too close to a new node. Once a new node is found to be too close to another old one, it is discarded from the list.

One is less fortunate with checking the distance towards the other new nodes that are also waiting to be introduced. Along the front, we might find a set of very acute triangles that can lie rather oblique to it. The initial front along the boundaries in Figure 1 can serve as a good example. New nodes that are too close to each other might not lie in neighbouring triangles and one cannot make use of the underlying grid. Extensive search throughout the list of new nodes has to be performed, but the list contains only $O(\sqrt{N})$ nodes at a time. This advocates the use of an intelligent data structure that provides some kind of bucketing to reduce further the cost of searching and will retain an optimal count of operations of $O(N \log N)$. Once two new nodes are found to be too close, they are merged. This merging is actually the only step in the algorithm that introduces irregularity into an initially regular mesh. All other steps are independent of the order in which triangles or nodes are encountered. While it is generally not important which neighbours are merged, we do want to have preferred merging of the two nodes that are formed from the two short faces of an obtuse triangle as shown in Figure 7(b). If one searches backwards through the list of new nodes, this pair is encountered first and treated with higher priority.

In order to achieve large minimum angles, we may tolerate nodes being closer to each other than allowed by the background mesh. Otherwise, the skewness mechanism providing refinement may be counterbalanced too much. Initially, the number of new nodes is completely determined by the number of boundary nodes. While the front propagates outwards, the nodes in the front will eventually become too numerous, as the background mesh demands more and more distance between the nodes and the spacing check coarsens the front. In this way, liberal spacing will allow more completely regular rows with the original number of nodes around the bodies and finer spacing will be retained further from the frontal surfaces. On the other hand, being too lax allows node insertion where no improvement can be achieved. An optimum value of requiring 60% of the background distance between inserted nodes has been found in Reference 14. This value leads to the most narrow distributions of minimum and maximum angles around the ideal value of 60°.

It is to be noted that no criterion for too large cells is needed if the front emerges from the finely spaced boundaries. The spacing mechanism will gradually coarsen up that front until it meets the outer boundary. Further refinement in the field can be left to solution-adaptive interaction with the solver. However, the implementation would not pose any problems. If, after retriangulation, two connected nodes in the new front are found to lie too far apart, another node can be introduced between them.

2.7. *Computational cost*

Suppose that a total of N nodes are generated in such a way that N^p new nodes are created every time the front advances. Watson's/Bowyer's algorithm takes $O(\log N)$ operations to introduce a single node into a triangulation. With a dissecting data structure like a split-tree, the cost of searching the list of N^p new nodes requires $O(\log N^p)$ operations. We need N^{1-p} stages to construct the full triangulation; the total cost is thus $O(N(p+1) \log N)$. As p ranges between 0 and 1, the number of operations necessarily increases by a factor of two in the worst case when all nodes are formed in one single stage, compared to a triangulation of specified nodes. Hence, the method is asymptotically optimal.

Actual times are given for the three-element aerofoil case given in Figures 9–12. The initial triangulation of the 328 boundary nodes took 0·54 seconds on a DEC 5000/200, i.e 0·0016 sec per node. The insertion algorithm created 2018 interior nodes and used 10·9 seconds, i.e 0·0054 sec per

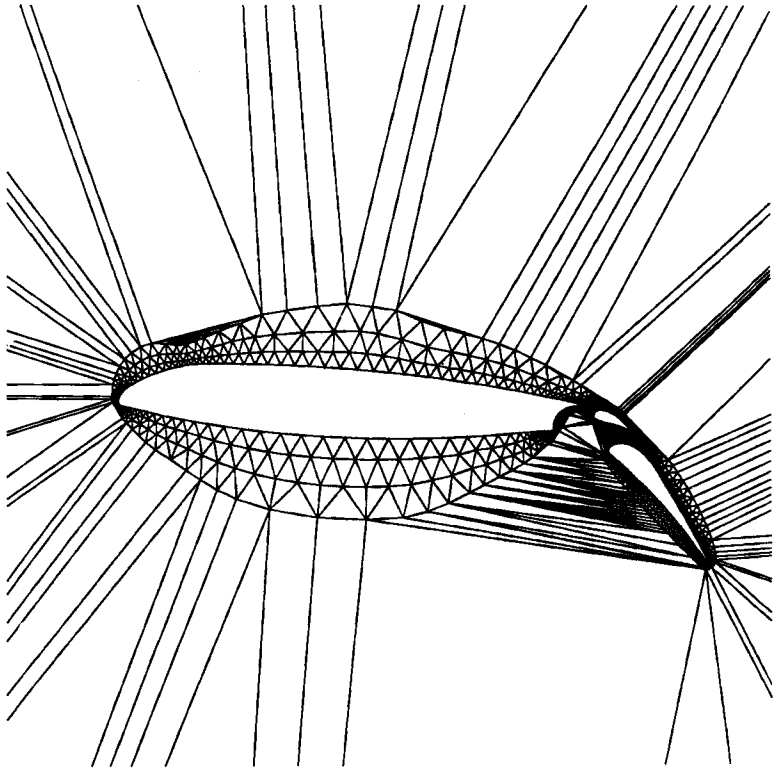

Figure 9. Grid around three element aerofoil after three rows of nodes have been inserted. Note the coarsening of the front in the second row on the upper surface

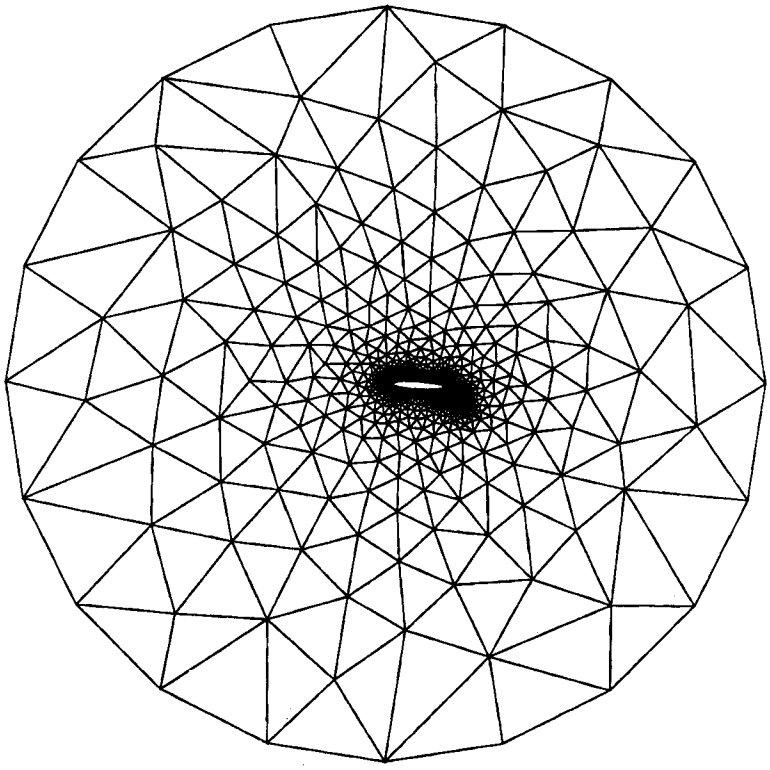

Figure 10. Grid around three-element aerofoil

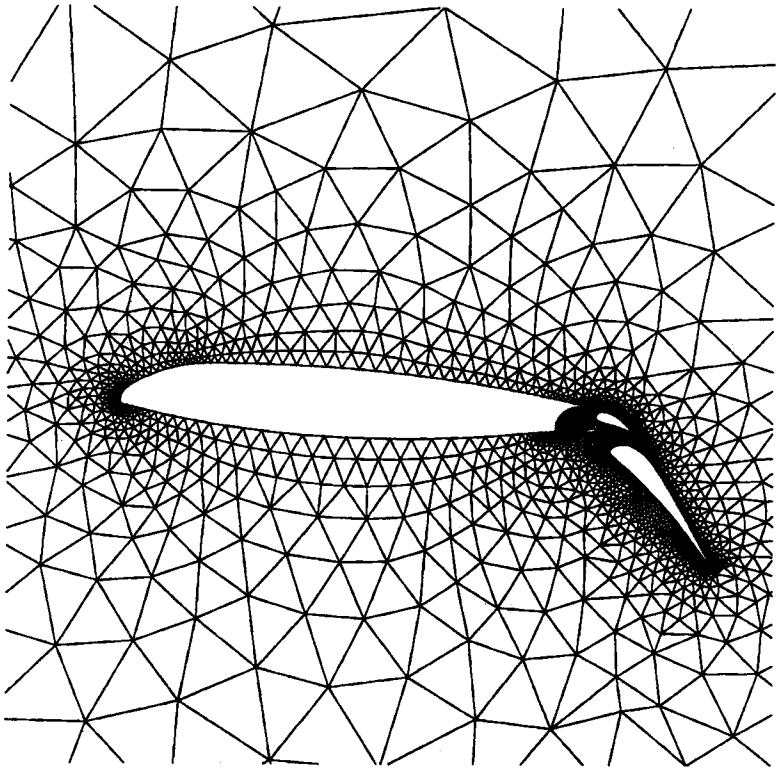

Figure 11. Close-up of grid around three-element aerofoil

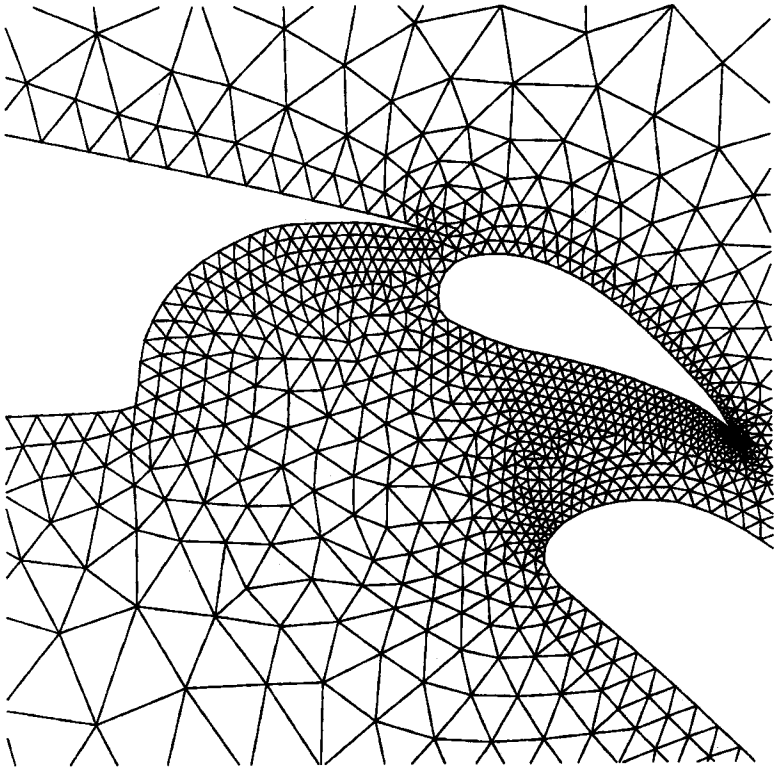

Figure 12. Detail of grid around three-element aerofoil

node. Thus, ignoring the reduced efficiency as the size of the grid increases, generating a new node and triangulating it costs about three times as much computer time as the triangulation of a node alone. Note that the current implementation does not employ any tree data structure and, therefore, the given times could be reduced further.

3. EXAMPLES

A classic case for an unstructured grid generator is the grid around a multi-element aerofoil. Structured grid generation already requires sophisticated extensions to deal with this problem. The background grid for the aerofoil given in Figure 6 was modified by the automatic insertion of four nodes. The initial triangulation is boundary conformal without any modifications.

Figure 9 shows the grid after three rows of nodes have been constructed, the resulting grid is shown in Figure 10, a close-up of the aerofoil in Figure 11. The different rows of nodes can be identified clearly in the final triangulation.

On the upper surface of the main aerofoil,it can be demonstrated how essential the construction algorithm of section 2.2 is to grid quality. In the second row the spacing check has eliminated several nodes and the length of the faces in the new front varies from normal to double. Still the nodes in the third row are aligned evenly, providing nearly equilateral triangles again. The disturbances introduced in the second row are completely eliminated in the fourth row.

The regularity of the grid is entirely due to the frontal insertion, no smoothing filter was applied. Figure 12 shows a detail of the grid between the three elements. The fronts do not break down and merge into each other smoothly. Only very few triangles with maximum angles exceeding 90° can be found. If fronts are aligned to each other, the the resulting point cloud is perfectly regular as between the main flap and the vane flap. The gradual increase in node spacing between the main aerofoil and the main flap is due to the automatic insertion of additional background nodes which are not present in the foreground grid. Overall, the cell surface varies very smoothly with a factor of about 100 000 from the smallest cells at the trailing edge of the vane flap to the largest cells at the outer boundary. The algorithms prove to be very robust, as can be seen from the regularity of the triangulation of the lower rear corner and the trailing edge of the main aerofoil in Figure 12.

The only user input for the case were the 328 boundary nodes and a statement requiring no connection between the main aerofoil and the main flap.

4. CONCLUSIONS

A frontal mechanism for the creation of the interior nodes of a Delaunay triangulation has been developed. The method combines the high node distribution quality of the advancing front method with the optimal connectivity of the Delaunay triangulation. Precise control of node spacing is achieved by the use of the initial triangulation of the boundary nodes as a background mesh with no additional effort of the user. The node generation does not require explicit tracking of the front and is independent of the order in which triangles are listed.

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

All features of this concept extend to three dimensions, where the optimal operation count and the simplicity of front tracking and node construction of the method become even more attractive. The method has been generalized to incorporate stretching to obtain a non-isotropic two-dimensional triangulation. Currently, a three-dimensional implementation is being worked on.

REFERENCES

1. J. Peraire, M. Vahdati, K. Morgan and O. C. Zienkiewicz, 'Adaptive remeshing for compressible flow computations', *J. Comput. Phys.*, **72**, 449–466 (1987).
2. R. Löhner and P. Parikh, 'Three-dimensional grid generation by the advancing front method', *Int. j. numer. methods fluids*, **8**, 1135–1149 (1988).
3. J. Peraire, K. Morgan, J. Piero and J. Bonet, 'Unstructured mesh methods for computational fluid dynamics', Part 4, *Data Structures*, Lecture Series 1990-06 in Numerical Grid Generation, Von Karman Institute.
4. B. Delaunay, 'Sur la sphère vide', *Bull. Acad. Science USSR VII: Class. Sci. Mat.-Nat.*, pp 793–800 (1934).
5. D. T. Lee and B. J. Schachter, 'Two algorithms for constructing a Delaunay triangulation', *Int. J. Comput. Inform. Sci.*, **9**(3), 219–241 (1980).
6. M. D. Rees, 'Numerical solution of the heat equation on triangular grids', *Numerical Analysis Report 88/2*, Computing Laboratory, University of Oxford, 1988.
7. T. Barth, 'On unstructured grids and solvers', Von Karman Institute Lecture Series 1990-03 in Computational Fluid Dynamics.
8. S. Rippa, 'Minimal roughness property of the Delaunay triangulation', *Ph.D. Thesis*, Tel-Aviv University, 1989.
9. T. Baker, 'Element quality in tetrahedral meshes' *Proc. 7th Int. Conf. on Finite Element Methods in Flow Problems*, Huntsville, Alabama, April 3–7, 1989.
10. D. F. Watson, 'Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes', *The Comput. J.*, **24**(2), 167–171 (1981).
11. A. Bowyer, 'Computing the Dirichlet tessellation', *The Comput. J.*, **24**(2), 162–166 (1981).
12. N. P. Weatherill, 'Grid generation, Part 3, in The Delaunay triangulation', Von Karman Institute Lecture Series 1990-06 in Numerical Grid Generation.
13. D. G. Holmes and D. D. Snyder, 'The generation of unstructured triangular meshes using Delaunay triangulation', in *Proc. 2nd Conf. on Grid Generation in Computational Fluid Dynamics*, Pineridge Press, Swansea, 1988.
14. J. Peraire, K. Morgan and N. P. Weatherill, 'A comparison of Delaunay and advancing front methods of mesh generation for 3D configurations', *Proc. 2nd World Congress on Comp. Mech.*, Stuttgart, 1990.
15. J.-D. Müller, 'Proven angular bounds and stretched triangulations with the frontal delaunay method', 11th AIAA CFD Conference, Orlando, 1993, to be presented.