INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 37, 3605–3619 (1994)

THE ADVANCING-FRONT MESH GENERATION METHOD REVISITED

PAUL LOUIS GEORGE AND ERIC SEVENO

INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France

SUMMARY

This paper presents a mesh generation method of the advancing-front type which is designed in such a way that the well-known difficulties of the classical advancing-front method are not present. The retained solution consists of using the first steps of a Voronoï–Delaunay method to construct a background mesh which is then used to govern the algorithm. The two-dimensional case is considered in detail and possible extensions to adaption problems and three dimensions are indicated.

1. INTRODUCTION

The advancing-front mesh generators, adapted to arbitrary geometries, have been studied by in References 2–4 and, more recently, in Reference 5. This kind of mesh generator constructs the mesh of the domain from its boundary. The elements created are triangles in two dimensions (variations exist which enable to create quadrilaterals almost everywhere, depending on the number of sides forming the given boundary, see References 6 and 7) and tetrahedra in three dimensions. The data required consists of the boundary, or more precisely, a polygonal discretization of it (dimension 2), input as a set of segments, or a polyhedral discretization (dimension 3), input as a set of triangular faces. Additional items (points, edges or faces), which must be considered as prescribed items and thus which must be present in the created triangulation, can be included in the data.

The main reason for which the input data have been defined in this way is to be independent of the manner in which the domain is designed. As a consequence, the mesh generation algorithm is not connected with such or such CAD system and can be discussed separately.

The process is iterative: a *front*, initialized by the set of given items (those describing the boundary and the specified items, if any), is analysed to determine a *departure zone*, from which one or several internal elements are created; the front is then updated and the element creation process is pursued as long as the front is not empty.

Section 2 attempts to give a survey on the classical meaning of the advancing-front method. Possible variations, necessary steps, as well as difficulties are mentioned. Section 3 presents, in two dimensions, a new method combining the Voronoï–Delaunay approach and the advancing-front technique. It will be shown that this method permits us to avoid the classical difficulties of the advancing-front mesh generation algorithms by introducing a *control space* (Section 5) and a *neighbourhood space* (Section 6). Section 7 proposes a possible scheme for the mesh generation method after which, Section 8, some application examples are depicted. Section 9 indicates some key ideas regarding the extension of the method to adaption context and to three dimensions.

CCC 0029–5981/94/213605–15$9.00

Received 25 February 1993
Revised 22 January 1994

2. GENERAL PRINCIPLES

The process of any advancing-front method can be summarized as follows:

(1) Initialization of the front
(2) Analysis of the front by

  (a) determining the departure zone;
  (b) analysing this region by
    (i) creating internal point(s) and internal element(s);
    (ii) updating the front.

(3) As long as the front is not empty, go to 'analysis of the front'.

The determination of the departure zone, the analysis of the front and the way in which the elements are created can be done in several ways. With each choice a particular advancing-front method is associated. The departure zone can be chosen as: (i) a portion of the current front where some properties are satisfied (length of consecutive segments, angles,. . .); (ii) a front defined iteratively as the translation of a set of items of the initial contour; (iii) the entire current front.

The first approach caters, first of all, for the treatment of particular zones (for example, those with 'small' angles or 'small' edges). The second choice induces a propagation from an initial zone, while the third produces an inflation from a given contour. It is not clear how to define an optimal strategy in all geometrical cases.

Whatever the choice, the retained items are analysed in terms of geometrical properties with regard to the local tools available for the creation of points and elements, For example,[15] the first reference in this domain, lists a series of patterns and finds if there exist such configurations in the examined zone. A point and element construction is associated with each of these cataloged patterns. After a construction, the front is updated and the process is repeated as long as the front is not empty.

The difficulties when considering such an algorithm are well identified. They consist in

(1) determining the type of situation which governs the type of construction,
(2) knowing if a point is inside the domain or not,
(3) defining a suitable position when a point is created. Three constraints must be satisfied: the created point must be inside the domain; the element(s) resulting from this point creation must be well-shaped and it (they) must be such that the remaining zone can be dealt with without difficulties at a later stage.

In practice we have to answer the following two questions: *where are we* and *what do we have to do* in the place we are? The first question means that we need to check, at each state, that every point under consideration (Figure 1) is inside the domain, and need to know if there is any existing point or edge that can possibly interfere with the envisaged construction. The second question means that we want to decide if the computed position of the so-created point is optimal with regard to some neighbourhood. A typical example of an ambiguous situation results from the junction of elements which are very different in size and must be joined. This configuration appears when the initial discretization of the contour includes segments with very different lengths (Figure 2). The traditional advancing-front method leads to merge two fronts of widely differing length scales and, thus, may generate cross-over or, at least, badly shaped elements. Dealing first with 'smallest' items (in terms of an adequate definition of the departure zone) and creating non-equilateral element is a method to avoid this difficulty in some cases. Another solution can be used to solve this problem: the method includes a step that enables to remove

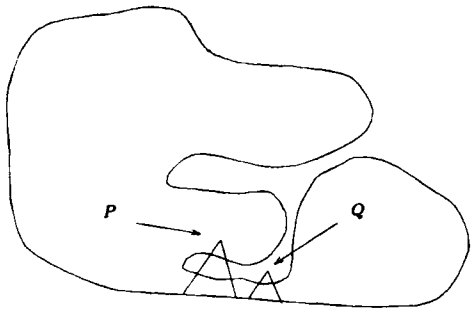

Figure 1. Checking if a point is inside the domain

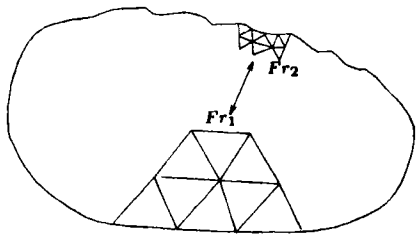

Figure 2. Merging of two widely different fronts

some elements created previously when they cause trouble at a later step. Obviously, something is conceptually missing: a control space, i.e. a way to govern the process by indicating, in some way, what the optimality is in the zone under consideration.

The method discussed in the following sections aims to give a new way of considering the advancing-front technique, while avoiding all the trouble mentioned above by introducing a control space and a neighbourhood space. It means that we assume that we have a fast way of deciding, in any region of the domain, what to do. As a consequence, while the general scheme remains similar to that of the classical approach, the above difficulties disappear.

3. CREATION AND INSERTION OF AN OPTIMAL POINT

Let us introduce a control space and a neighbourhood space. These structures will be described in detail later on (Sections 4 and 5) and are only briefly introduced here. The control space consists of a covering-up of the domain with which, at each point of the domain, a stepsize (denoted by h_{loc} in that follows) is associated. Similarly, the neighbourhood space is a covering-up of the domain with which pointers are associated. These pointers are used to know what the items located in some neighbourhood of any examined region are. Thus, it is now possible to perform all the steps of the advancing-front method, so revisited, without difficulty. These steps are discussed below in the two-dimensional case.

One of the problems arising when implementing any advancing front is to decide quickly if a given point is inside the domain or not. In practice, to ensure that a point is internal to the domain, one has to verify that

(1) all edges including this point as endpoint do not intersect any edge belonging to the front,
(2) all created elements do not include any point.

The previous properties involve *a priori* only the items of the current front and not all the existing items. In fact, only part of the front needs to be considered and the decision is therefore obtained rapidly. To define the part of the front which must be visited, we use the control and the neighbourhood spaces.

With regard to the manner in which the current front is analysed and the nature of the departure zone, front items are dealt with in a predefined order. Assume that we examine a front item (edge) obtained by one method or another, then we are able to define an optimal point very easily. The process is as follows (we denote by AB the front edge under consideration, Figure 3):

(1) Point C is constructed so that triangle ABC is equilateral,
(2) $h_{\text{loc}}(C)$ is computed using the control space: according to h_{loc} and the current distances from C to A and B, point C is relocated on the mediatrice of AB in such a way that these three values become equal.
(3a) Points of the front other than C, falling in some neighbourhood, are examined in order to decide if they can be connected with AB in such a way that triangles ABP_i, where P_i denote these points, are valid. The retained points form the set $\{P_i\}$, $i = 1, n$.
(4) This set of points is sorted with respect to the distance from C, after which point C is placed at the end of the so-formed set.
(5) The previous set is now visited: we pick the points in the set (starting with the first) and, considering one of them (namely, P_k), we check if P_k is inside the domain (see above discussion), if triangle ABP_k is positive and empty and, in addition, if angles β_1 and β_2 are sufficiently large. For example, in Figure 5, a case is depicted where optimal point C is the only point in the list but will cause trouble furthermore. In this case, r, the size defining the examined region, is increased until a point P is found which can become a member of the list and then can be retained. As soon as a point verifies all these properties, it is retained. In the case where no point is selected, we return to (3a) while the size of the visited neighbourhood (r in the Figure 4) is increased.
(6) Triangle ABP_k is formed (see below discussion) and the front is updated.

Angles β_1 and β_2 are used to ensure that the element in creation will not cause trouble latter (Figure 5). The created triangle must be empty, i.e. no specified point must be contained in it and no specified edge must intersect one of the constructed edges. This latter requirement is necessary if the data include some specified items other than that of the contour discretization: while a specified edge may be included in the initial front so that this additional treatment becomes

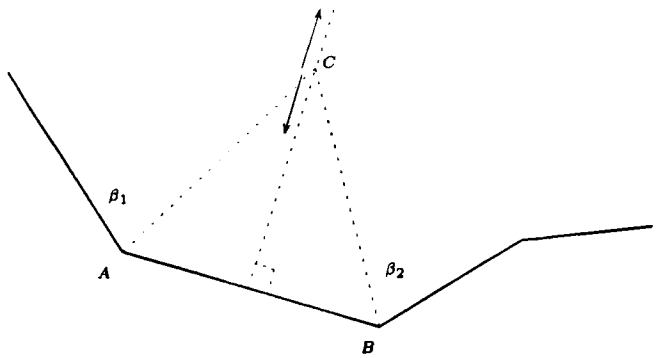

Figure 3. Optimal point with respect to edge AB

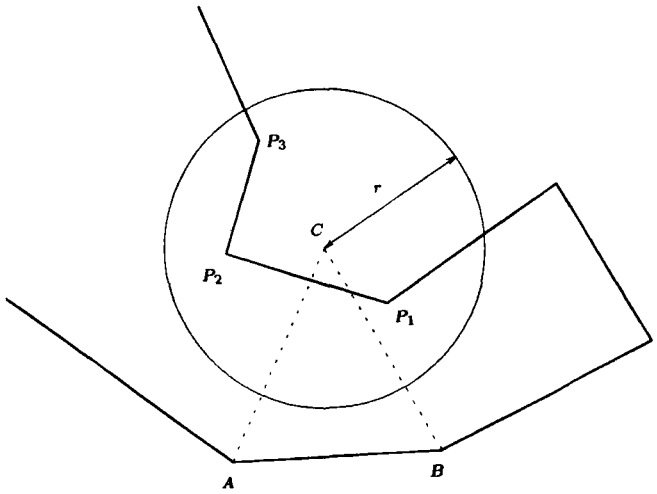

Figure 4. Candidates

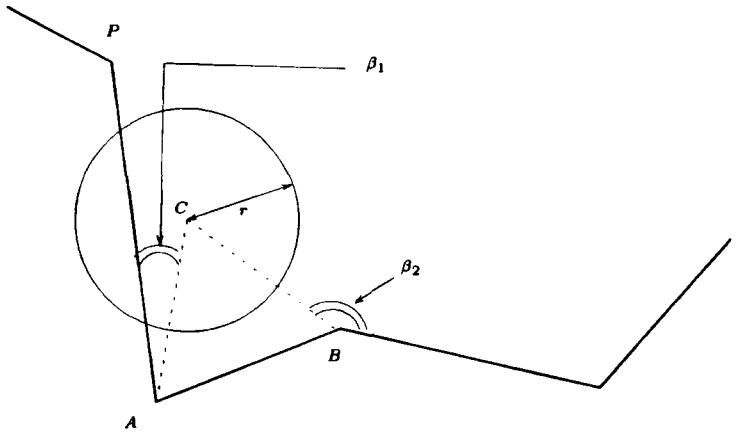

Figure 5. Case where no 'natural' candidates other than C exist

useless, this is not the case for specified points, if any, which must be taken into account in this way.

Due to the method in which the points are created, the method prefers to use an existing point rather than create a new one. This remark leads to minimize further delicate situations.

The effective insertion of the point defined in the previous step consists in creating *only one element* by connecting the point with the item of the front from which it has been constructed. The only thing to ensure is that the surface of the element created is positive.

In some situations, it is clear that several elements, and not only one element, may be created when considering a point. In the present discussion, the way in which a point is defined ensures that the two methods are equivalent and, as a consequence, the scheme, involving only one case is simplified.

To update the front, one has to

(i) suppress the edge of the former front which was used to define the point just created, in the case where such a point has effectively been constructed, or suppress the edges which are included in the triangles just created in the case where an already existing point has been selected as an optimal point;
(ii) add to the front the edges of the triangle(s) just formed which are not common to two elements.

Once the front is empty, the method has converged and the mesh of the domain is completed.

4. CONTROL SPACE

The control space is a way to associate a stepsize all over the domain easily. The control space consists of a control structure, i.e. a covering-up of the domain with which a function defining the desirable stepsize in each point is associated. Several choices can be made to construct the control structure.[8] An example is given below depicting such a choice for the case where the only data at our disposal is the contour discretization of the domain.

Thus, the control structure is defined as a regular grid enclosing the domain. A cell in this grid has a size, Δ_c, depending on the size of the initial data items. For example, a solution is to set Δ_c to be about the length of the smallest edge in the data. This value may induce a too large number of cells so an alternative solution is to set Δ_c to be the average length of the given edges. When such choices lead to problems, the solution is to define the control structures as a quadtree structure which offers the desired flexibility (see References 9 and 10).

We have now to define the function associated with the control structure and, more precisely, with the cell vertices. The problem is, again, that the only data known consists of the boundary discretization. While it is easy to find the grid cells intersecting one edge of the data and then define a desirable stepsize by means of interpolation based on the size of these edges, it is not easy to define this stepsize for the other cells. This is the reason why we first use the first steps of a Voronoï–Delaunay mesh generation method to obtain the *boundary mesh* of the domain. This boundary mesh (see References 8 or 11) is defined as a mesh resulting from the insertion of the boundary points which are the only points we know at this stage. Once the triangles of this boundary mesh are constructed (Figure 6), we define the stepsizes at the grid cell vertices as follows (Figure 7).

For all cell vertices:

(a) find the triangle in the boundary mesh containing it,
(b) compute the desirable stepsize, h_{loc}, as the P1-interpolate of the h_{loc} of the three vertices of the triangle.

On completion of this operation, we have fully defined the control space which can be now used (Figure 8) in the following manner: let P be the point under consideration in the current construction, then $h_{loc}(P)$ is computed as follows:

(1) find the grid cell within which point P falls,
(2) compute the desirable stepsize, $h_{loc}(P)$, as the Q1-interpolate of the h_{loc} of the four vertices of the cell.

When the data includes information other than the boundary discretization, similar processes can be used by effectively taking this information into account.

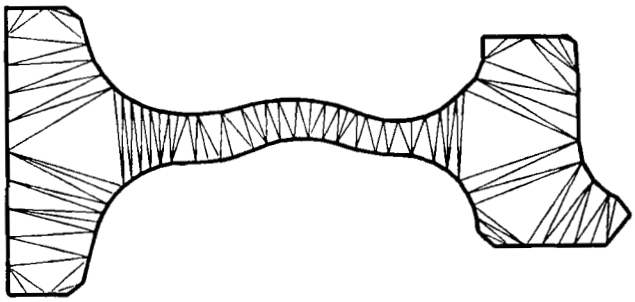

Figure 6. Boundary mesh

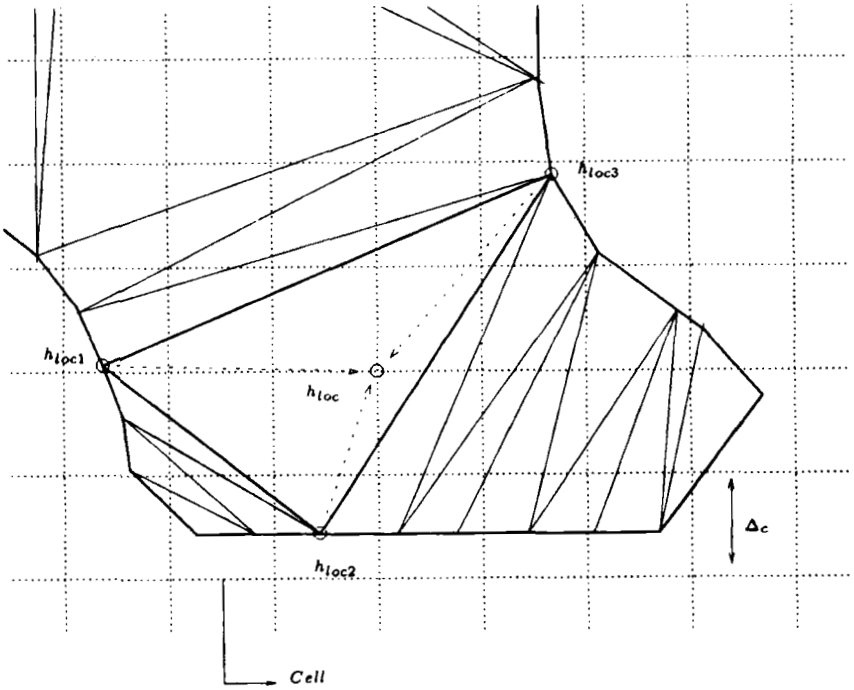

Figure 7. Control space (construction by using the boundary mesh)

The above method for constructing the control space relies on the boundary mesh and needs the use of a Voronoï–Delaunay method. In general, classical advancing-front methods are not conceived in this way. It means that they cannot take advantage of a boundary mesh to help the control space construction. In this situation, it is more delicate to develop a mesh generation algorithm able to deal with complex shapes. The main problem which is not easy to solve deals with the configuration shown in Figure 2 where elements with very different size must be joined. To overcome this difficulty, the mesh generation algorithm can be modified as follows.

(1) at the definition of the departure zone step: it seems that the first encounter with small edges (in two dimensions) and small or badly shaped triangles (in three dimensions) leads to avoid the problem of joining two parts of the front including items with different sizes;

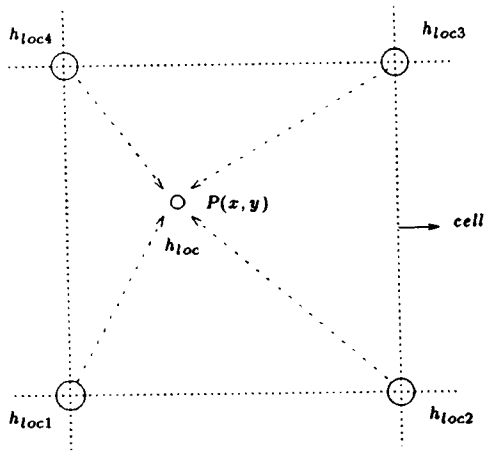

Figure 8. Control space (utilization)

(2) at the element creation step: when only a badly shaped element can be constructed, the method must include a removal procedure able to suppress elements previously created in such a way that the present difficulty disappears (such a procedure must be carefully designed, in particular to avoid recursivity).

Thus, it seems that our solution permits to avoid all these troubles. In addition, it can be noted that the construction of the boundary mesh used to define the control space is very cheap in terms of CPU times.

5. NEIGHBOURHOOD SPACE

The neighbourhood space is a way to easily find items (points, edges or faces in three dimensions) located in a given neighbourhood of the region under consideration. Similarly as to the notion of a control space, the neighbourhood space is composed of a covering-up of the domain (possibly identical to the previous) with which the useful information is associated. Several choices can be envisaged of which only an example is given.

The control structure is defined as a regular grid enclosing the domain. The cell of this grid has a size, Δ_n, which can be equal to Δ_c. We then construct a binary tree governed by the data points (points of the boundary discretization and specified points if any). This construction is done as follows: for each point (see Figure 9 as well as Figure 10):

(1) find the cell, n, in which point P_i under consideration falls ($n = 26$ for point number $P_i = 1$);

(2a) consider the next point ($P_{i+1} = 2$) and find the cell containing it (26): if the cell number is n, class P_{i+1} in cell n, else if the cell number is p not equal to n, define the right branch ($p > n$) or the left branch (otherwise) attached to node n and class P_{i+1} in this cell. Then set $i = i + 1$ and return to (2a).

In this way, we obtain an ordered tree attached to the points whose nodes point out to the grid cells.

On completion to this operation, we have defined the neighbourhood space which can now be used in the following manner: let P be the point under consideration in the current construction,

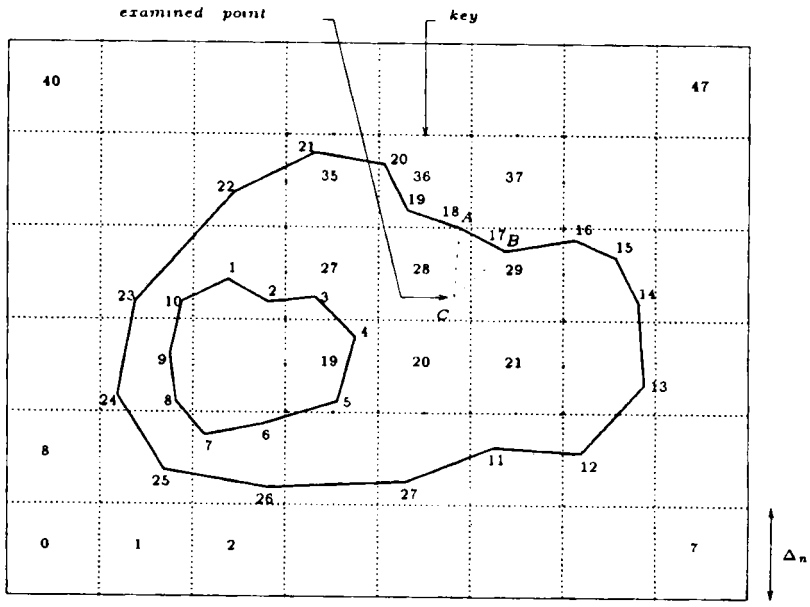

Figure 9. Neighbourhood space

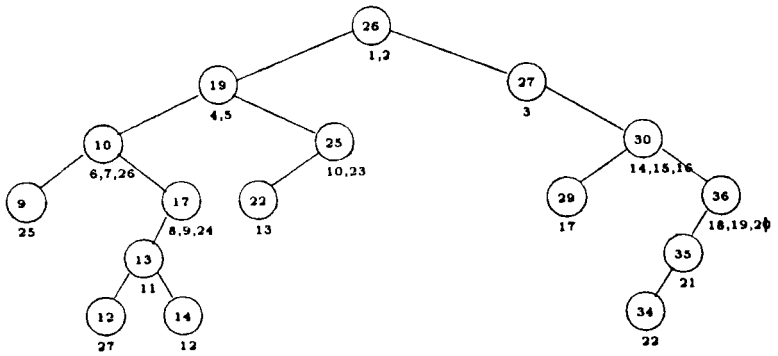

Figure 10. Corresponding tree

then the points which must be examined are those pointed out in the cells located in the appropriate neighbourhood of the cell containing the point used to define point P. Due to the way the previous tree was created, this operation can be done very rapidly.

Variations exist to construct this neighbourhood space but it seems that there are no major difficulties to create a suitable structure containing, in one way or another, the desired information accessible via a binary tree structure or a different basic structure (quadtree,. . .).

6. MESH POINT AND ELEMENT SMOOTHING

It can be of interest to smooth the triangulation resulting from the mesh generation method. This process can be based on two different techniques which can be coupled.

(i) Moving the points: this operation can be performed in three ways:

Barycentrage: The set of points surrounding a given point P is determined; they correspond to the vertices of the elements sharing vertex P. Point P is then repositioned at the weighted barycentre of its surrounding points:

$$P = \frac{1}{n} \sum_{j=1}^{n} \alpha_j P_{k_j} \tag{1}$$

where n denotes the number of points connected to point P, P_{k_j} denote these points with numbers k_j, and α_j is the associated weight ($\sum \alpha_j = 1$).

Relaxation: This variant consists of 'relaxing' the above expression:

$$P^{m+1} = (1 - w)P^m + \frac{w}{n} \sum_{j=1}^{n} \alpha_j P_{k_j}^m \tag{2}$$

using identical notation with w a relaxation parameter.

Iterations governed by a criterion: Let P be a vertex of the mesh and Q be an associated measure (for example, the maximum of the quality of the elements with vertex P), then an iterative process for the displacement of P along an arbitrary direction is defined, where the amplitude is initialized to some given value. The displacement of P is simulated and the associated value Q is computed by successive iterations; the way in which it varies directs the process. The quality, following Reference 1, is defined as

$$Q = \alpha \frac{\rho}{h}$$

where α is a normalization factor, ρ the incircle radius and h the element diameter.

(ii) Local modifications: The use of diagonal swapping can be of interest to obtain better shaped elements.

All of the above smoothing methods must be controlled to guarantee that the triangulation remains valid and that its quality is enhanced with respect to the given measure of element quality.

7. SCHEME FOR THE MESH GENERATOR

Results of the previous sections are used to propose the following steps as a possible automatic mesh generator based on the advancing-front method:

1. Create the set of points associated with the data, i.e. the boundary points of the domain (the endpoints of the segments discretizing this boundary) and the field points if they are known.
2. Compute the location of four supplementary points in such a way that the quadrilateral so-formed contains all the points in the set.
3. Create the mesh of this box using two triangles.
4. Insert, using the Voronoï–Delaunay method, the points of the initial set to obtain a mesh including these points as element vertices.
5. Regenerate, using the same method, the boundaries and define the outside elements.

These steps involve the first part of a Voronoï–Delaunay-type method (see Reference 12) and enable us to define the useful control and neighbourhood spaces we need in the present case easily

and rapidly. Thus, the following steps can be envisaged:

6. Create the control and the neighbourhood spaces.
7. Define the initial front.
8. Define a departure zone, create an optimal point or find an existing point satisfying the suitable properties, insert the retained point to construct one or several elements and update the front.
9. If the current front is not empty, return to 8.
10. Apply smoothing processes.

Obviously steps 1–5 must be suppressed when a background grid is used which enables us to construct the control and neighbourhood spaces needed in the following steps of the algorithm directly. They must also be omitted when a different way to construct equivalent information is possible. The steps appear to be an elegant way to govern the mesh generation method in the case where no information other than the boundary discretization is known that is the case in which this study takes place.

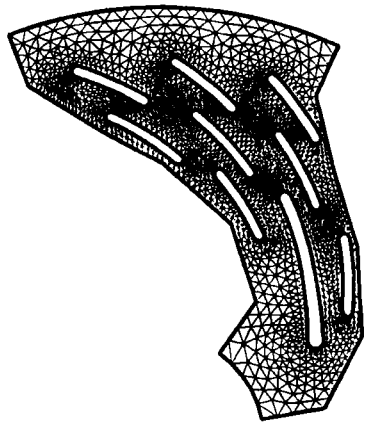

Figure 11. First example (courtesy of Telma and Simulog)

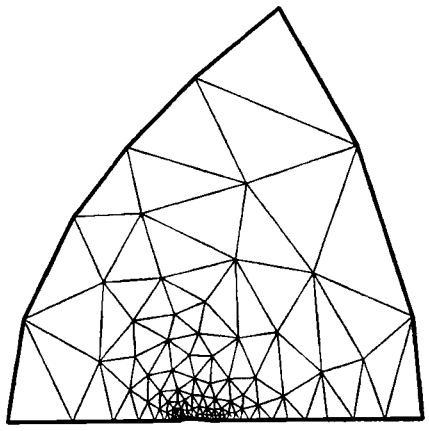

Figure 12. Second example (classical mesh surrounding a profile)

8. APPLICATION EXAMPLES

The examples depicted in this section rely on geometrical considerations only. It means that the triangulation created is governed by the given boundary of the domain under consideration. The finer the discretization of the boundary is the finer the triangulation is in the corresponding regions. Figures 11 and 12 illustrate the triangulations obtained in this case.

Triangulation 1 consists of 4442 elements, 2496 vertices and its worst quality (i.e. Q as defined in Section 6) is 2·04 (only one triangle has quality worse than 2), while triangulation 2 includes 195 elements and 124 vertices, with a quality equal to 2·70. The construction of the boundary mesh requires 6 s (example 1) and 1 s (example 2) while the total CPU times are, respectively, 39 s and 10·3 s (HP 425 workstation).

A parameter can be introduced to dilute the boundary information in such a way that elements are larger in regions far from it. Figure 13 depicts the mesh obtained by the advancing-front method in the same domain than that of Figure 6 (the boundary mesh resulting from a Voronoï–Delaunay method used as example in Section 4) while Figure 14 is the mesh resulting from the advancing-front method applied with a dilution parameter of 1·2. The first triangulation consists of 512 elements, 313 vertices, its worst quality being 2·24, while the diluted triangulation includes only 300 elements and 207 vertices, with a quality of 2·09. The construction of the boundary mesh requires 1·8 s (examples 3 and 4) while the total CPU times are, respectively, 5·1 s and 4·4 s (HP 425 workstation).

In Reference 13 is described the solution for a scattering waves problem, i.e. the Helmholtz equation, using a mesh constructed with the proposed method. The main interest of the present mesh generation method is that the triangles constructed are almost identical and equilateral (Figures 16 and 17) because the data discretization consists of identical edges (Figure 15). This property induces that the problem is solved with efficiency and accuracy as, on the one hand, the physical behaviour is well captured, and, on the other hand, the timesteps used in the solution method can be sufficiently large so that the computational effort is reduced. The resulting mesh consists of 284 037 triangles and 143 932 vertices. The worst quality is 1·96, i.e. all the elements

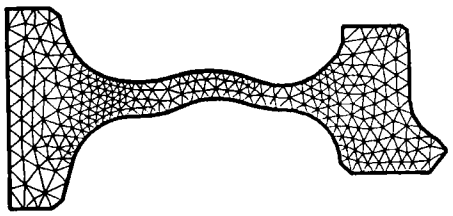

Figure 13. Third example (automatic mode, i.e. no dilution)

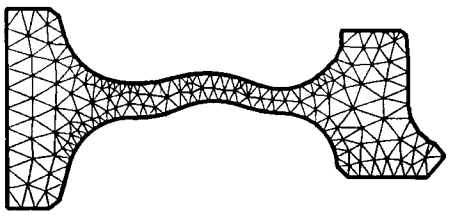

Figure 14. Same example with dilution of boundary information

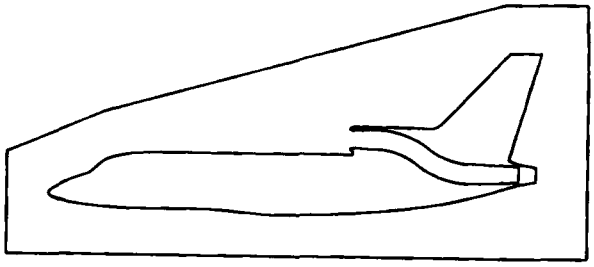

Figure 15. Falcon 50 boundary discretization serving as data (Courtesy of Dassault-Aviation)

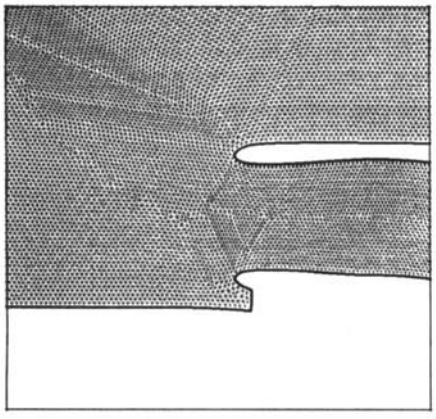

Figure 16. Detail of the mesh (1)

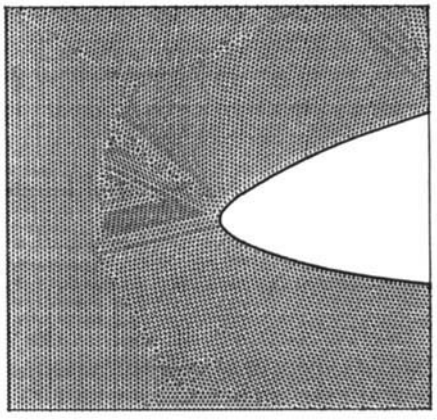

Figure 17. Detail of the mesh (2)

have a quality better than 2. The times required are 1180 s for the final mesh and 37 s for the boundary mesh

9. CONCLUDING REMARKS AND POSSIBLE EXTENSIONS

The mesh generation method discussed throughout this paper is fully automatic. It relies on the use of a Voronoï–Delaunay method to be able to define the necessary control space which is introduced. It also uses some basic data structures as regular grid or quadtree structure to help the construction of the above space. While involving these different methods, which can be by themselves used as mesh generation methods, it is clearly of the advancing-front type.

The Voronoï–Delaunay method appears to be an easy and consistent way to define the h_{loc} everywhere in the domain under consideration whatever it is. The grid or the quadtree structures are only a convenient manner to find the h_{loc} associated with a point.. It is possible without major difficulties to obtain directly these values by using the background mesh but, in this case, a bigger computational effort is necessary.

The method has proved to be numerically robust and efficient, in particular when the data includes widely different items. On the other hand, it has been compared with a purely Voronoï–Delaunay method. In the case where the data are composed of identical (in terms of sizes) segments, the method generates preferentially 'more' equilateral triangles than the Voronoï–Delaunay method; in the general situation it represents an alternative mesh generation method.

One of the advantages of the method is that it can be, in principle, extended to the two following situations:

(1) the mesh generation of adapted meshes,
(2) the mesh generation in three dimensions.

In the case of adaption, the principle can be as follows: firstly an initial mesh is created using the proposed method, i.e. by using only as data the boundary discretization of the domain; then the solution of the physical problem is computed and analysed in such a way that for each nodes of the mesh a desirable h_{loc} is obtained. After which, a natural extension of the method is, on the one hand, to replace the boundary mesh used when the control space is constructed by the mesh obtained at the current iteration of the adaption process and, on the other hand, to replace the computation of the desired h_{loc} by taking into account the h_{loc} associated with the nodes of the current mesh. This process is then iterated until the desired rate of convergence is obtained.

A paper is in preparation to make clear this extension of the method in the case of adaption.

For the mesh generation problem in three dimensions, steps 1–5 can be performed following Reference 14 while step 6, consisting of the creation of the control and neighbourhood spaces can be made as follows:

(a) the grid cells consist now of hexahedra (or an octree structure could replace a quadtree structure).
(b) P1-interpolation is based on tetrahedra (instead of triangles),
(c) Q1-interpolation is based on hexahedra (instead of quadrilaterals),

All the other steps clearly apply to three dimensions, the only difficulty that can be met leads in the fact that there exist polyhedra (seen via their boundary discretization as a list of triangular faces) which are impossible to mesh without additional point as soon as the data faces must be preserved (see Reference 14); in this situation, a so-useful point is created in violation of the h_{loc} control.

A work devoted to the three-dimensional case is in progress to propose solutions in this case (both for the classical and the adapted situation).

The software complexity of the proposed method should not be underestimated. As several techniques are used, more software are necessary. In fact, the authors have already experimented a Voronoï–Delaunay mesh generation method both in two and three dimensions and the re-use of this algorithm at the beginning of the method does not constitute an additional effort.

ACKNOWLEDGEMENT

The authors want to thank the reviewer for very fruitful comments which are at the source of this revised version.

REFERENCES

1. P. G. Ciarlet, *The Finite Element Method for Elliptic Problem*, North-Holland, Amsterdam, 1978.
2. R. Lohner and P. Parikh, 'Three-dimensional grid generation by the advancing front method', *Int. j. numer. methods fluids*, **8**, 1135–1149 (1988).
3. R. Lohner and P. Parikh, 'Generation of 3D unstructured grids by advancing front method', *AIAA 26 Aerospace Sciences Meeting*, Reno Nevada, 1988.
4. J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O. C. Zienkiewicz, 'Finite element Euler computations in three dimensions', *Int. j. numer. methods eng.*, **26**, 2135–2159 (1988).
5. A. Golgolab, 'Mailleur tridimensionnel automatique pour des géométries complexes', *RR INRIA no. 1004*, 1989.
6. S. H. Lo, 'Automatic mesh generation and adaptation by using contours', *Int. j. numer. methods eng.*, **31**, 689–707 (1991).

7. J. Z. Zhu, O. C. Zienkiewicz, E. Hinton and J. Wu, 'A new approach to the development of automatic quadrilateral mesh generation', *Int. j. numer. methods eng.*, **32**, 849–866 (1991).
8. P. L. George, *Automatic Mesh Generation. Applications to Finite Element Methods*, Wiley, New York, 1991.
9. D. J. Mavriplis, 'An advancing front Delaunay triangulation algorithm designed for robustness', *Icase Report no. 92-49*, 1992.
10. M. A. Yerri and M. S. Shephard, 'Automatic 3D mesh generation by the modified-octree technique', *Int. j. numer. methods eng.*, **20**, 1965–1990 (1984).
11. P. L. George and F. Hermeline, 'Delaunay's mesh of a convex polyhedron in dimension d. Application to arbitrary polyhedra', *Int. j. numer. methods eng.*, **33**, 975–995, 1992.
12. P. L. George, 'Génération de maillages par une méthode de type Voronoï, *RR INRIA no. 1398*, 1991.
13. M. O. Bristeau, R. Glowinsky and J. Periaux, 'Scattering waves using exact controllability methods', *AIAA papers, 93-0460*, 1993.
14. P. L. George, 'Génération de maillages par une méthode de type Voronoï'. Partie 2: Le cas tridimensionnel', *RR INRIA no. 1664*, 1992.
15. J. A. George, 'Computer implementation of the finite element method, Stan-CS', *Ph.D.*, 1971.
16. P. L. George, F. Hecht and E. Saltel, 'Fully automatic mesh generator for 3d domains of any shape', *Impact Comp. Sci. Eng.*, **2**, 187–218 (1990).
17. P. L. George, F. Hecht and E. Saltel, Automatic mesh generator with specified boundary, *Comput. Methods Appl. Mech. Eng.*, **92**, 269–288 (1991).
18. P. L. George and E. Seveno, 'Génération de maillages par une méthode de type frontal', *RR INRIA no. 1725*, 1992.
19. F. Hecht and E. Saltel, Emc2: Editeur de maillages et de contours bidimensionnels, Manuel d'utilisation, *RT INRIA no. 118*, 1990.
20. D. G. Holmes and D. D. Snyder, 'The generation of unstructured triangular meshes using Delaunay triangulation, in *Numerical Grid Generation in Computational Fluid Mechanics*', Pineridge Press, Swansea, 1988, pp. 643–652.
21. M. S. Shephard, F. Guerinoni, J. E. Flaherty, R. A. Ludwig and P. L. Baehmann, 'Finite octree mesh generation for automated adaptive 3D flow analysis', *Numerical Grid Generation in Computational Fluid Mechanics' 88*, Miami, 1988.
22. N. P. Weatherill, 'The integrity of geometrical boundaries in the 2-dimensional Delaunay triangulation', *Commun. Appl. Numer. Methods*, **6**, 101–109 (1990).