INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 15, 985–997 (1992)

GRID GENERATION FOR 2D FLOW PROBLEMS

TAKEO TANIGUCHI

Engineering Science Department, Faculty of Engineering, Okayama University, Tsushima-naka, Okayama 700, Japan

KLAUS-PETER HOLZ

Institut für Strömungsmechanik und Elektron. Rechnen im Bauwesen, Universität Hannover, Hannover 1, Germany

AND

CHIKASHI OHTA

Ohbayashi-gumi Co. Ltd., Chuoh-ku, Osaka 540, Japan

SUMMARY

A grid generation method is proposed for an arbitrary two-dimensional domain. The method, based on the Delaunay triangulation, is modified so that it can be used as a grid generator for an arbitrary two-dimensional area with complex boundary geometry. Input data for the method are the co-ordinates of all nodes and the ordering of nodes on each boundary. Its efficiency is examined through a number of actual problems, and a numerical experiment clarifies that the grid generation requires a CPU time which is proportional to the number of nodes.

KEY WORDS Delaunay triangulation Grid generation 2D domain Triangular mesh Coastal engineering FEM

1. INTRODUCTION

A numerical approach to 2D flow problems requires a grid generation method with two functions: the first is the automatic generation of boundaries such as coastlines, islands and breakwaters, while the second is the subdivision of the domain with complex boundary geometry into elements.

The Delaunay triangulation can be an effective grid generator, but the method necessarily generates triangles not only inside the domain but also outside it. At the same time it may often fail to generate exact boundaries. These failures are originally caused by the Delaunay triangulation itself, since the method merely subdivides the convex domain occupied by the nodes into triangles which satisfy a geometrical condition.[1–4] Thus, the method must be modified so that it can truly generate the boundary of an arbitrary 2D domain and generate triangles only inside the domain. Weatherill proposed a method to construct the boundary configuration using the geometry of nodes,[5] while Baker proposed a weakened Delaunay criterion so as not to break the boundary configuration.[6]

The purpose of this paper is to modify the Delaunay triangulation proposed by Sloan[4] so that his algorithm can be applied not only to convex but also to arbitrary 2D domains. For this purpose we first consider how to recognize the boundary of an arbitrary 2D domain using nodes and also how to introduce it into the Delaunay triangulation. We then propose a new grid

0271–2091/92/210985–13$11.50

Received March 1992

generation method which is applicable to an arbitrary 2D domain with complex boundary geometry. The efficiency of the proposed method is surveyed through a number of test problems.

2. DELAUNAY TRIANGULATION

Two-dimensional Delaunay triangulation is a result of the geometry and the method uniquely decides a set of triangles for an arbitrary set of nodes in a plane. When the Delaunay triangulation is complete, no node may lie inside the circumcircle of any triangle. The geometry of the surrounding configuration of generated triangles is convex.[1,2] The geometrical characteristic of triangles generated by the Delaunay triangulation is appropriate for their use as elements for finite element analysis.

In order to apply the Delaunay triangulation as an actual grid generator for the finite element method (FEM), the method must be improved to be fast enough and also to be applicable to non-convex domains, since actual 2D domains generally show very complex boundary geometries and include several thousands of nodes. The triangulation method for 2D domains proposed by Sloan achieves a fast Delaunay triangulation by introducing the following items:[4]

(1) a supertriangle
(2) a bin-sorting technique
(3) a fast algorithm for searching a triangle
(4) a swapping algorithm.[7]

Item (1) is for simplification of the triangulation, item (2) is introduced for ordering the nodes to be set in the domain, and the ordering can save CPU time via item (3). The last item can accelerate the process of Delaunay triangulation, since the Delaunay triangles are obtained only by the comparison of two diagonals of a rectangle.

Sloan's algorithm generates triangles inside a convex polygon which includes all nodes prepared beforehand. Then the method can be used as a grid generator for an arbitrary convex 2D domain but not effectively for an arbitrary 2D domain. Our aim is to improve his method so that it can be applied as a grid generator for an arbitrary 2D domain.

3. GENERATION OF BOUNDARIES

3.1. Recognition of boundaries

Assume a 2D domain which is defined by a number of boundaries. Some of them define the exterior boundary, the others the interior boundary. Each of these boundaries can be expressed as a polygon if the nodes on the boundary are connected by lines. The Delaunay triangulation can be applied for these nodes on boundaries and generates triangles using these nodes, but the method may fail to generate exact edges, which should be located on the boundaries. That is, some edges of generated triangles are located such that they cross real boundaries, since the generation of all edges is determined by the co-ordinates of nodes. Thus our aim is to modify the Delaunay triangulation so that the method can truly generate all edges forming the boundaries.

The generation of triangles can be controlled by considering the location of nodes, since the Delaunay triangulation generates triangles according to the co-ordinates of nodes. Then it is possible to generate all edges located on boundaries by use of the Delaunay triangulation, but for this purpose all nodes on the boundaries must be carefully prepared in the domain.

The above consideration clarifies that the co-ordinates of nodes are insufficient for the generation of boundaries. We first consider how to recognize a boundary using nodes. This consideration is then introduced into the proposal of the boundary generation method in Section 3.2.

Assume that n nodes form a boundary. Then the boundary is expressed as a set of n line segments and each line segment is expressed as a pair of nodes. For simplicity we assume that nodes forming the boundary are ordered along one direction, i.e. clockwise or counterclockwise. We assume also that three nodes forming each triangle are ordered counterclockwise after the Delaunay triangulation, since this ordering is generally used for the description of finite elements. Then we can easily examine whether the boundary is exactly generated after the application of the Delaunay triangulation, because all line segments expressing the boundaries must be included among the edges of the generated triangles. The above discussion clarifies that the ordering of nodes on a boundary can provide effective information to recognize the boundary and that this additional information can be accepted by the Delaunay triangulation.

3.2. *Generation of boundaries*

In Section 3.1 we proposed additional information on nodes which is necessary for the recognition of a boundary. Now we consider the generation of a boundary using the Delaunay triangulation.

Let n be the number of nodes located on a boundary. For simplicity we assume that these nodes are ordered clockwise from 1 to n on the boundary. Assume that the Delaunay triangulation has already been applied for nodes 1 to i ($<n$) of the boundary and that all line segments showing the boundary from 1 to i have been generated.

Now we place the $(i+1)$th node in the plane. Then the procedure of Delaunay triangulation searches the triangle that includes the new node.[4] In the case where one node of the triangle is node 'i', the triangle is divided into three smaller triangles using the $(i+1)$th node and a new boundary segment connecting i and $i+1$ is necessarily generated. However, in the case where the triangle is not formed using node 'i', the line segment connecting i and $i+1$ cannot be generated. In this case we have to modify some of the triangles so that the boundary segment connecting i and $i+1$ is newly generated.

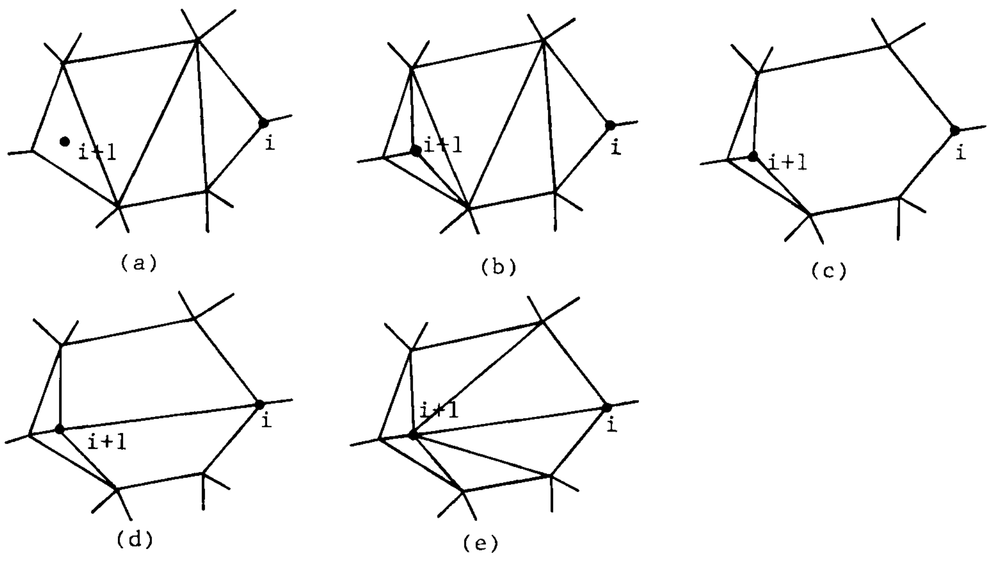

Figure 1. Generation of a boundary

The procedure of this modification is shown in Figure 1. First we divide the triangle including the $(i+1)$th node into three smaller triangles as shown in Figure 1(b). Successively we search all triangles located between the two triangles which are formed by 'i' and '$i+1$' respectively and we obtain a polygon by assembling these triangles as shown in Figure 1(c). The addition of a new edge connecting the two nodes 'i' and '$i+1$' subdivides the polygon into two parts as shown in Figure 1(d). The final stage is the subdivision of these two parts into triangles, the triangulation being done so as not to break the boundary segment newly generated between 'i' and '$i+1$' (see Figure 1(e)).

4. DELAUNAY TRIANGULATION INSIDE A 2D DOMAIN

4.1. Delaunay triangulation for nodes inside a domain

In Section 3.2 we showed how to generate the boundaries of a 2D domain. The nodes used in Section 3.2 are those which form the supertriangle and boundaries. Thus the process of generating all boundaries necessarily generates triangles inside the supertriangle. Then we are generally required to subdivide the triangles located inside the exterior boundary into smaller triangles using additional nodes inside the domain.

The grid generation for these nodes is also based on the Delaunay triangulation by Sloan,[4] but the following modification must be added to the method: the swapping algorithm for the exchange of the diagonal is used only when the original diagonal is not the boundary segment. An example is presented in Figure 2. The line connecting nodes a, b, c, and d in Figure 2 expresses the boundary segments. Assume that a new node denoted by x is set in the domain and that the circumcircle of the triangle (cbx) includes node a. If the swapping algorithm is used, the diagonal (bc) must be exchanged by another diagonal (ax). However, an exchange of diagonals should not be done in this case, since the diagonal (bc) is part of the boundary. In other cases the swapping algorithm is applied in order to improve the configuration of triangles.

4.2. Recognition of triangles inside a domain

After the triangulation for nodes on boundaries and additional nodes inside the domain, the supertriangle is divided into smaller triangles. This indicates that unnecessary triangles are generated outside the exterior boundaries and also inside the interior boundaries. These triangles must be removed.

Triangles located inside the supertriangle are classified into six categories:

(1) triangles having at least one node forming the supertriangle
(2) triangles formed only by nodes on exterior boundaries

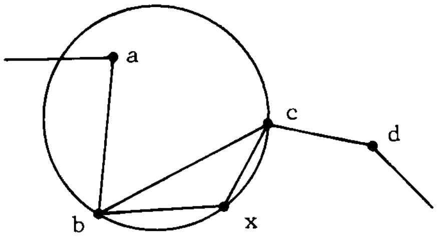

Figure 2. Triangulation of nodes on a boundary

(3) triangles having at least one node located inside the domain
(4) triangles formed by nodes on both exterior and interior boundaries
(5) triangles formed only by nodes on several interior boundaries
(6) triangles formed only by nodes on interior boundaries.

As is obvious from Figure 3, triangles of category (1) are located outside the domain. On the other hand, triangles of categories (3), (4) and (5) are located inside the domain. Triangles of categories (2) and (6) may be located outside or inside the domain. Thus we have to establish a judgement as to whether triangles are located inside the domain or not.

We assume that nodes on a boundary are ordered e.g. clockwise from a node arbitrarily selected among them. Figure 4 is an illustration of two triangles located outside and inside the domain respectively. The thick lines in Figure 4 denote boundary segments. We find that the ordering of two nodes on a boundary segment appears reversely for these two triangles if the three nodes forming a triangle are stored counterclockwise. Thus we can examine the location of these triangles from the ordering of nodes forming the triangles.

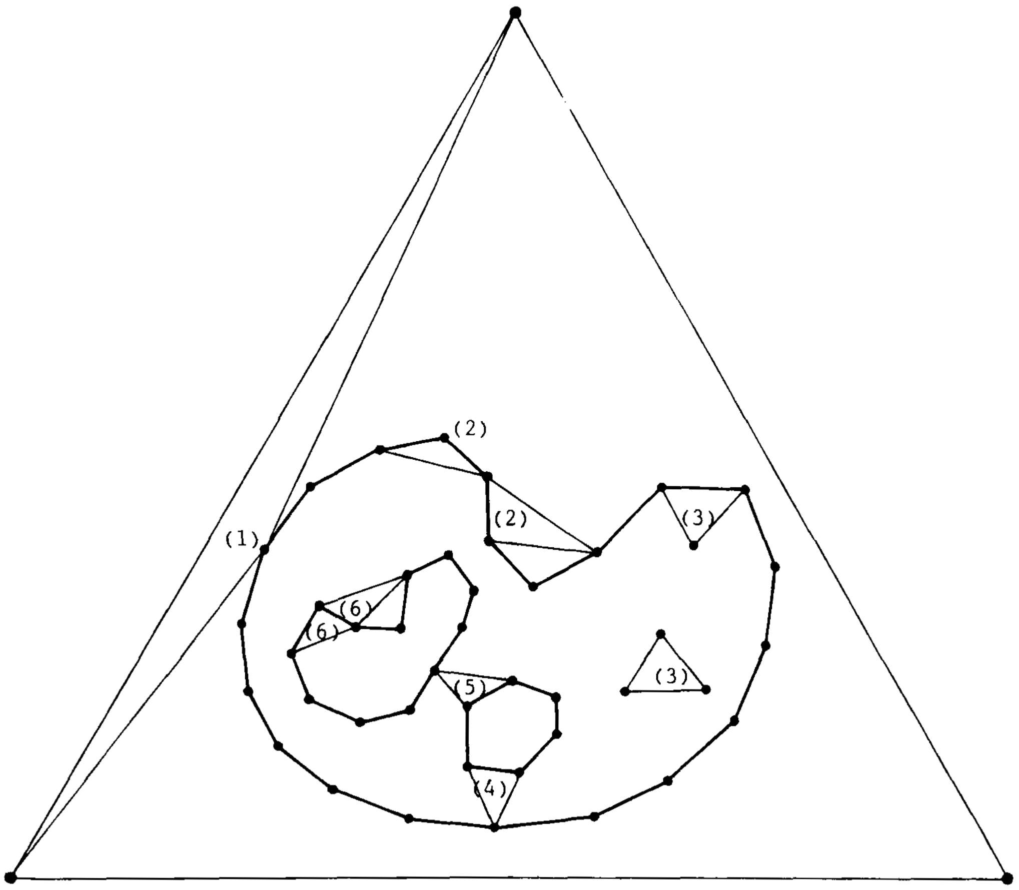

Figure 3. Classification of triangles

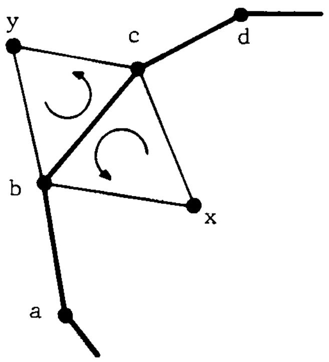

Figure 4. Triangles inside and outside a domain

4.3. Improvement of the geometry of generated triangles

The Delaunay method generates triangles for nodes which are prepared by the user and the geometry of generated triangles is wholly determined by the co-ordinates of nodes. It may happen that their geometry must be improved after the Delaunay triangulation.

One effective method for this purpose is the introduction of the Laplacian method, which relocates the co-ordinates of any node to the centre of gravity of all triangles related to that node. Repetition of this procedure can improve the geometry of triangles. However, this relocation should be applied only for nodes inside the domain, since nodes on boundaries must be fixed at their original positions to express the geometry of the domain. In the case where there are a number of fixed nodes inside the domain, they are also excluded from this relocation.

5. ALGORITHM OF MODIFIED DELAUNAY TRIANGULATION

The modification of the Delaunay triangulation proposed by Sloan has been explained in Sections 3 and 4. The procedure of the modified Delaunay triangulation is divided into two processes: coarse triangulation for nodes on boundaries and fine triangulation for nodes located inside the domain. The first process is the generation of all line segments on the boundaries and at the same time the generation of rough triangles not only inside but also outside the domain. The second process subdivides the rough triangles located inside the domain into small triangles.

The grid generator is effective for an arbitrary 2D domain which consists of a number of subdomains with interior boundaries. Each subdomain is defined by an exterior boundary and therefore some of the nodes on the exterior boundary are used for defining several exterior boundaries.

Before using this grid generation method, the user is required to prepare the following data:

(1) number of exterior and interior boundaries
(2) number of nodes on each boundary
(3) nodes on each exterior boundary (ordered clockwise)
(4) co-ordinates of all nodes.

The grid generation procedure consists of the following steps. Note that MTJ in the following steps is the element–node relation obtained by the grid generation and $N(j)$ is the number of nodes set on the jth boundary.

Coarse grid generation

Step 1. Input all data ($j = 1$).
Step 2. Place the supertriangle.
Step 3. Set $i = 1$.
Step 4. Place the ith node and find the triangle including it. Divide the triangle into three small triangles and store them in STACK.
Step 5. If the triangle includes the ($i - 1$)th node, then go to Step 6. Otherwise, go to Step 11.
Step 6. Pick out the last entry of STACK.
Step 7. Find an edge which is not a boundary segment and search another triangle adjacent to the edge.
Step 8. Compare two diagonals of the rectangle formed by these two triangles.
Step 9. If the old diagonal is longer than the other, form new triangles by exchanging the diagonals. Store them in STACK and go to Step 6. If the old diagonal is shorter than the other, store these two triangles in MTJ and go to Step 10.
Step 10. If STACK is not empty, go to Step 6. Otherwise, replace i by $i + 1$. If $i + 1 > N(j)$, go to Step 14. Otherwise, go to Step 4.
Step 11. Find all triangles located between the newly generated triangles which include the ith node and the triangle includng the ($i - 1$) th node.
Step 12. Remove all common edges for these triangles and form a polygon. Divide it into two polygons by adding an edge connecting $i - 1$ and i.
Step 13. Divide these two polygons into triangles and store them in STACK. Apply Steps 6–10 and modify the triangles.
Step 14. Apply Steps 11–13 by replacing $i - 1$ in these steps by 1 and go to Step 16.
Step 15. Replace j by $j + 1$ and go to Step 2.

Fine grid generation

Step 16. Apply Steps 4–10 for all nodes which are prepared in the domain. Step 5 must be removed.

Removal of unnecessary triangles

Step 17. Remove all triangles which are located outside the domain.

If necessary, the following step is applied after Step 17.

Improvement of geometry of triangles

Step 18. Apply the Laplacian method for improvement of the shape of generated triangles.

6. TEST PROBLEMS AND RESULTS

Thirteen test problems have been prepared for examination of the grid generation method proposed in this paper. The purposes of these tests are (1) to examine whether the method can generate the boundaries and (2) to survey the execution time necessary for generating a grid. The test of the grid generation method was done using a CADMUS workstation at Institut für Strömungsmechanik und Elektron. Rechnen im Bauwesen, Universität Hannover.

Table I summarizes the characteristics of the test problems and the results of the tests. All these problems are selected from maps and charts and the co-ordinates of nodes are prepared using the

Table I. Test problems and CPU times

Test no.	No. of nodes	No. of boundaries	Time (s)
1	279	4	6·4
2	653	4	11·7
3	1755	4	28·7
4	5141	1	80·5
5	2831	13	48·9
6	788	13	15·4
7	158	8	5·6
8	172	8	6·4
9	205	8	6·8
10	288	5	6·0
11	141	5	4·0
12	114	10	3·7
13	416	10	11·1

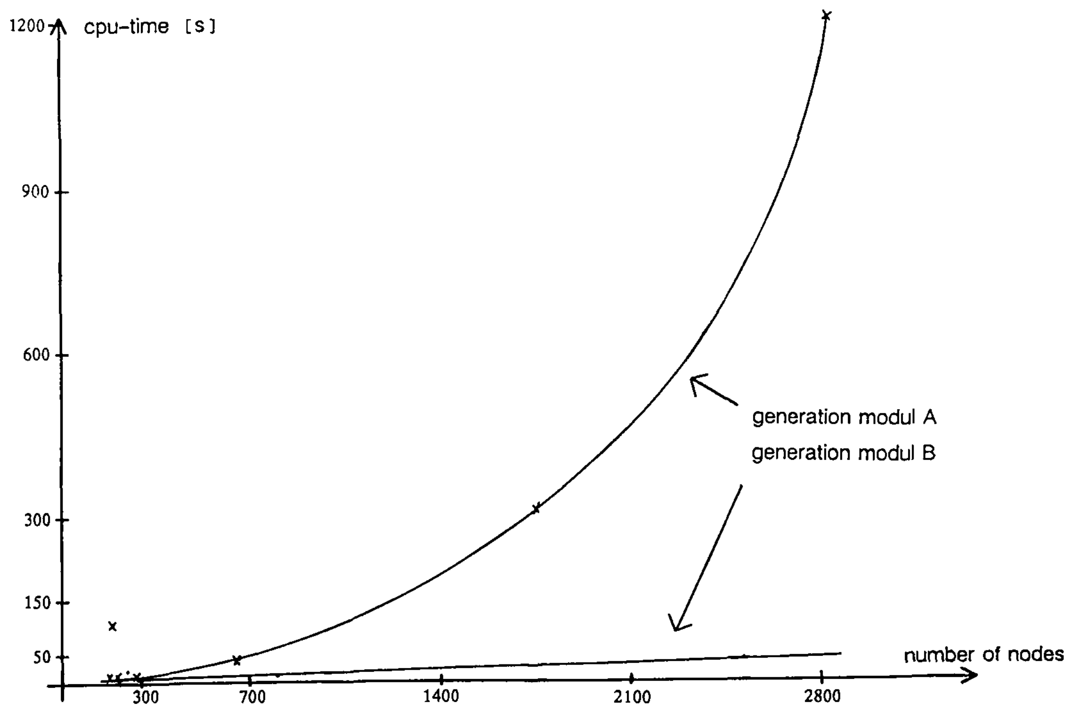

Figure 5. CPU time for proposed grid generation

digitizer. The number of nodes for these problems is between 114 and 5141 and the number of boundaries is between one and 13. The execution time required for their grid generation is also given in the table. The comparison of the execution time for these tests is summarized in Figure 5, where generation modules A and B indicate the result by the method which has been used previously in the Institut and the new one proposed in this paper respectively. The figure shows

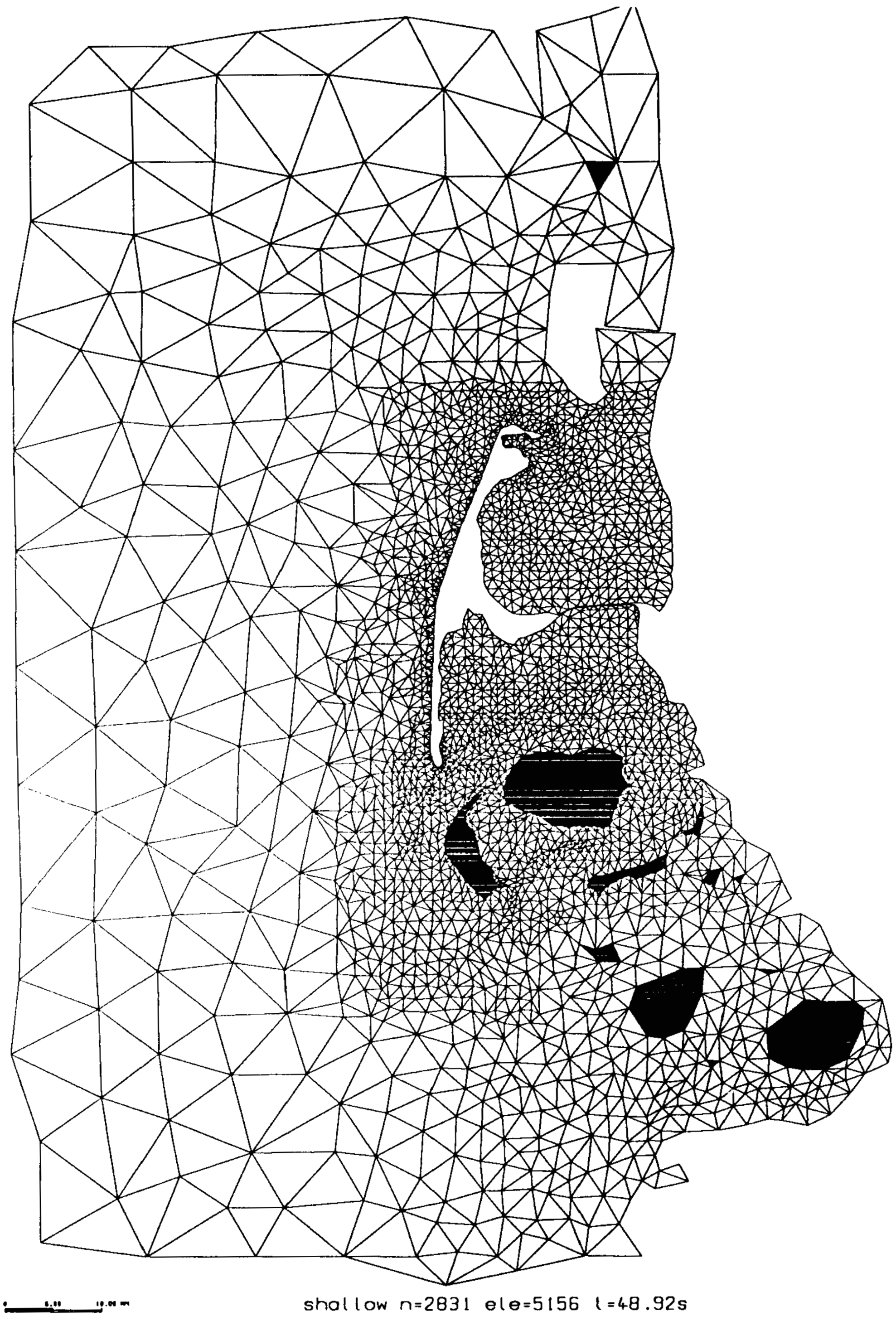

Figure 6(a). Grid of Test 5

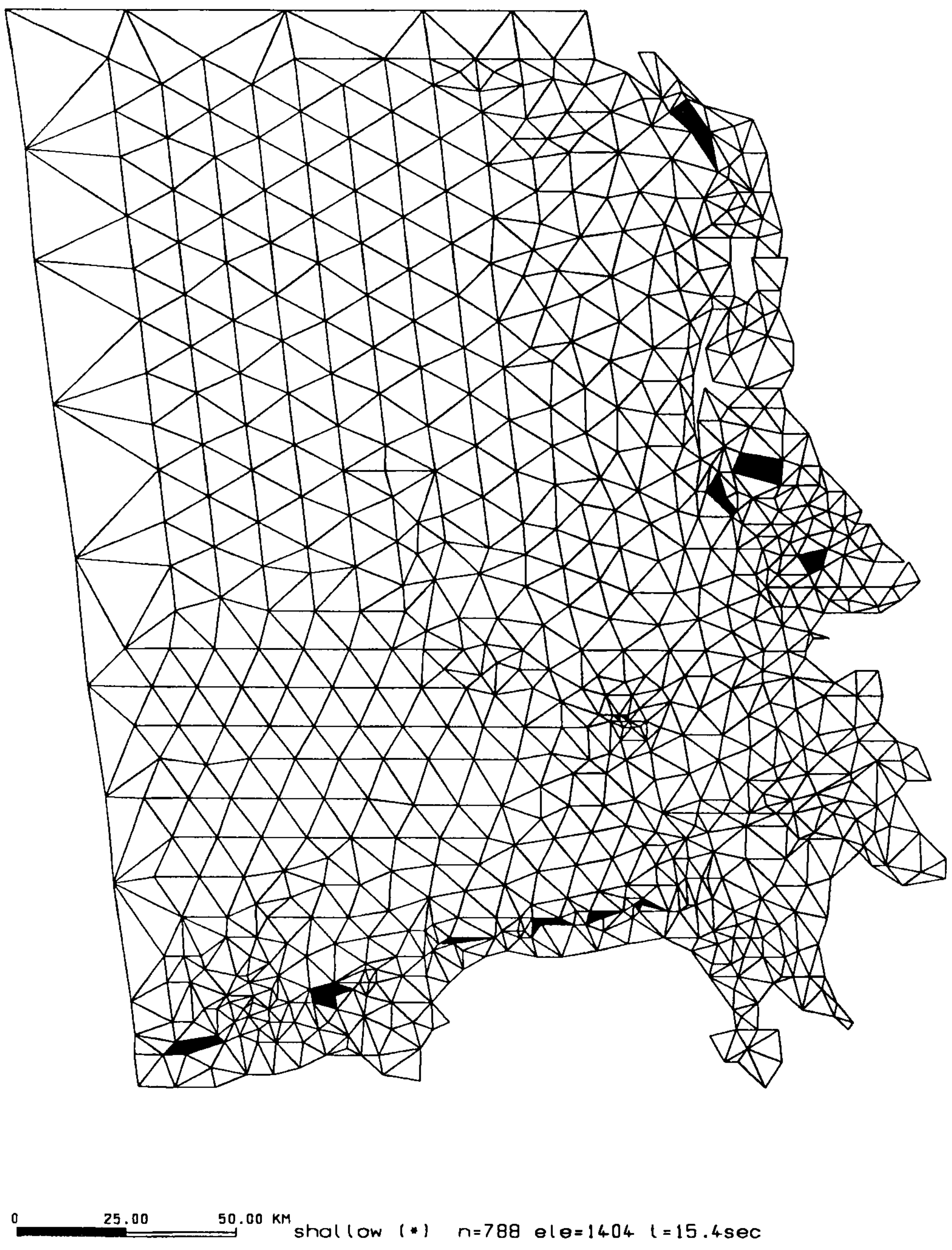

Figure 6(b). Grid of Test 6

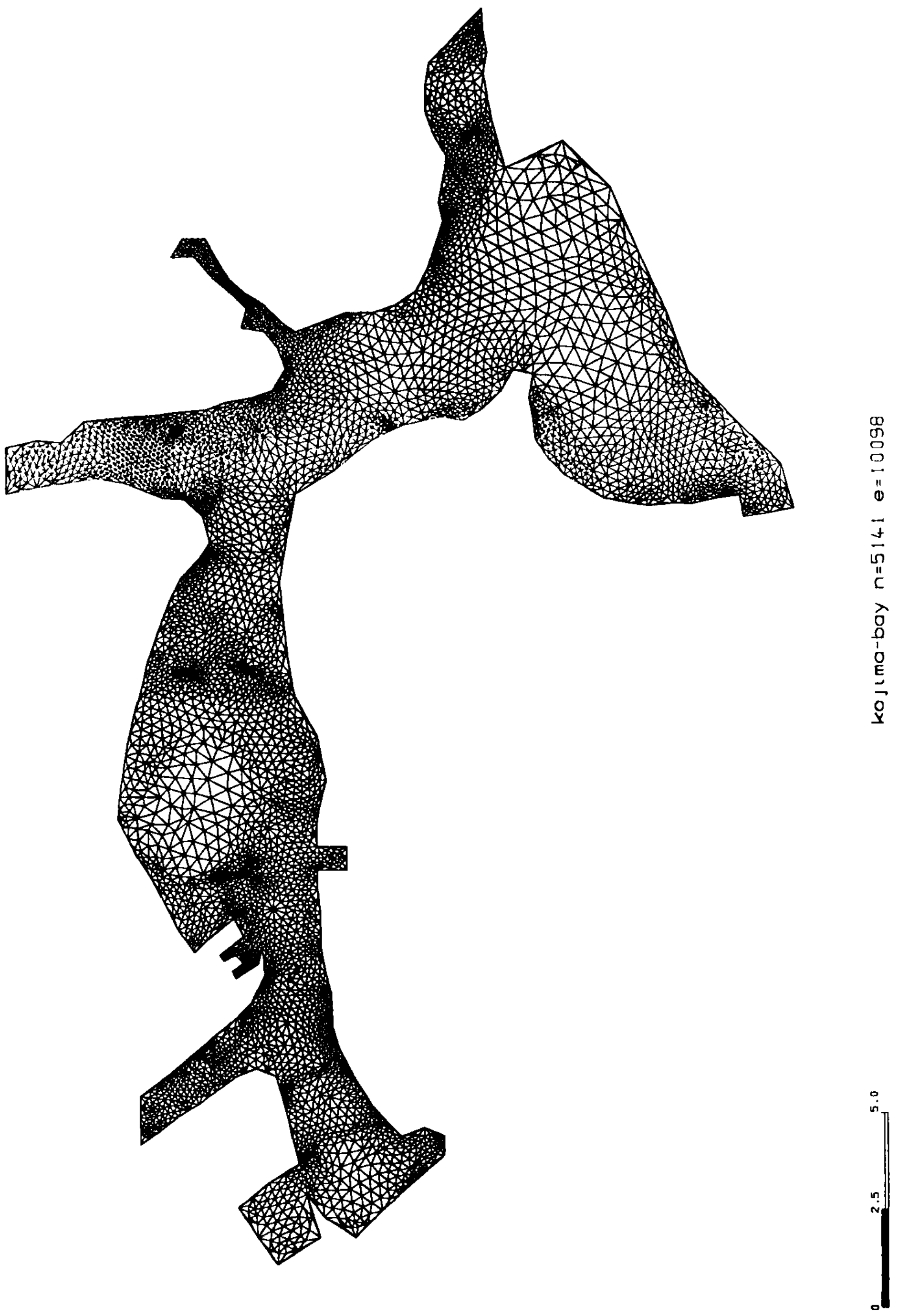

Figure 6(c). Grid of Test 3

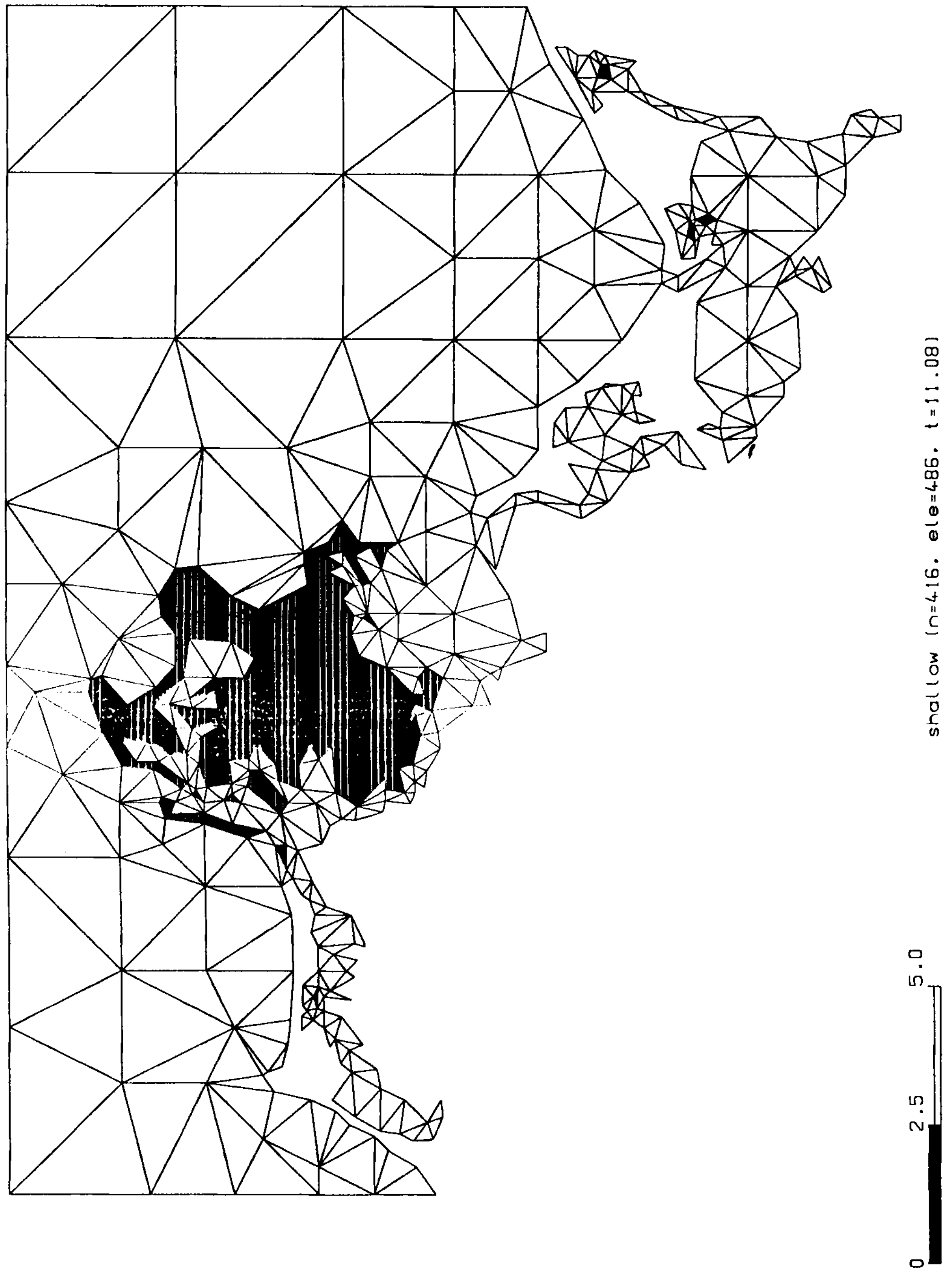

Figure 6(d). Grid of Test 13

that for our method the CPU time increases almost linearly with the increase in the number of nodes.

The four grids in Figures 6(a)–6(d) show the results of Tests 5, 6, 3 and 13 respectively. The proposed method was able to generate all boundaries for all cases. Black zones in these figures indicate the location of islands.

The improvement in the geometry of generated elements (i.e. Step 18 in the algorithm in Section 5) is not used, since the co-ordinates of all nodes are carefully placed and picked up by the digitizer.

The problems shown in this section have only one exterior boundary, but the method can be applied for 2D domains with several exterior boundaries.

7. CONCLUDING REMARKS

We have proposed a new grid generation method based on the Delaunay triangulation for 2D flow problems. The method is useful for the modelling of an arbitrary 2D domain enclosed by multiple boundaries with geometrical complexity. Numerical tests clarified that the method can generate boundaries and that the execution time increases almost linearly according to the number of nodes.

The proposed method can be thought as a kind of blocking method,[8] i.e. the coarse grid generation works by subdividing the whole area into a number of simple subareas, while the fine grid generation subdivides all subareas into smaller elements. However, the proposed method is easy to use and more flexible than the conventional blocking method, since the conventional blocking method requires more input data and has a restriction on the number of nodes placed on the boundaries.

In this paper we did not discuss any method to place nodes inside the domain, since all nodes not only on the boundaries but also inside the domain were prepared using the digitizer. In the case where only the geometry of all boundaries is given, the user must prepare a method to set nodes inside the domain.

ACKNOWLEDGEMENTS

The authors would like to thank Dip.-Ing. H. Nöthel and Dip.-Ing. A. Plüss of Universität Hannover for discussion of the grid generator proposed in this paper.

REFERENCES

1. A. Bowyer, 'Computing Dirichlet tesselation', *Comput. J.*, **24**, 162–166 (1981).
2. D. F. Watson, 'Computing *n*-dimensional Delaunay tesselation with application to Voronoi polytopes', *Comput. J.*, **24**, 167–182 (1981).
3. S. W. Sloan and G. T. Houlsby, 'An implementation of Watson's algorithm for computing 2-dimensional Delaunay triangulations', *Adv. Eng. Softw.*, **6**, 192–197 (1984).
4. S. W. Sloan, 'A fast algorithm for constructing Delaunay triangulation in the plane', *Adv. Eng. Softw.* **9**, 34–55 (1987).
5. N. P. Weathrill, 'A method for generating irregular computational grids in multiply connected planar domains', *Int. j. numer. methods fluids*, **8**, 181–197 (1988).
6. T. J. Baker, 'Tetrahedral mesh generation for the calculation of flows around complex configurations', Manuscript for *Second Nobeyama Workshop on Fluid Dynamics and Supercomputers*, 1987.
7. C. L. Lawson, in J. Rice (ed.), *Mathematical Software III*, Academic, New York, 1977, pp. 161–194.
8. T. Taniguchi, 'An interactive automatic mesh generator for the microcomputer', *Comput. Struct.*, **30**, 715–722 (1988).