INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 14, 861–881 (1992)

A STRUCTURED TRI-TREE SEARCH METHOD FOR GENERATION OF OPTIMAL UNSTRUCTURED FINITE ELEMENT GRIDS IN TWO AND THREE DIMENSIONS

S. Ø. WILLE

Department of Informatics, University of Oslo, PO Box 1080, Blindern, N-0316 Oslo 3, Norway

SUMMARY

A new method for generating finite element grids in two and three dimensions is developed. The method is based on a new search tree structure. The search tree is built upon triangles in two dimensions and tetrahedra in three dimensions. The density of elements can be varied throughout the computational domain. Efficient search algorithms for finding points in space and for finding the boundary of the domain have been developed. The speed of the grid algorithm will permit adaptive gridding during computation. The grid algorithm is generally applicable to both hydrodynamic as well as aerodynamic finite element computations. The technique has been used with success for gridding the North Sea–Skagerrak area.

KEY WORDS Grid generation Tri-tree Unstructured

INTRODUCTION

A main problem in computational aerodynamics as well as hydrodynamics has been to construct a grid describing the geometry of the computational domain. In both aerodynamics and hydrodynamics it is important to have a grid of variable density in space in order to resolve high gradients in the solution. The grid generator should also have the capability of local refinements based on the solution of the differential equations at each time step. To be able to refine the grid at each time step, the grid generator must be very efficient.

In recent years the automatic generation of unstructured grids has been paid intensive attention. Several algorithms building on different gridding techniques have been developed. The 'moving front' algorithm[1–4] starts with the boundary as a front and generates triangles or tetrahedra by selecting new points inside the geometry. As new triangles or tetrahedra arise, the front which is the base for generating new elements, moves into the geometry until the entire volume is triangulated. The advantage of the 'moving front' algorithm is that directional refinements can be achieved. In the 'Delauney' algorithm[5–8] points are substituted into the present triangulated volume. The algorithm is very simple in theory but severe problems are likely to occur during the triangulation process. For instace, the quality of the elements cannot be guaranteed and postprocessing of the elements is necessary to remove collapsing elements or to move points to obtain an acceptable grid. The major difficulty with 'Delauney' triangulation is to generate points inside the computational volume which do not introduce numerical difficulties.[7] The 'octree' method,[8,9] which is basically a tree search algorithm, has been used successfully as a basis for triangulation. The domain, which is divided into squares in two dimensions and cubes in three dimensions, forms a tree with pointers to the subdivisions. The leaves of the tree, which

0271–2091/92/070861–21$10.50

Received March 1991
Revised July 1991

are squares or cubes, are then triangulated in such a way that the neighbouring elements are compatible, valid mesh elements. This triangulation uses templates as well as element removal schemes. This method has been shown to be efficient and can be successfully coupled with 'Delauney' triangulation.[8,9]

The work of Rivara[10] describes an efficient two-dimensional triangulation algorithm based on iterative application of four-triangle conforming mesh refinement algorithms. One advantage of this algorithm is that a lower bound for the angles in the triangles can be estimated. However, it is not trivial to see how this algorithm is extended to three-dimensional triangulation.

The present method is also based on a tree search method similar to the quad-tree and oct-tree methods. Since the quad-tree and oct-tree algorithms produce triangles in two dimensions and tetrahedra in three dimensions, it seems more natural to build a tree structure on triangles and tetrahedra than on squares and cubes. The new tri-tree search method starts with a triangle or a tetrahedron which is subdivided into four new triangles or eight new tetrahedra respectively. The tri-tree structure then has pointers like the quad-/oct-tree. The main and essential difference is that the leaves in the tree consist of triangles and tetrahedra. The triangulation procedure of the tri-tree structure is then much simplified and will only cosist of connecting triangles *or tetrahedra* of different size. By introducing very mild restrictions on the tri-tree structure, which hardly affects the ability of local refinements, the triangulation procedure is very simple. The elements generated are optimal in the sense that the elements are equilateral triangles and tetrahedra, or at the interface of elements of different size, the equilateral triangles will be divided in two and the equilateral tetrahedra will be divided in two or four.

During the triangulation procedure an efficient search algorithm is needed for finding points in space. In the present work a lexical tree search algorithm of the point co-ordinates has proved to be very efficient. The boundary of the computational domain is found by means of an algorithm that detects whether a point lies inside or outside the computational domain. The tri-tree grid generation method has been tested by both two- and three-dimensional triangulation of the North Sea–Skagerrak area. This domain is rather complex since islands and large depth variations are present.

TRI-TREE DATA STRUCTURE

In the tri-tree search algorithm, equilateral triangles and tetrahedra are used as basic domains. The equilateral triangles and tetrahedra are then divided into equilateral triangles (Figure 1) and tetrahedra (Figure 2). In two dimensions an equilateral triangle is divided into four triangles. In three dimensions an equilateral tetrahedron is divided into eight tetrahedra. The ordering of successive divisions is organized as a tree structure as shown in Figures 3 and 4. The tree structure needs nine integers in two dimensions and 14 integers in three dimensions in order to keep the necessary information at each level of subdivision. The first integer in the tri-tree record contains the level of the division. The initial triangle and tetrahedron are at level 1. When a division occurs, the level number is increased by one. The level number therefore indicates the degree of division and all triangles or tetrahedra of equal size will have the same level number. When a division is terminal, the level number is given a negative sign. In addition to the level number, a point index to each of the corners of the structure is stored. This is not strictly necessary because the co-ordinates of each point can be calculated when they are needed. However, if the corner points are stored, the computing time is reduced considerably. The next positions in the structured record are pointers to the records of the divisions. When a triangle or tetrahedron is terminal, some of these pointers are used as pointers to the neighbouring triangles and tetrahedra instead. The last

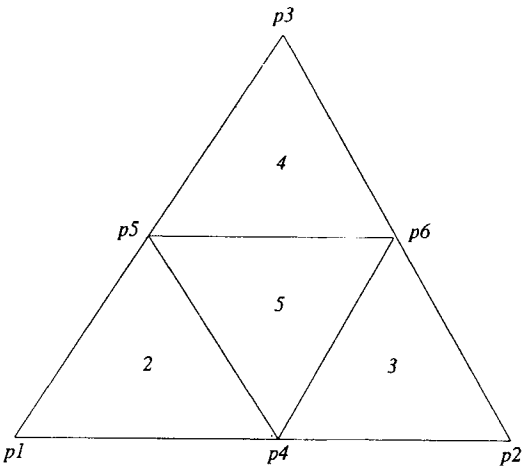

Figure 1. The initial equilateral triangle consists of the corners $T_1 = \{P_1, P_2, P_3\}$. This triangle is divided into four new equilateral triangles: $T_2 = \{P_1, P_4, P_5\}$, $T_3 = \{P_4, P_2, P_6\}$, $T_4 = \{P_5, P_6, P_3\}$, $T_5 = \{P_6, P_4, P_5\}$

	Level	Corners			Pointers to divisions				Parent
1	1	p1	p2	p3	2	3	4	5	0
2	-2	p1	p4	p5	5	0	0		1
3	-2	p4	p2	p6	0	5	0		1
4	-2	p5	p6	p3	0	0	5		1
5	-2	p6	p4	p5	2	4	3		1

Figure 2. The initial equilateral tetrahedron consists of the corners $T_1 = \{P_1, P_2, P_3, P_4\}$. This tetrahedron is divided into eight new equilateral tetrahedra:

$$T_2 = \{P_1, P_5, P_6, P_7\}, \quad T_3 = \{P_5, P_2, P_8, P_9\},$$
$$T_4 = \{P_6, P_8, P_3, P_{10}\}, \quad T_5 = \{P_7, P_9, P_{10}, P_4\},$$
$$T_6 = \{P_8, P_5, P_6, P_7\}, \quad T_7 = \{P_5, P_7, P_8, P_9\},$$
$$T_8 = \{P_6, P_8, P_7, P_{10}\}, \quad T_9 = \{P_7, P_9, P_{10}, P_8\},$$

integer in the record points to the record of the parent triangle or tetrahedron. It is therefore possible to perform both up- and down-searches in the tri-tree.

When searching for a tetrahedron which contains a given point, the tri-tree is traversed towards the root if the point is outside the present tetrahedron and towards the leaves if it is inside. Let the co-ordinates of the point be x, y, z, the area co-ordinates of the tetrahedron be L_k

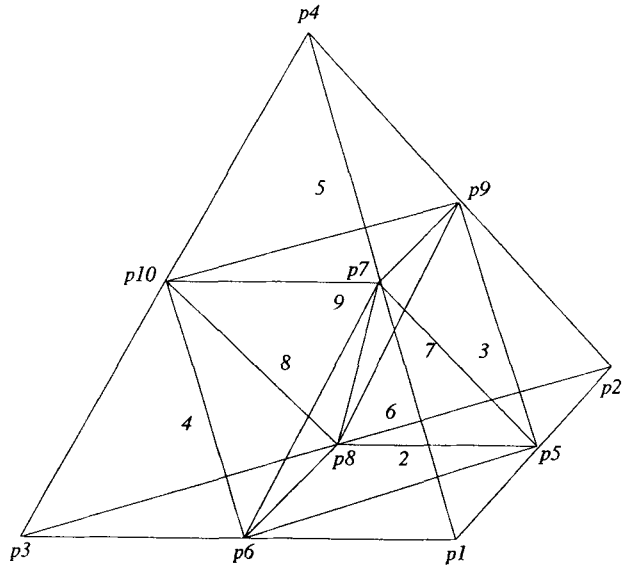

Figure 3. The information on the tri-tree structure in two dimensions is contained in a record consisting of nine integers. The first integer describes the level of refinement. When this integer is negative, it indicates a terminal triangle. The next three integers are the indices to the corners of the present division. If the refinement level integer is positive, the next four integers point to the triangles into which the triangle is refined. If the triangle is terminal, the following three integers are pointers to the neighbouring triangles. If one of these integers is zero, the triangle has no neighbour in that direction. The last index points to the parent triangle

and the co-ordinates of the nodes be x_k, y_k, z_k for $k=1, 2, 3, 4$. A simple test to exclude tetrahedra which do not contain the point is to decide if the point is outside the circumscribing box. If one or moe of the following conditions is valid, the point is outside the box:

$$(\max_k x_k < x), \quad k=1, 2, 3, 4, \tag{1}$$

$$(\min_k x_k > x), \quad k=1, 2, 3, 4, \tag{2}$$

$$(\max_k y_k < y), \quad k=1, 2, 3, 4, \tag{3}$$

$$(\min_k y_k > y), \quad k=1, 2, 3, 4, \tag{4}$$

$$(\max_k z_k < z), \quad k=1, 2, 3, 4, \tag{5}$$

$$(\min_k z_k > z), \quad k=1, 2, 3, 4, \tag{6}$$

If the point is inside the box, the following equation system has to be solved to determine if the point is inside the tetrahedron:

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{7}$$

If

$$0 \leqslant L_k \leqslant 1, \quad k=1, 2, 3, 4, \tag{8}$$

then the point is inside the tetrahedron.

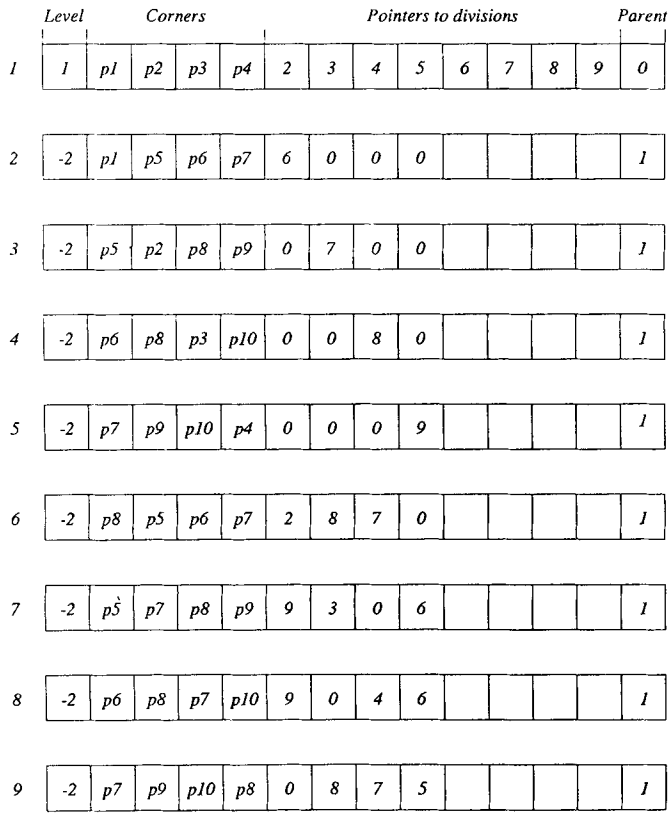

Figure 4. In three dimensions the record length of information for each division consists of 14 integers. The first integer describes the level of refinement. The next four integers are the indices to the corners of the present tetrahedron. If the refinement level integer is positive, the next eight integers point to the tetrahedra into which the tetrahedron is refined. If the tetrahedron is terminal, the following three integers are pointers to the neighbouring tetrahedra. The last index points to the parent tetrahedron

When a triangle or tetrahedron is divided, the midpoint on each line between the corners is calculated. This point may already exist if the neighbour has a larger level number. If a point does not exist, it is added to the list of points. In order to be able to search for and add points fast, the list is organized as a binary tree. The binary tree is sorted lexically on the point co-ordinates x, y, z. A point with co-ordinates x, y, z is defined to be less than a point with co-ordinates u, v, w if

$$\begin{aligned} & x<u \\ \text{or} \quad & x=u \text{ and } y<v \\ \text{or} \quad & x=u,\ y=v \text{ and } z<w. \end{aligned} \tag{9}$$

In order to find the neighbours of a tri-tree element, a search in the tri-tree is performed to find which tri-tree element contains a point slightly outside the edge or side of the present triangle or tetrahedron. The point to use in the tri-tree search is given by

$$\mathbf{P}=\mathbf{P}_g+(\mathbf{P}_g-\mathbf{P}_c)/d+\varepsilon(\mathbf{P}_g-\mathbf{P}_c). \tag{10}$$

In this expression $\mathbf{P}_g$ is the centre of gravity and $\mathbf{P}_c$ is a corner in the tri-tree element. The spatial dimension is d ($d=2$ or 3) and ε is a small constant which depends on the accuracy of the actual computer. If ε is zero, $\mathbf{P}$ is the point where the line from the corner $\mathbf{P}_c$ through the point of gravity hits the opposite edge or side. For small ε the point $\mathbf{P}$ will be on the line from the corner through the point of gravity slightly outside the tri-tree element. ε should be chosen so that the computer representation of

$$\mathbf{P}_g+(\mathbf{P}_g-\mathbf{P}_c)/d \neq \mathbf{P}_g+(\mathbf{P}_g-\mathbf{P}_c)/d+\varepsilon(\mathbf{P}_g-\mathbf{P}_c) \tag{11}$$

in only two or three of the least significant digits. The point $\mathbf{P}$ defined in this way is a point slightly outside the element edge or side opposite to the corner $\mathbf{P}_c$. A search in the tri-tree for a tri-tree element which encloses a point can either start at the root of the tree or at the location of the last search. If the points which are searched for are introduced in a random fashion, it will be most efficient to start at the root of the tree. When a search for the point $\mathbf{P}$ defined above is performed, the *a priori* knowledge is that the point is enclosed in an adjacent tri-tree element. The probability is therefore great that the adjacent tri-tree element belongs to the same subtree. If the tri-tree element belongs to the same subtree, it is faster to start the search at the present location, or even better at one level above the present location, than from the root of the tree. On average, experiments indicate that it is most efficient to start the search at one level above the present. At each level the the four triangles in two dimensions and the eight tetrahedra in three dimensions are explored to find which one contains the point.

DETECTION OF THE BOUNDARY SURFACE

In two dimensions the boundary is given by line segments. In three dimensions the boundary surface consists of triangles. During the triangulation procedure nodes are generated both inside and outside the computational domain. In a simple algorithm which is applicable for oceanographic problems, where the depth is given in equally spaced horizontal grid points, the x- and y-co-ordinates of a point can be used to look up the corresponding z-co-ordinate of the bottom. If a point is inside or outside the computational domain, then this is an easy decision. However, this algorithm is not general for other problems, e.g. gridding the domain around an aircraft, where the surface definition may consist of more than one point in some directions.

A general algorithm for deciding whether a point is inside or outside the domain is therefore developed. From each point an arbitrary line is constructed. The number of times which this line intersects the boundary elements is counted, and if the number of intersections is an odd number, the point is inside the domain; if the number of intersections is even, the point is outside. Problems with this algorithm occur if the line is parallel to a boundary element or if the line hits an edge or a node in the elements. Both these situations can be detected and a new direction of the line is drawn by a stochastic number generator. The equation of the line is given by

$$\mathbf{P}=\mathbf{P}_0-t\,\mathbf{D}, \tag{12}$$

where $\mathbf{P}_0$ is the point, $\mathbf{P}$ is a point on the line, $\mathbf{D}$ is the direction of the line and t is a scalar:

$$\mathbf{P}=\begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad \mathbf{P}_0=\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}, \qquad \mathbf{D}=\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}. \tag{13}$$

Initially the direction $\mathbf{D}$ is chosen to be $\mathbf{D}=[\frac{1}{3}, \frac{1}{7}, \frac{1}{11}]^T$. The initial direction selected here is quite arbitrary. This choice of $\mathbf{D}$ works well in most cases and few stochastic drawings are necessary.

The equation system for the intersection of the line and the plane given by the three nodes in the triangle is

$$\begin{bmatrix} x_1 & x_2 & x_3 & d_x \\ y_1 & y_2 & y_3 & d_y \\ z_1 & z_2 & z_3 & d_z \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ t \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}. \tag{14}$$

First, boundary elements which obviously do not intersect the line are excluded by a simple test. Use one of the co-ordinates of the corner, x_k, of the triangle to find the corresponding t.

$$t = \frac{x_k - x_0}{d_x}. \tag{15}$$

This t is used to compute the other two co-ordinates, which are the points on the line x_0. If only one of the conditions below is fulfilled, the line does not intersect the boundary triangle.

$$\begin{aligned} &y_k > y_0 + td_y, \quad k = 1, 2, 3, \\ &z_k > z_0 + td_z, \quad k = 1, 2, 3, \\ &y_k < y_0 + td_y, \quad k = 1, 2, 3, \\ &z_k < z_0 + td_z, \quad k = 1, 2, 3. \end{aligned} \tag{16}$$

If the above test fails, then it is possible tha the line intersects the boundary triangle and above equation system has to be solved. Two special cases have to be considered when the equation system is solved.

(i) The line is parallel to the boundary element.
(ii) The line hits a node or an edge.

In case (i) the corresponding diagonal of the matrix will become zero or close to zero during the forward elimination of the equation system. If the diagonal element is less than $\varepsilon_\alpha = 10^{-20}$, a new stochastic direction is generated with a random number generator available as a library function.

In case (ii), when the line hits an edge or a node of a boundary element, one or more of the area co-ordinates $L_k = 0$ for $k = 1, 2\ 3$. If the absolute value of at least one L_k is less than ε_α, a new direction of the line is generated stochastically in the same way.

The two tests described above have proved to be very robust against failure in practice. If failure does occur, which can be easily seen from the boundary plot of the domain, the failure is corrected by simply increasing ε_α. The line intersects the interior of the triangle if $0 < L_k < 1$ for $k = 1, 2, 3$.

All tri-tree elements are then checked and the elements with all nodes outside the computational domain are discarded and not considered for further processing.

TRI-TREE BALANCING

In order to obtain a smooth interface between triangles and tetrahedra with different levels, some rules are introduced on these interfaces. An arbitrary two-dimensional triangulation is shown in Figure 5. From this figure it is seen that each corner and centre triangle contains more than one node on one of its sides. This node distribution would lead to very narrow triangles, which would be disadvantageous when integrating the differential equations over the elements. The triangulation in Figure 5 is therefore balanced so that no triangle has more than one excess node on one of

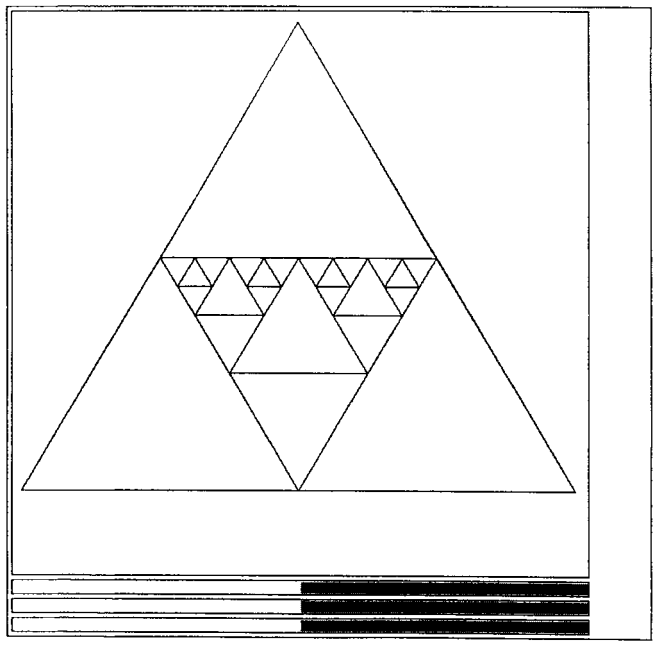

Figure 5. A triangle is first divided until the desired degree of refinement is obtained. The triangulation shown is arbitrary and it can be seen that such corner triangle has several nodes on one if its edges. A triangulation of this structure will result in very narrow triangles

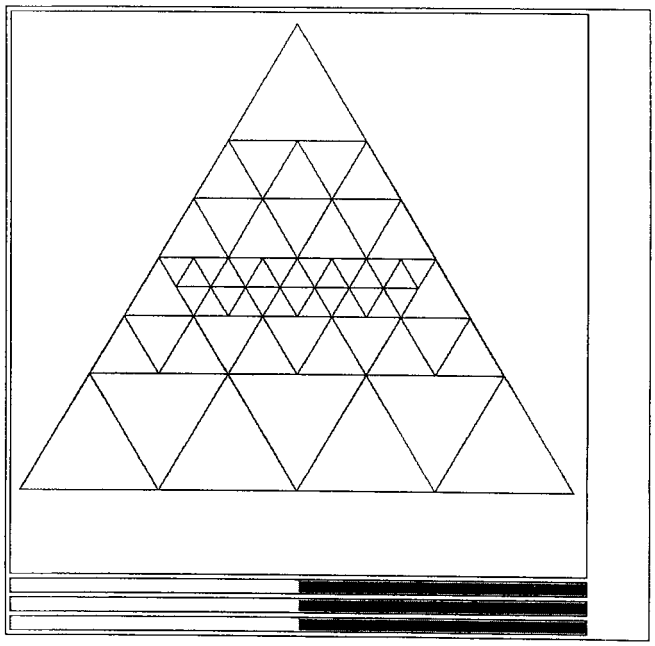

Figure 6. The triangular structure in Figure 5 is balanced so that no triangle has more than one node on one of its edges

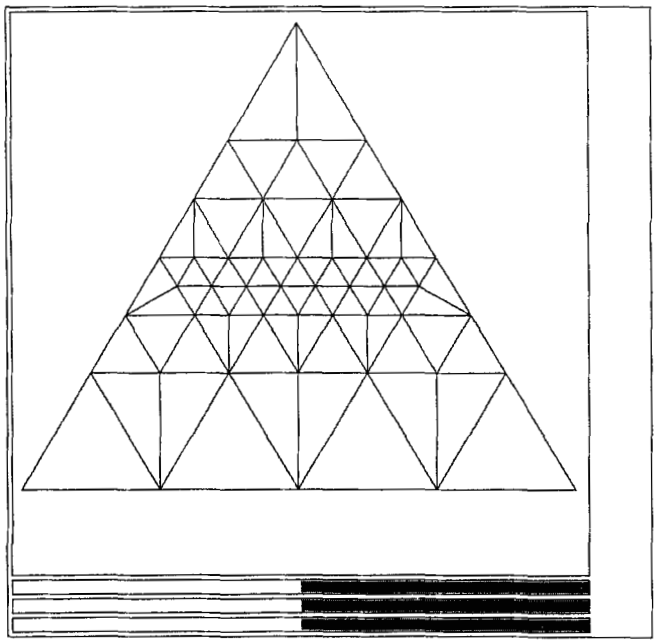

Figure 7. The triangulated grid from the tri-tree structure in Figure 6. The triangles which deviate most from equilateral shape are those having a right angle

its sides. Figure 6 shows the result of the balancing procedure. After balancing the tree, the tri-tree structure is triangulated as in Figure 7.

In three dimensions the balancing procedure is slightly more complicated, since more rules have to be applied to obtain a valid grid. Figures 8 and 9 show the two allowed ways of subdividing the equilateral tetrahedra into finite elements. In Figure 8 the tetrahedron is divided into two and in Figure 9 into four finite elements. These subdivisions show the finite elements which deviate most from the equilateral shape. An arbitrary three-dimensional triangulation is shown in Figure 10. After balancing this triangulation, the mesh of tetrahedra has the structure shown in Figure 11. Figure 12 show the triangulation of the balanced three dimensional tri-tree structure.

Let L_p be the level number of equilateral triangle or tetrahedron p and let L_n be the level number of its neighbours n. Let n_l and n_g be the numbers of neighbours with level number less than and greater than tree element p respectively. The refinement algorithm is then given by

Algorithm 1

1	if $L_p < L_n - 1$	Refine(p)	
2	if $L_p > L_n + 1$	Refine(n)	
3	if $L_p > L_n$	$n_g = n_g + 1$	
4	if $L_p < L_n$	$n_l = n_l + 1$	(17)
5	if $n_l > 1$	Refine(p)	
6	if $n_l > 0$ and $n_g > 0$	Refine(m)	

In line 1 the level of each tree element is compared to the level of its neighbours. If the level of a tree element is less than the level of a neighbour minus one, then the element is refined.

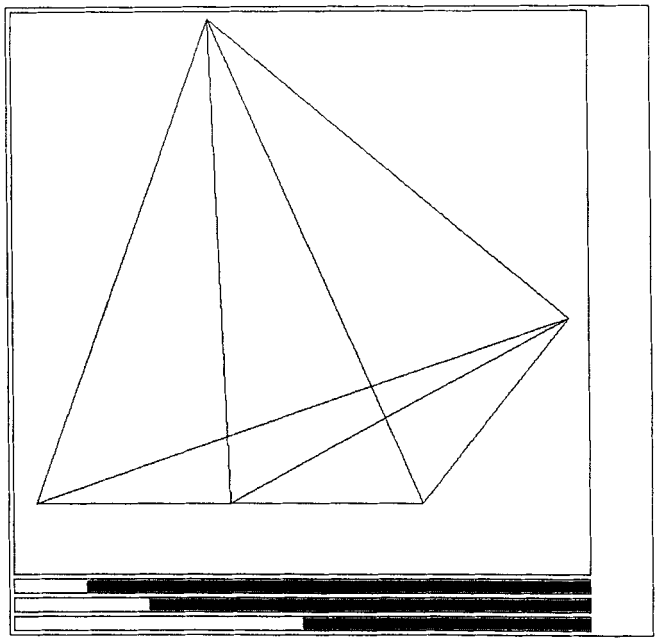

Figure 8. An equilateral tetrahedron may be divided into two finite elements depending on the division of the neighbouring elements

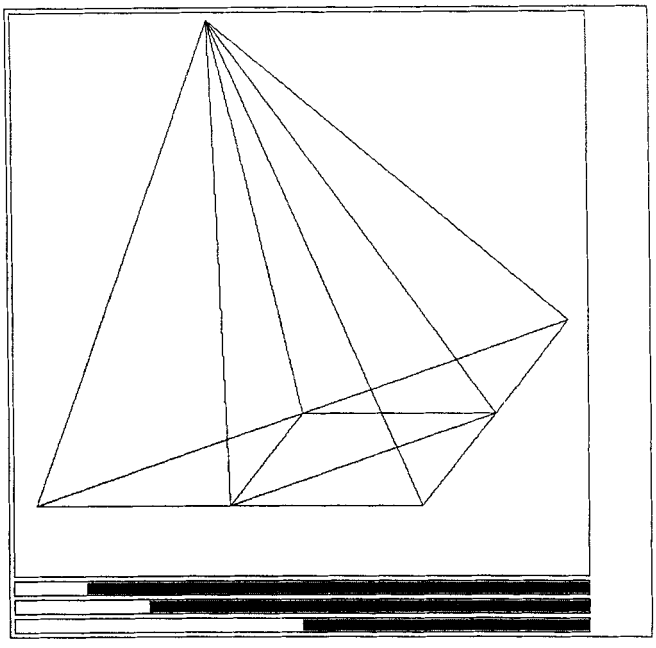

Figure 9. An equilateral tetrahedron may also be divided into four finite elements. Although the shape of these elements deviates most from equilateral shape, they still have an excellent shape well suited for integration

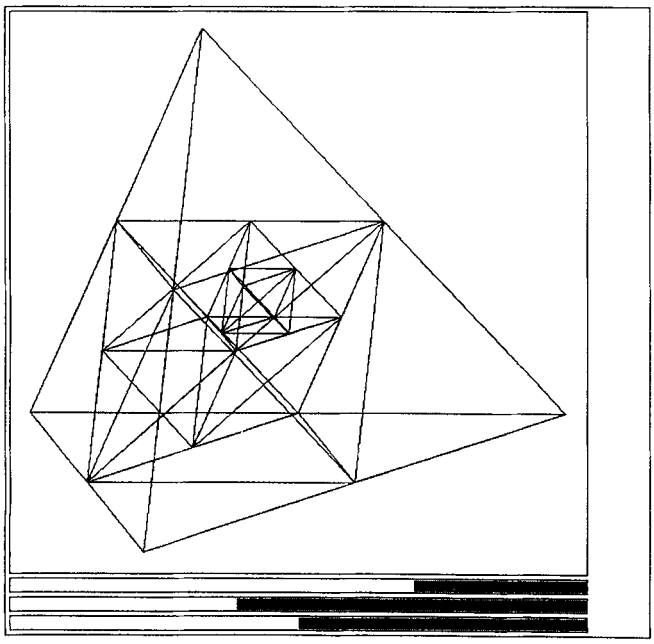

Figure 10. An arbitrary tri-tree structure of tetrahedra

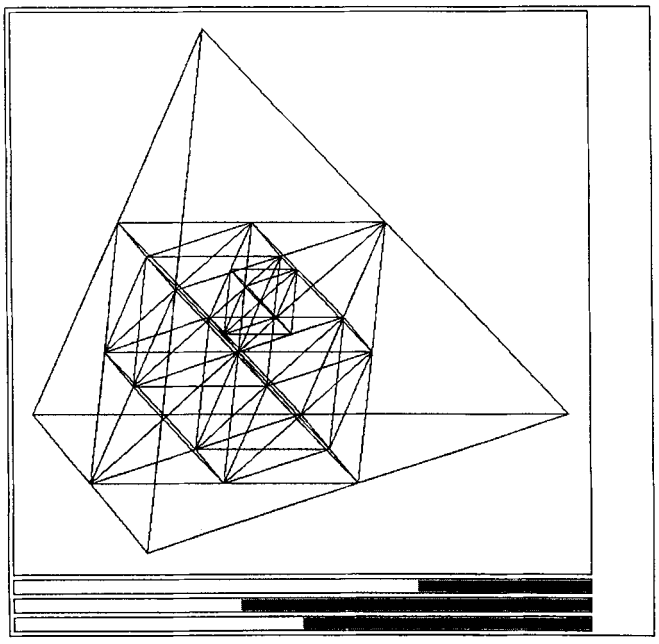

Figure 11. The tri-tree structure of the tetrahedra shown in Figure 10 is balanced similar to the balancing in two dimensions. However, the balancing procedure in three dimensions is more complicated

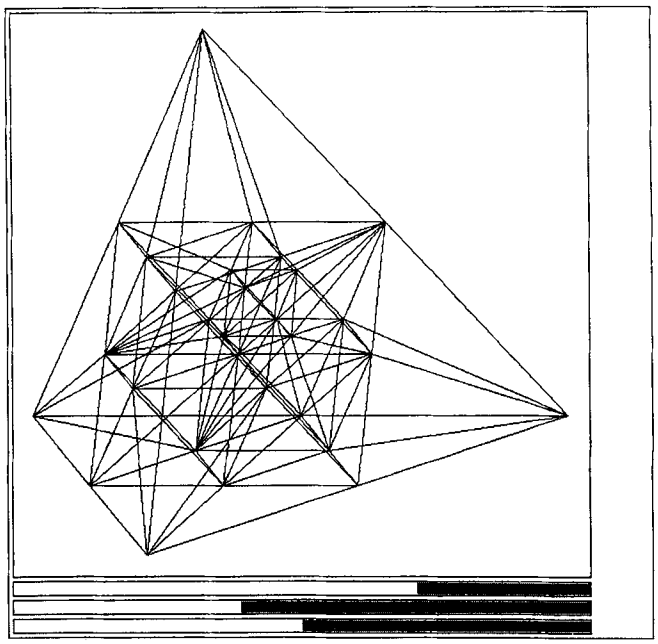

Figure 12. Finite element tetrahedra generated from the balanced tri-tree structure in Figure 11

In line 2 a neighbouring tree element is refined if the level of the actual element is greater than that of the neighbouring element plus one. In lines 3 and 4 the numbers of neighbouring elements which have greater and smaller levels respectively are counted. In line 5 a tree element is refined if more than one neighbour has a smaller level. In line 6 a neighbouring tree element with the smallest level number is refined if an element has neighbours with both smaller and greater level numbers.

In three dimensions more rules have to be applied to ensure the generation of valid finite elements. Let n_m be the number of midpoint nodes at the edges of the tetrahedron:

Algorithm 2

```
7     if n_m ≠ 1 and n_m ≠ 3
8       {
9         if L_p < L_n   Refine(p)
10        if L_p > L_n   Refine(n)
11      }                                                    (18)
```

After Algorithm 1 is applied there will only be nodes at the midpoints of the edges of the tetrahedra. If the number of midpoint nodes is different from one and three, the element itself, or the neighbouring elements which also contain one of these nodes, is refined depending on which has the smallest level number.

The additional rules applied in three dimensions can also be applied in two dimensions, although it is not necessary. However, if these rules are included, the algorithm will be identical in both two and three dimensions.

Both Algorithms 1 and 2 are applied iteratively. First Algorithm 1 is repeated until no refinement occurs. Then Algorithm 2 is applied. If a refinement occurs, Algorithm 1 has to be

applied once more. The whole balancing procedure terminates when no refinements occur either in Algorithm 1 or in Algorithm 2.

TRIANGULATION OF THE TRI-TREE

After the balancing procedure the tri-tree is valid for triangulation. In two dimensions there is at most one node at the midpoint of one of the edges of the triangles. This tri-tree triangle is divided into two finite element triangles as shown in Figure 7 (see also Figure 17). In three dimensions the situation is more complex. Each equilateral tri-tree tetrahedron can have either one node on one of the edges as in Figure 8, or three nodes at the edges of one of the sides as in Figure 9. If there is one node at one edge, the tetrahedron is divided in two as in Figure 8. If there are three nodes at the edges of one side, the tetrahedron is divided into four finite element tetrahedra as in Figure 9. The triangulation procedure is only applied to tri-tree elements which are inside the computational domain. When the tri-tree is triangulated, the finite elements are kept in a finite element structure and the tri-tree structure is stored to be used later when the grid is adapted to the solution.

TRIANGULATION OF THE NORTH SEA–SKAGERRAK

The topology of the North Sea–Skagerrak is available as an equidistant regular grid in the horizontal plane with the depth given at the grid points. The distance between grid points is $dx = dy = 6173{\cdot}33$ m. The rectangular grid is first converted to triangular finite elements describing the bottom boundary. Elements which have all nodes on land are excluded. The coast line and the boundary line in the North Sea and the English Channel are described by line segments found from the topology matrix.

The triangulation in two dimensions starts with the boundary shown in Figure 13. The boundary is circumscribed by an initial triangle which is the root of the tri-tree. The level of refinement is given on this boundary and the initial triangle is refined until the desired level of refinement is obtained. For every point on the boundary the tri-tree is searched to find the terminal containing the boundary point. By looking at the refinement level of this triangle, a decision is made on further refinement. This process is an iterative process which continues until the triangles containing the boundary points are refined satisfactorily. However, this refinement is not sufficient since there may be boundary tri-tree triangles which do not contain boundary points. Therefore all tri-tree elements which have at least one node inside and one node outside the boundary are correspondingly refined. This refinement process is also an iterative process.

Figure 14 shows the refinement of the initial triangle along the boundary. The result obtained by removing the elements outside the boundary is shown in Figure 15. As seen from this figure, there are several tri-tree triangles which have more than one node on one of their side edges. To obtain a tri-tree structure with at most one node on one of the edges of the triangles, the tri-tree is balanced. The balanced tri-tree is shown in Figure 16. The balanced tri-tree is the basis for the final triangulation. The tri-tree elements which have neighbours with the same or smaller level numbers are used directly as finite elements. The tri-tree elements which have elements with a greater level number on one side are divided into two finite elements. The final triangulation is shown in Figure 17.

A contour map of the North Sea–Skagerrak is shown in Figure 18. The depth in the North Sea–Skagerrak is seen to vary considerably, especially along the coast of Norway. The contour map can also be presented in three dimensions as in Figure 19. The depth is scaled with respect to the horizontal co-ordinates. From this angle of view the variation of depth is even clearer than in

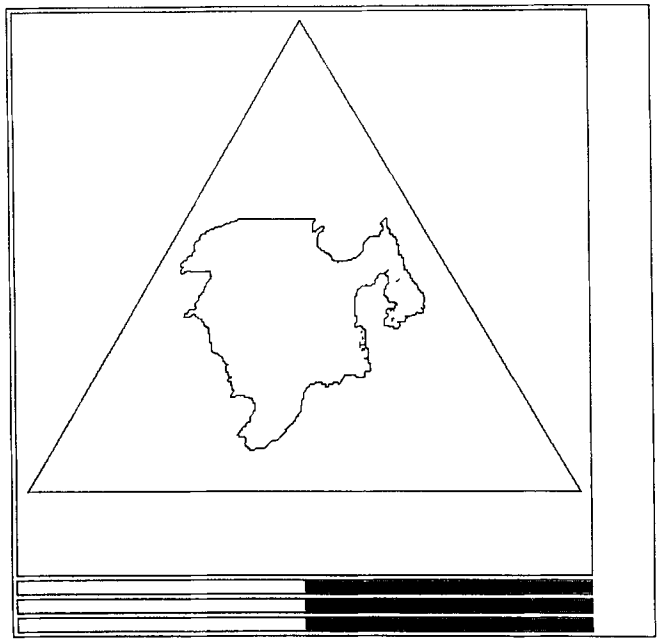

Figure 13. The two-dimensional North Sea–Skagerrak area which is to be triangulated. The initial triangle includes the area to be triangulated. This initial triangle is refined until the desired degree of refinement is achieved

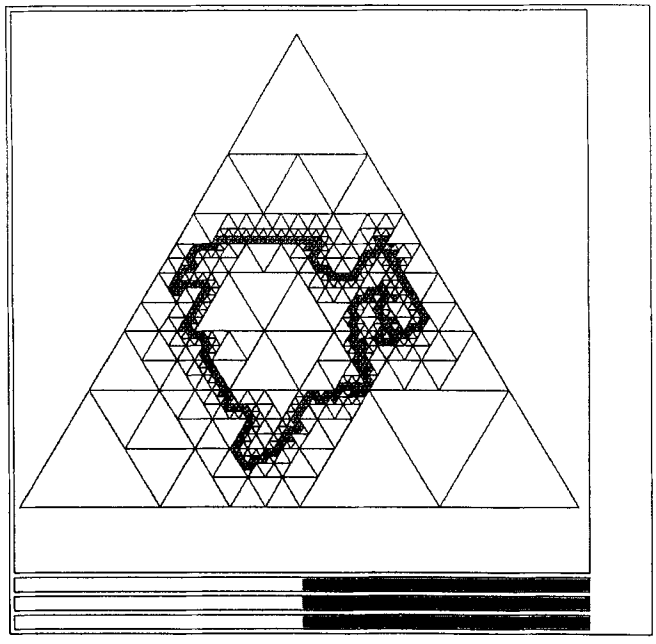

Figure 14. A tri-tree structure of the triangulated North Sea–Skagerrak area. The level of refinement is specified at boundary contours

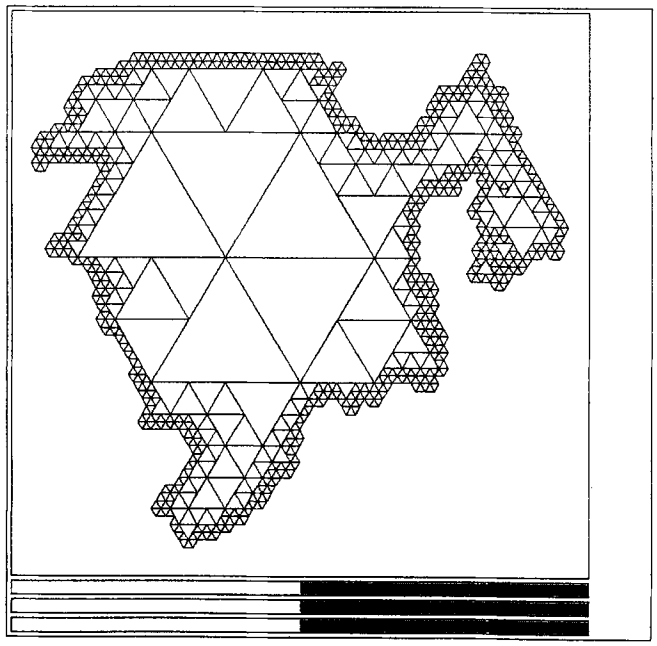

Figure 15. A magnified tri-tree structure of the North Sea–Skagerrak area with tri-tree elements outside the computational domain removed

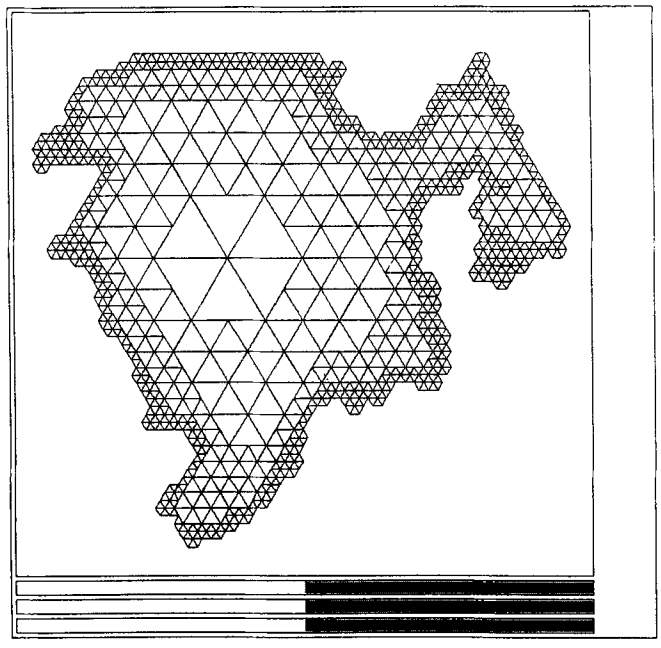

Figure 16. The tri-tree structure of the North Sea–Skagerrak area is balanced so that no tri-tree element has more than one node at the midpoint of one of the edges

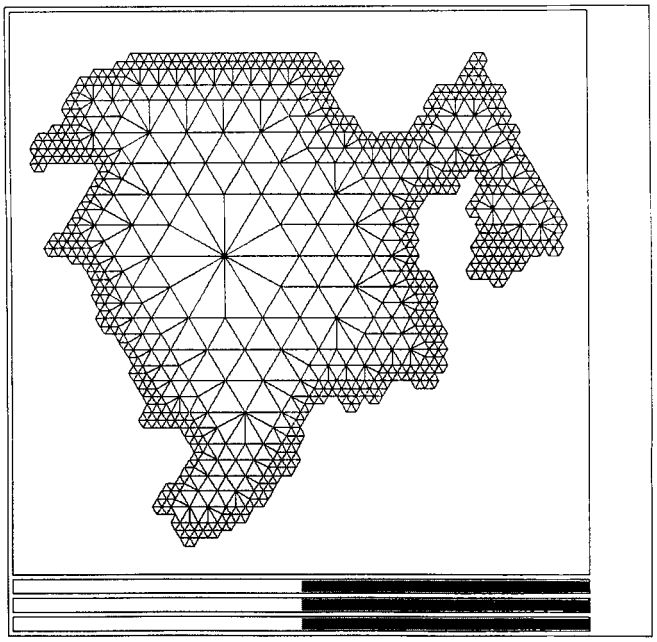

Figure 17. The triangular finite elements of the tri-tree structure in Figure 16

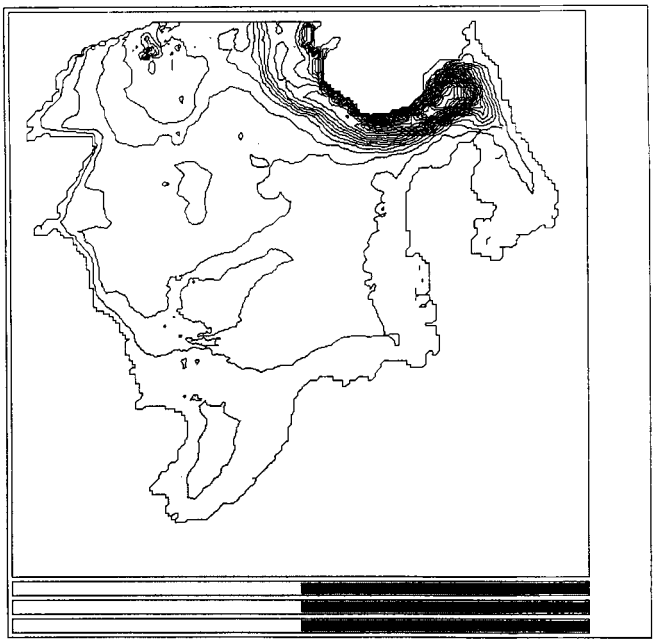

Figure 18. The iso-level contours of depth in the North Sea–Skagerrak area. The depth in the Norwegian Deep is seen to be much greater than in other parts of the area

Figure18. In Figure 20 the three-dimensional contour map is included in the initial tetrahedron. The level of refinement is specified at the bottom and at the inflow-outflow boundaries. The triangulation follows the same procedure as in two dimensions. First the tri-tree elements which include boundary points are refined iteratively to the specified level. Since not all boundary elements include such points, the tri-elements which cut the boundary surface are then refined iteratively. Figure 21 shows the boundary surface of the triangulated volume with a specified refinement level equal to seven. This means that the size of the edges of the finest tri-tree elements is $1/2^6$ times the length of the edge of the initial tetrahedron. The general formula for this length is

$$d = D/2^{n-1}, \tag{19}$$

where d is the size of the edge of the finest element, D is the length of the side of the initial tetrahedron and n is the level of refinement. The boundary surface with a refinement level of eight and nine is shown in Figures 22 and 23 respectively.

DISCUSSION

In the present work a new unstructured grid generation algorithm is developed. The algorithm allows the generation of finite elements of different sizes. It is therefore possible to have a finer grid at the boundaries as demonstrated in Figure 17 and to adapt the grid to the solution of the differential equations. A grid which is adapted to the topology will then serve as a basis grid for further refinement. When the solution of the differential equations is found, the basis grid can be further refined to resolve interesting gradients in the solution.

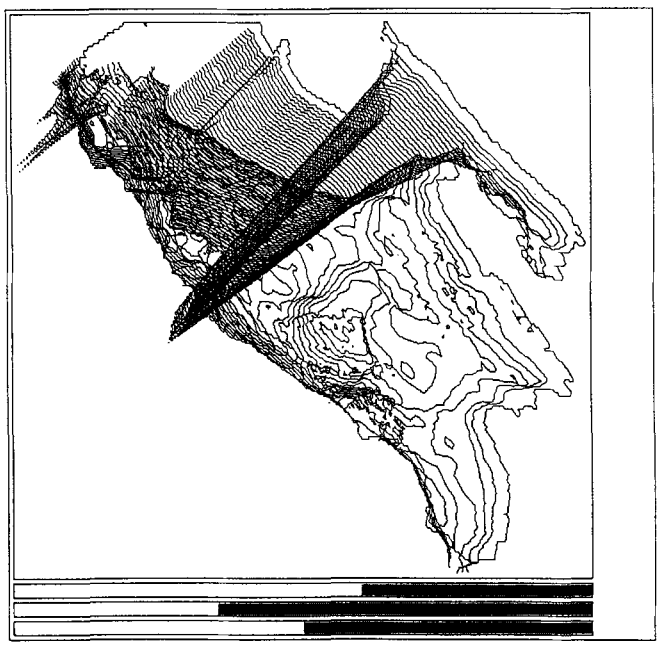

Figure 19. A three-dimensional representation of the same iso-level contours as in Figure 18, showing even more clearly the contrast in depth between the Norwegian Deep and the rest of the area

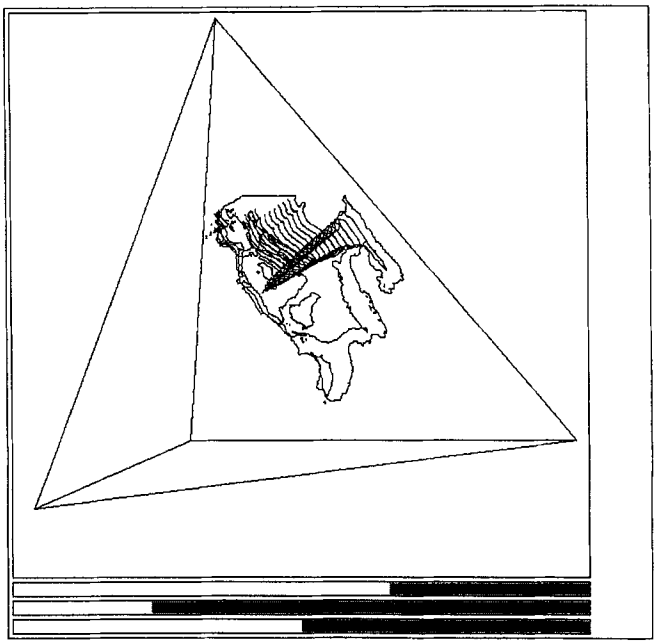

Figure 20. A three-dimensional representation of the volume to be filled with tetrahedra contained in the initial tri-tree tetrahedron

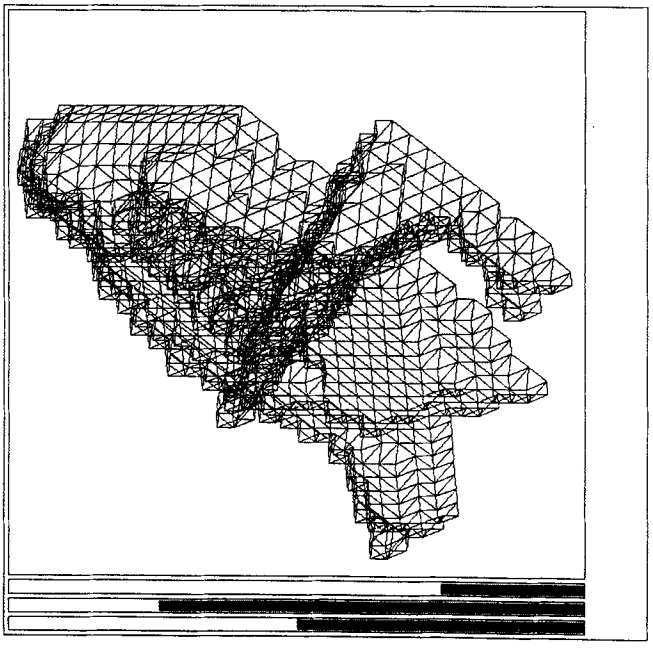

Figure 21. A coarse three-dimensional tetrahedrization of the computational domain. The level of refinement at the boundary surfaces is seven. Only those sides of the tetrahedra which coincidence with the boundary are shown

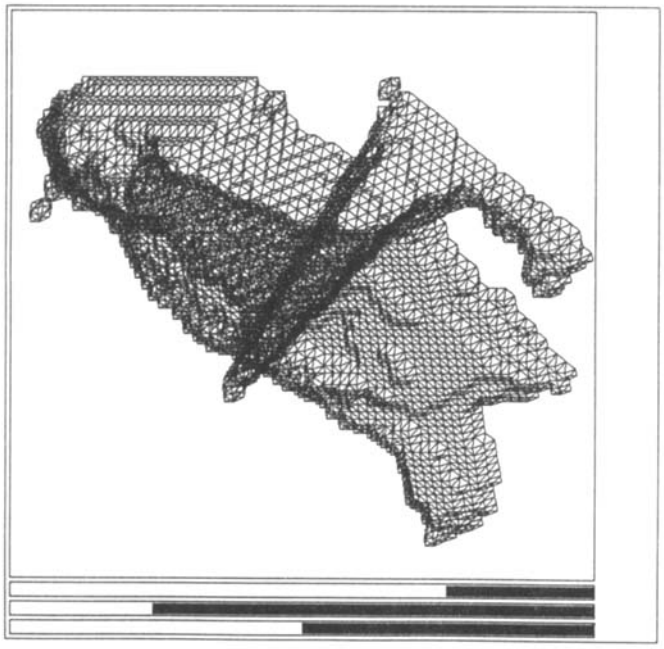

Figure 22. A tetrahedrization with higher resolution than shown in Figure 21. The level of refinement is eight

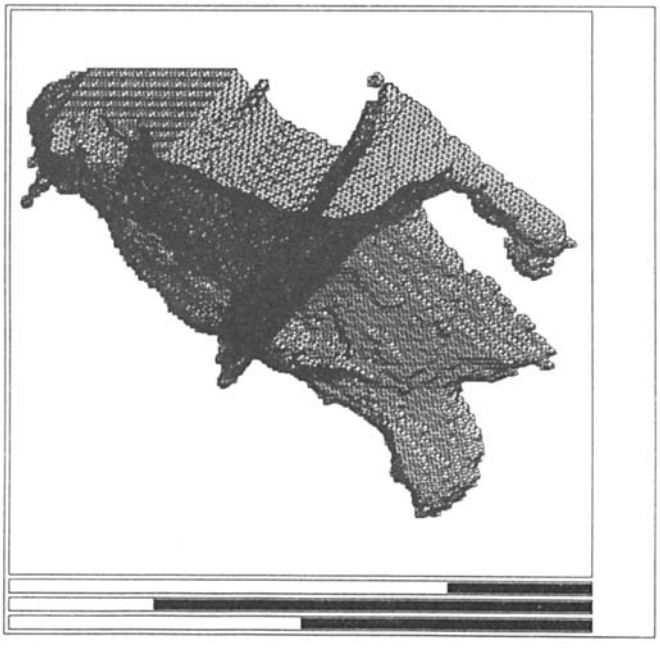

Figure 23. The tetrahedrization of the North Sea–Skagerrak with the highest reslution computed. The refinement level is nine. With respect to the accuracy and resolution of the background data, an increase in refinement level would not reveal more details of the topology

The refinement criterion in the present work is to specify the level number at points on the boundary. Other refinement criteria can easily be introduced. Such criteria might be to specify the level numbers at points inside the computational domain or to relate the size of the gradients in the solution to the level number.

At the interface between tri-tree elements of different size the equilateral tri-tree elements in two dimensions are divided into two and in three dimensions into two or four. The grid generation algorithm is optimal with respect to integration properties for a grid with variable degree of refinement. The algorithm guarantees that no element is distorted more in shape than shown in Figures 7–9.

In the quad-/oct-tree algorithm, templates are investigated for each terminal cell in order to find a triangulation to ensure compatibility between elements.[8] When elements degenerate, 'Delauney' points are substituted to obtain better shaped elements. When triangulation has been performed, a Laplacian filter is used to obtain a better geometry of the elements. The present method seems simpler and more efficient since there is no need for template matching. The elements obtained by the present algorithm are geometrically well shaped. The deviations from equilateral shapes of the triangles and tetrahedra are very mild and caused by the local refinements of the mesh. A smoothing filter would ensure greater smoothness in the mesh with respect to the area of each element. However, at interfaces between elements of different size a smoothing filter would also change the angles between the edges and sides inside these elements. This change would probably be unfortunate for the integration procedure of the finite element equations.

The disadvantage with 'Delauney' triangulation, described by Baker,[7] is the possibility of algorithmic failure. When algorithmic failure occurs, different failures have to be treated specially. One problem with 'Delauney' triangulaton is to decide whether a new point is inside the circumscribing sphere of an existing tetrahedron or not.[7] This problem arises whether the mesh is fine or coarse. To reduce failures, high-precision arithmetic and 64 bit real representation are necessary. With the present algorithm a new point in the tri-tree structure is generated by bisection of line segments between two already existing points. If this point already exists in the tri-tree structure, it must have been generated from bisection of the line between the same two end points. It is therefore very probable that the new point and the existing point will have the same binary represenation. However, a tolerance is introduced in the decision as to whether two points are identical or not. If this tolerance is set to less than half the smallest expected distance between points, no failure will occur. With the present method, single-precision arithmetic and 32 bit real representation have been used. No algorithmic failure due to inaccurate representation of real numbers has so for been detected.

The tri-tree structure reveals properties which seem ideal for using multigrid methods in solving the differential equations. Such methods will be investigated in the near future.

ACKNOWLEDGEMENTS

The author is grateful to Olav Dahl for discussions of numerical methods and to Knut Liestøl for discussions of the manuscript. The discussions of sorting and searching algorithms with Ole-Johan Dahl are highly appreciated. Otto Milvang and Jens Thomassen have been helpful concerning the computer facilities. The author is also grateful to Harald Svendsen for discussions and patient introduction to oceanography. The author is indebted to Lars Walløe who suggested the project, raised funds to make the project possible and for encouraging discussions throughout the work. The project has been supported by the Norwegian Research Council for Fisheries.

REFERENCES

1. J. Peraire, M. Vadati, K. Morgan and O. C. Zienkiewicz, 'Adaptive remeshing for compressible flow computations', *J. Comput. Phys.*, **71**, 449–466 (1987).
2. R. Lohner, 'Generation of three-dimensional unstructured grids by the advancing front metod', *AIAA Paper 88-1515, AIAA 26th Aerospace Science Meeting*, Reno, NV, January 1988.
3. R. Lohner, K. Morgan and O. C. Zienkiewicz, 'An adaptive finite element procedure for compressible high speed flows', *Comput. Methods Appl. Mech. Eng.*, **51**, 441–464 (1985).
4. H. K. Ruud and S. Ø. Wille, 'An advancing front algorithm for three dimensional mesh generation', *Proc. NUMETA 90, Numerical Methods in Engineering: Theory and Applications*, Elsevier Applied Science, Swansea, 1990, Vol. 2, pp. 1141–1148.
5. N. P. Weatherill, 'A metod for generating irregular computational grids in multiply connected planar domains', *Int. j. numer. methods fluids.*, **8**, 181–197 (1988).
6. A. Bowyer, 'Computing Diriclet tessellations', *Comput. J.*, **21**, 162–166 (1981).
7. T. J. Baker, 'Three dimensional mesh generation by triangulation of arbitrary point sets', *AIAA Paper 87-1124-CP, AIAA 8th Computational Fluid Dynamics Conf.*, Hawaii, June 1987.
8. W. J. Schroeder and M. S. Shepard, 'A combined Octree/Delauney method for fully automatic 3-D mesh generation', *Int. j. numer. methods eng.*, **29**, 37–55 (1990).
9. T. J. Baker, 'Unstructured mesh generation by a generalized Delauney algorithm', presented at the *AGARD Conference*, Loen, Norway, 1989.
10. M.-C. Rivara, 'A grid generator based on 4-triangles conforming mesh-refinement algorithms', *Int. j. numer. methods eng.*, **24**, 1343–1354 (1987).