Computer Science. Vol. 3. 2001 519.61/.64

Tomasz Jurczyk[*], *Barbara Głut*[*]

GENERATION OF TRIANGULAR MESHES FOR COMPLEX DOMAINS ON THE PLANE[**]

1. Introduction

Many physical phenomena can be modeled by partial differential equations (PDEs). Numerical methods for solving PDEs (FEM, FVM, BEM) approximate the solution of a linear or nonlinear PDE by replacing the continuous system with a finite number of algebraic equations. The result is a finite number of points in space at which variables such as velocity or electric field are calculated.

The collection of nodes and elements is called a finite element mesh. Structured meshes have a regular connectivity, while in an unstructured mesh each point have a different number of neighbours. A mesh must conform to the object or domain being modeled, and should meet constraints on both the size (possibly varying throughout the mesh) and shape of its elements.

A mesh generator should be fully automatic and should simplify input data as much as possible. It should offer rapid gradation from small to large sizes of elements. The generated mesh must be always valid and of reasonably good quality. All these requirements were taken into account during selection of algorithms used in successive stages of the presented mesh generation scheme.

2. Specification of a physical domain

The domain being triangulated is defined as a set of sub-domains (either filled or empty) (Fig. 1 demonstrates an example of a complex domain).

Boundaries of sub-areas and holes are described by a collection of edges. Exterior boundaries separate meshed and unmeshed portions of the space, while interior boundaries

* Institute of Computer Science, University of Mining and Metallurgy, Cracow

** This work was supported by KBN Grant 8 T11F 012 16

enforce the constraint that elements may not pierce them. Each edge can be specified as a straight line segment, arc, or fragment of a B-spline. The direction of edges determines which side of the edge is adjacent to the filled area. Edges adjacent to two filled areas and free edges (e.g. the *PR* edge in Fig. 1) are bidirectional.

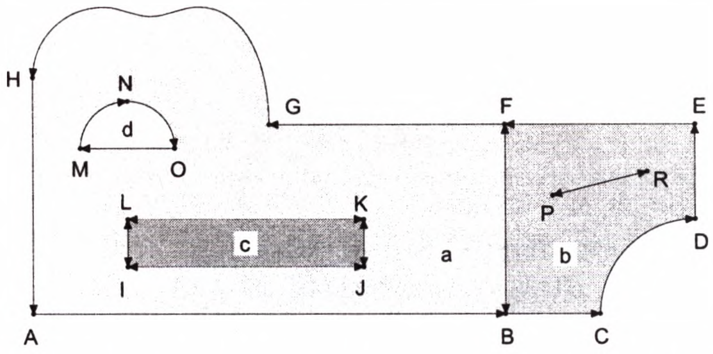

Fig. 1. Valid complex domain: a), b), c) filled subdomains; d) empty subdomains; ABFGHA, BCDEFB, IJKLI, MNOM – domain boundaries, PR – enforced inner edge

A valid specification of a domain must conform to few constraints:

- edges can't cross themselves (a corresponding graph must be planar);
- each edge must be incident to two different vertices (loops aren't allowed);
- there mustn't be any two edges incident to the same two vertices (any two nodes can be connected by at most one edge).

3. Generation of a requested number of boundary nodes

Before triangulation of the domain can take place, an initial set of nodes must be provided by discretizing boundaries of the domain. The number and location of boundary nodes should be specified with respect to a required density of a mesh in the vicinity of particular boundary edges. Additionally, a piecewise linear approximation of boundaries should be as close to the shape of original curves as possible.

The described algorithm (Fig. 2) of the discretization of the boundary leaves nodes already existing in the domain without changes. An additional number of nodes (specified by the user) is distributed uniformly along boundaries.

First, the length of all boundary edges and the total length of boundaries is evaluated. On the basis of the known number of additional nodes, which should be distributed among all edges (subtracting nodes already existing in the domain), an expected average length of a boundary edge is evaluated. This average value allows the estimation of a number of segments for each edge. After a repeated inspection of boundary edges, all vacant nodes are assigned and edges can be split. In order to reduce the influence of the sequence of inspected edges on the distribution of nodes, an estimated number of nodes for each edge is gradually increased (with a coefficient d_Δ).

```
evaluate the entire length of boundaries $l_t = \sum l'_i = \sum w_i l_i$,
    where: $l_i$ – length of a successive boundary edge,
           $w_i$ – weight of an edge (a geometric average of node weights),
evaluate the average expected length of an edge $l_{ave} := \frac{l_t}{n_e + n_\Delta}$,
    where: $n_e$ – number of initial boundary edges,
           $n_\Delta$ – number of additional boundary nodes,
evaluate the number of additional nodes for each boundary edge:
    $n_i := 0$, $d_\Delta := 0$,
    while $n_\Delta > 0$:
        for each boundary edge:
            $n'_i := \left\lfloor \frac{l'_i}{l_{ave}} + d\Delta \right\rfloor$
            if $n'_i > 0$:
                $\delta_n := min(n'_i - n_i, n_\Delta)$,
                $n_i := n_i + \delta_n$,
                $n_\Delta := n_\Delta - \delta_n$,
        $d_\Delta := d_\Delta + 0.2$,
split edges accordingly to the determined number of new nodes $n_i$
for each boundary edge,
```

Fig. 2. Algorithm of determining the location of boundary nodes

A positive value is assigned to each node in the domain. These values (called weights) can be used to diversify sizes of boundary edges (and consequently sizes of adjacent elements). Higher value of the weight at some node increases the density of the mesh in its neighbourhood (Fig. 3). The ratio of lengths of edges adjacent to two nodes is inversely proportional to the ratio of weights of these nodes.

The influence of weights of boundary nodes is ensured by scaling the length of each edge by a factor equal to the (geometric) average weight of vertices at ends of this edge. As a result, an edge with the higher average weight is split into adequately more (shorter) segments.

If weights at both ends of an edge are different, additional nodes are distributed along this edge with the adequately varying density (Fig. 4). The location of nodes is determined accordingly to the following rules (which allow to unambiguously evaluate distances between successive nodes):

- the ratio of lengths of the last sub-edge to the first one is equal to the ratio of the first weight to the last one $\frac{l_n}{l_0} = \frac{w_0}{w_1}$,
- the ratio of lengths of successive sub-edges l_i is constans $\frac{l_{i+1}}{l_i} = \sqrt[n]{\frac{w_0}{w_1}}$,
- the sum of lengths of all sub-edges is equal to the length of the whole edge.

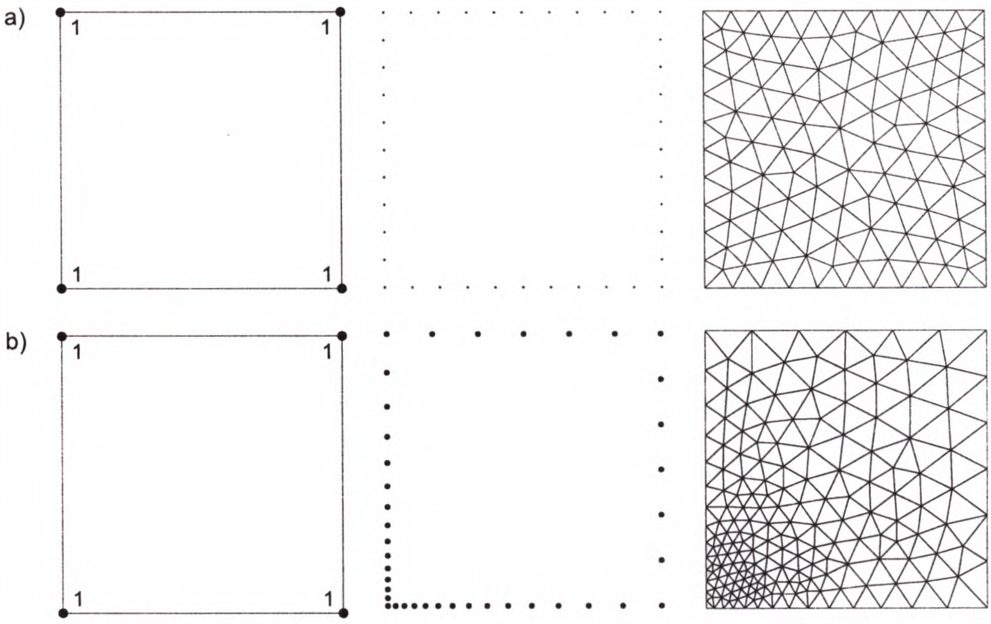

Fig. 3. The influence of weights of boundary nodes on the density of boundary edges and the resulting mesh: a) the uniform mesh; b) variable mesh

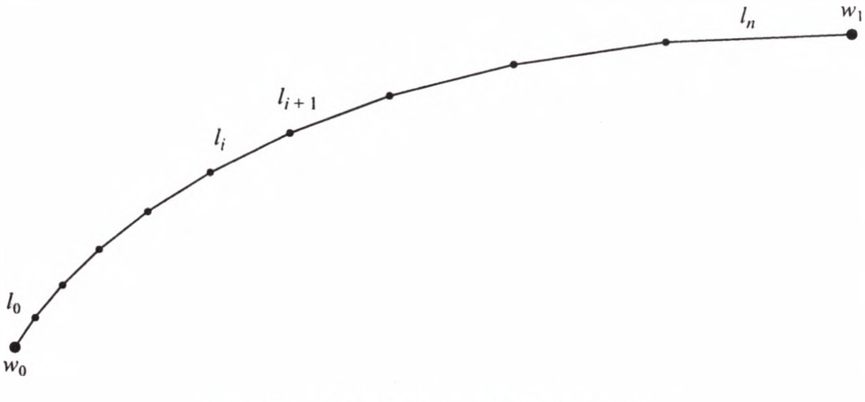

Fig. 4. Varying distribution of boundary nodes

4. Triangulation of a set of boundary nodes

The process of the generation of triangular mesh is based upon the well-known Delaunay triangulation (described e.g. in [8, 13, 16, 17]). This construction has some advantageous properties — in particular it is guaranteed, that the Delaunay triangulation of a vertex set maximizes the minimum angle among all possible triangulations of that set.

4.1. Delaunay triangulation of a set of vertices

The algorithm (Fig. 5) describes the process of triangulation of a given set of boundary nodes. The scheme is similar to the Bowyer-Watson algorithm [3, 18]. The construction starts with some simple triangulation (one or two triangles) which encloses the whole set of boundary nodes. This initial mesh is obviously Delaunay's one. Then, vertices are incrementally inserted into the mesh one by one, while maintaining the Delaunay's property.

```
determine the bounding box of all boundary nodes,
insert vertices of this rectangle into the mesh,
initialize the mesh with two triangles based on these vertices,
for each boundary node:
    insert this node into the mesh,
    retriangulate locally,
```

Fig. 5. Algorithm of Delaunay triangulation

The sequence of the insertion of boundary points is randomized in order to improve the efficiency of the triangulation. The local retriangulation of the mesh after an insertion of each node is carried out with respect to one of the properties connected with the Delaunay triangulation. The choice of the employed criterion has no influence on the shape of created mesh, because the resulting mesh is always Delaunay's one and, save for degenerated case, such triangulation is unique. However, the efficiency and robustness of the process of triangulating may vary.

4.1.1. Criterion of empty circumcircle

This criterion says, that any node of a triangle mustn't be contained within the circumcircle of any other triangle within the mesh. The only exception occurs, when there are more than three vertices which lie on one circle. The algorithm (Fig. 6) takes advantage directly of this criterion.

```
find a triangle containing the new node,
mark this triangle and add it to the list of invalid triangles,
loop for successive triangles from the created list:
    let T be the next triangle,
    if any triangle adjacent to T contains new node in its circumcircle
    and it is not marked:
        mark this triangle and add it at the end of the list,
determine all boundary edges of the cavity containing marked triangles,
delete all marked triangles,
add new triangles by joining edges of cavity with the new node,
```

Fig. 6. Algorithm of retriangulation with the criterion of an empty circumsphere

When a new vertex is inserted, each triangle whose circumcircle encloses the new vertex is no longer Delaunay's one, and should thus be deleted (Fig. 7). First, a triangle containing the new node must be found. Then, a set of triangles containing the new node in their circumcircles is determined by the broad-first search from the initial triangle. The set of deleted triangles forms a star-shaped cavity, which is left vacant. Each vertex of this cavity is then being connected to the new vertex with a new edge.

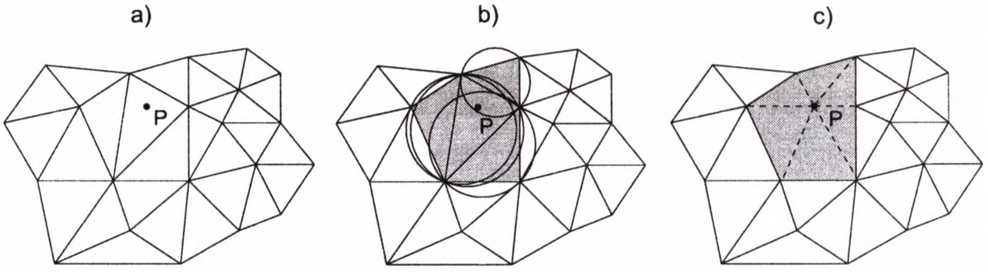

Fig. 7. Utilization of the property of an empty circumcircle: a) new node inserted into the mesh; b) triangles containing the new node in their circumcircles; c) the empty cavity and new triangles

4.1.2. Criterion of inner angles

The criterion of inner angles follows from the already mentioned property of maximizing of minimal angles, and it can be expressed in a few equivalent forms. The one used in the presented algorithm says, that for each pair of neighbouring triangles the sum of inner angles adjacent to the common edge mustn't be smaller than the sum of the remaining two inner angles (Fig. 8).

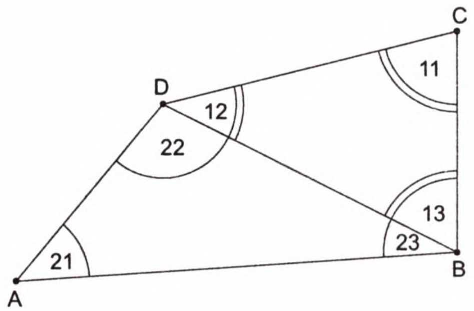

Fig. 8. Criterion of inner angles $\alpha_{12} + \alpha_{13} + \alpha_{22} + \alpha_{23} \geq \alpha_{11} + \alpha_{21}$

In the algorithm (Fig. 9) after each new node is inserted, the containing triangle is found and split into three new triangles (if the new vertex doesn't fall upon an edge of the triangle). Next, a recursive procedure tests whether any of the newly created triangles and one of theirs neighbours don't conform to the criterion of inner angles. Each affirmative test triggers an edge flip. The process of inserting of one node is shown in Figure 10.

```
find a triangle containing the new node,
remove this triangle,
if the node is inside this triangle:
    create new triangles by joining vertices of the removed
    triangle with the new node,
else if the node falls upon an edge of the triangle:
    remove an adjacent triangle (if it exists),
    create new triangles by joining vertices of this cavity with the new node,
    (if an adjacent triangle doesn't exist – create two new triangles),
add new triangles to the veryfying list,
for all triangles in the veryfying list:
    if a triangle and its neighbour don't conform to the
    criterion of inner angles:
        flip a common edge,
        add both altered triangles to the veryfying list,
    else remove this triangle from the veryfying list,
```

Fig. 9. Algorithm of retriangulation with the criterion of inner angles

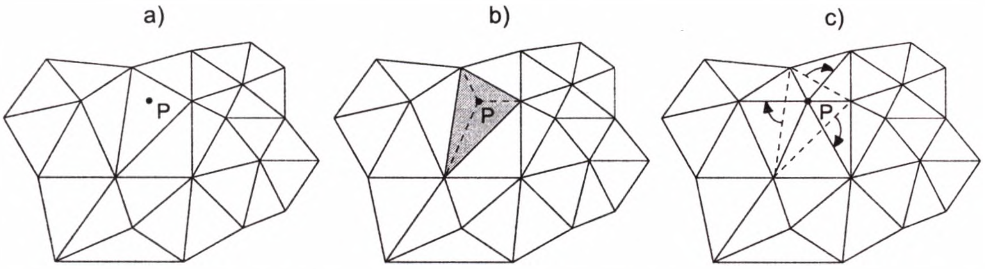

Fig. 10. Utilization of the criterion of inner angles: a) new node inserted into the mesh; b) division of the triangle into three smaller triangles; c) edges flipping

4.2. Localizing a containing triangle

In both retriangulating algorithms presented above, the first step requires to locate the triangle which contains the newly added node. To be precise, in the scheme based on the criterion of the empty circumcircle, localizing any triangle containing the new node in its circumcircle would be sufficient. However, choosing the triangle which contains the new node as an initial triangle has the advantage of avoiding some problems with the round-off error of evaluating whether a point lies within a circumcircle.

The algorithm (Fig. 11) of localizing the triangle containing the new node (which can be generalized to other polygons) is based upon the scheme of "mesh walking" in the direction of the new node [17] (Fig. 12a). This method requires the mesh to be coherent and to form a convex polygon without holes. Both these requirements are fulfilled at the first stage of the triangulation, before removing of obsolete triangles takes place. Afterwards, this method is no longer used, because inner nodes are inserted into a chosen triangle which needn't to be sought.

```
select an initial triangle:
    set the recently created triangle as an initial triangle,
    set the root of QuadTree as a current tree-node,
    *loop:*
        *if* the current tree-node contains a valid preferred triangle
        *and* this triangle is nearer to the given point than the current initial triangle:
            set this preferred triangle as the initial triangle,
        *if* the current tree-node has sub-nodes:
            set the sub-node in the direction of the given point as a current tree-node,
        *else* leave the loop,
localize the containing triangle:
    set the initial triangle as the current triangle,
        *while* the current triangle doesn't contain the given point:
        select an edge of the current triangle in the direction of this point,
        set the triangle adjacent by this edge as the current triangle,
```

Fig. 11. Algorithm of localizing of the triangle containing the given point

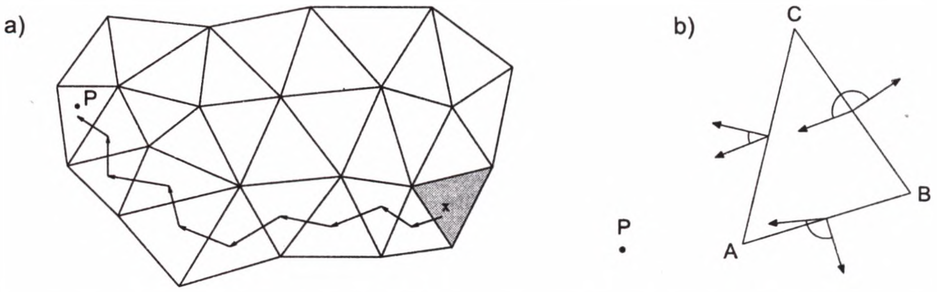

Fig. 12. Localizing of the triangle containing the point *P*: a) oriented scanning of elements in the mesh; b) selecting an edge in the direction of the given point

As the first checked triangle (called the initial triangle) the recently added element is selected. Such choice is advantageous, if successively inserted nodes are located near one to the another. Otherwise, an alternative method of choosing a good initial triangle must be utilized in order to minimize the number of triangles being checked.

For each checked element, a position of the node relatively to this element is determined. If the new node lies within the element (or at its edges), the procedure stops. Otherwise, one of the adjacent triangles in the direction of the node is selected as a candidate.

The method of choosing one of the neighbours is based upon values of angles between the normal vector and the directional vector, in the middle of edges (Fig. 12b). After evaluating scalar products of all pairs of vectors, the edge (and the adjacent triangle) with the largest value is chosen.

Since the problem of finding the containing triangle is crucial for the efficiency of the entire algorithm, the structure of geometrical QuadTree is utilized in order to ensure fast selection of a good initial element for a further procedure of scanning of the mesh. The QuadTree structure doesn't store all elements in the mesh. Each node of the tree, covering a rectangular area of the mesh, contains only one reference to the triangle (called a preferred triangle) which is located nearest to the middle of the node (Fig. 13a). During the evaluation of the distance from a triangle to the midpoint of a node, this triangle is represented by its centroid.

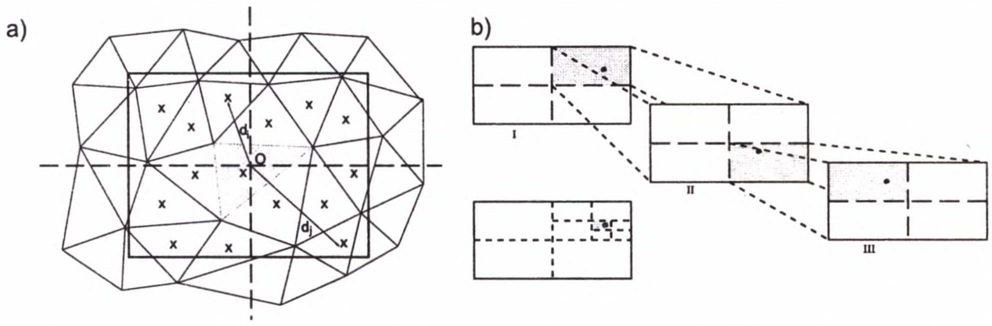

Fig. 13. Structure of the QuadTree: a) choosing the preferred triangle for a node of the tree; b) scanning the tree along the path determined by the given point

In order to preserve the coherence of the QuadTree, this structure must be updated each time any mesh element is inserted or removed from the mesh. A newly created triangle is sift through successive levels of the tree, according to coordinates of its centroid (Fig. 13b). For each visited node of the tree a check is performed, whether this triangle is nearer to the midpoint of this node than its preferred triangle. An affirmative test causes the replacement of the reference to preferred triangle. Additionally, in every visited node a counter of assigned triangles is increased. If the value of the counter becomes greater than a certain threshold, the node is split into four equal sub-nodes.

During the removal of any mesh triangle, the QuadTree must be updated in order to remove any references to this triangle. The triangle is sift through the tree and if the preferred triangle in any node is equal to the triangle being removed, such reference is invalidated. Because the QuadTree is used during the process of constructing the mesh and its size is gradually growing, removing a triangle (possibly connected with decreasing of some counters below the threshold) doesn't cause removing of any node of the tree.

During selecting of the optimal initial triangle, the QuadTree is scanned along the path determined by coordinations of the given point. For each visited tree node, the distance from the point to the centroid of the preferred triangle in this node is evaluated, and the current best initial triangle is updated in order to minimize the distance from initial triangle to the given point.

4.3. Recovery of the boundary

Delaunay triangulations are oblivious to the boundaries of the domain, and these boundaries may or may not appear in a triangulation (Fig. 14). In order to ascertain the presence of all boundary edges, one can enforce theirs construction during the triangulation or recover them afterwards. The presented scheme utilizes the second approach.

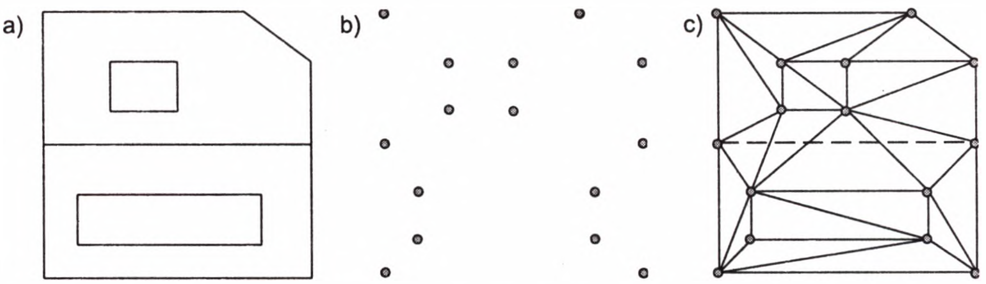

Fig. 14. Example of a missing boundary edge (marked with a broken line): a) shape of boundaries; b) set of boundary vertices; c) triangulation of boundary nodes

After all boundary points are inserted into the mesh, specified boundary edges are inspected. If any of them is missing, the process of the recovery is performed. Simultaneously, all boundary edges (both already existing and just recovered) are marked as unremovable and triangles adjacent to them are described in order to identify the sub-area of the physical domain, to which they belong.

Two methods of recovery of missing boundary edges are presented. One of them requires the insertion of additional nodes, while the other is based upon edges swapping.

4.3.1. Insertion of points

For each missing edge, the new node is inserted into its middle [19]. After a successful retriangulation, the presence of both sub-edges is ascertained. If any of them is still missing, the procedure is performed recursively (Fig. 15). During this recovery, both created edges as well as triangles adjacent to them are accordingly described.

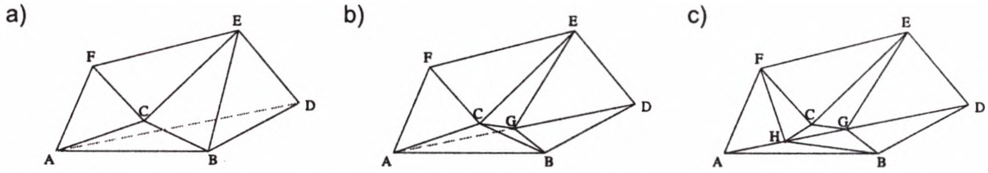

Fig. 15. Recovery of the boundary by the insertion of points: a) missing *AD* edge (marked with a broken line); b) inserting the *G* vertex in the middle of the *AD* segment; c) inserting the *H* vertex produces the required edge

4.3.2. Edges swapping

The algorithm (Fig. 16) of the recovery of the missing edge is based upon an iterative process of edges swapping in order to create the required edge [12]. It should be noticed, that the quadrilateral formed by two adjacent triangles can be concave, which prohibits the flip of the common edge (e.g. in Fig. 17a swapping of the *BC* diagonal in the *ABEC* quadrilateral is impossible). However, it is proved that an adequate order of edges swapping, resulting in the recovery of the desired edge, can always be found in 2D. While this method doesn't require the introduction of any additional points, the resulting mesh is usually of worse quality.

```
create a list of all edges, which cross the missing edge $E_0$:
    for each triangle $T$ incident to the point $P_1$ (a node of $E_0$):
        if the edge $E_1$ of the triangle $T$ opposite to $P_1$ crosses $E_0$:
            add the edge $E_1$ to the list,
            set this triangle as an initial triangle $T_1$,
    loop:
        let $T_2$ be a triangle adjacent to the triangle $T_1$ by the edge $E_1$,
        if the vertex of the triangle $T_2$ opposite to $E_1$ is incident to $E_0$:
            leave the loop,
        choose an edge $E_2$ of the triangle $T_2$ different to $E_1$, crossing $E_0$,
        add $E_2$ to the list,
        let $T_1 := T_2$, $E_1 := E_2$,
while the list is not empty:
    take the first edge from the list
    if swapping of this edge for neighbouring triangles is possible:
        remove this edge from the list,
        swap this edge,
        if the new edge crosses the missing edge:
            add the new edge at the end of the list,
        else move the edge to the end of the list,
```

Fig. 16. Algorithm of recovery of a missing edge by edges swapping

While missing edges are recovered, the resulting triangulation may be no more Delaunay's one and it is called "constrained Delaunay triangulation" [5, 12]. As a consequence of this redefinition, some alterations are introduced into triangulation schemes described above. In particular, all boundary edges are treated henceforth as unremovable. In the algorithm of the retriangulation by the criterion of an empty circle, adjoining triangles, whose common edge is marked as a boundary edge, are no longer considered neighbours. In the algorithm utilizing the criterion of inner angles any boundary edge can't be swapped. All these alterations guarantee that once recovered boundary edges won't be removed during the further processing of the mesh.

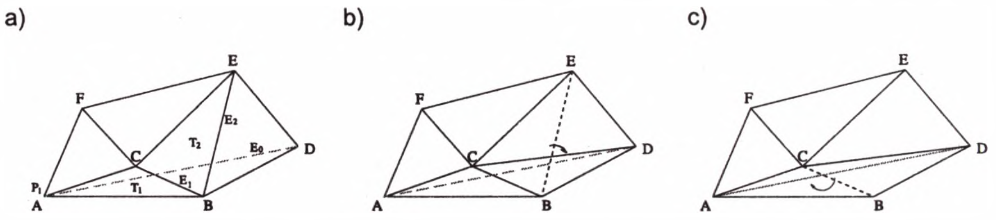

Fig. 17. Recovery of the boundary by edge swapping: a) missing *AD* edge (marked with broken line); b) first swap ($BE \to CD$); c) second swap ($BC \to AD$) resulting in the recovery of the missing edge

4.4. Deletion of obsolete elements

After enforcing the presence of all boundary edges, the deletion of obsolete elements can be performed. Up to this moment, the area of mesh was defined by two auxiliary triangles enclosing the whole domain. As a result, the mesh contains additional triangles (surrounding the domain or forming holes) which should be removed. The algorithm (Fig. 18) is coupled with identifying all valid triangles as being a part of one of the specified sub-areas in the domain.

```
mark and add to the verifying list all valid triangles, identified
during the validation of boundary edges (Section 4.3),
for each triangle from the verifying list:
    for each neighbour of this triangle:
        if the edge common to this pair of triangles is not a boundary one
        and the adjacent triangle isn't marked yet:
            mark this triangle,
            copy the area identifier,
            add this triangle into the verifying list,
remove all unmarked triangles,
```

Fig. 18. Algorithm of deletion of obsolete elements

5. Refining the mesh by an introduction of inner nodes

The quality of the mesh obtained by the triangulation of boundary points depends upon their arrangement. If thus created mesh doesn't conform to specified requirements concerning the size or the quality of its elements, it must be improved by the insertion of additional points. As an optimal shape of a triangle, the equilateral one is assumed. The information about node spacing within the domain is approximated on the basis of the density of boundary nodes.

5.1. Insertion of inner nodes

First, the quality of all mesh elements is evaluated according to the chosen criterion. Nodes are then inserted incrementally into the existing mesh in the vicinity of bad triangles, redefining the triangles locally as each new node is inserted, to maintain the constrained Delaunay triangulation (Fig. 19). Criterions of assessing the quality of triangles, and methods that are chosen for defining where to locate interior nodes are described in the following sections.

```
evaluate quality coefficients (according to the chosen criterion) for all triangles,
reorganize all triangles into the structure of the heap
(with the worst element at its top),
loop
    fetch the worst triangle from the top of the heap,
    if the quality of this triangle is higher then the required threshold:
        leave the loop,
    else:
        insert a new node inside this triangle,
        retriangulate locally,
        if the number of inner nodes is higher then the maximum value:
            leave the loop,
```

Fig. 19. Algorithm of insertion of inner nodes

Before the process of refining the mesh takes place, the set of triangles creating the mesh is reorganized into the structure of the heap [2], which is preserved during the subsequent retriangulations. Utilizing this structure allows to find the worst triangle in the single step (it is always located at the top of the heap), while it introduces a logarithmic complexity of adding and deleting of triangles.

In order to avoid the creation of triangles with too small area (which is particularly characteristic for the category of quality criterion based upon the shape of triangles), a restriction on their size is imposed. For each triangle adjoining the boundary the minimum area is determined (equal to the area of an equilateral triangle based upon a boundary edge). The smallest of all these minimum areas is used as a minimum area for all inner triangles. During the refinement, the quality coefficient of each triangle with area less then its minimum value, is set high enough in order to prohibit the division of this triangle.

5.2. Criterions of the quality of triangles

Each step of the procedure of the refinement selects the currently "worst" triangle and it attempts to improve the mesh by inserting a new node in the vicinity of this triangle. The definition of the "worst" triangle implies introducing of an ordering relation. Criterions of assessing the quality of triangles, which exert a significant influence on the shape of the created mesh, can be divided into three categories.

5.2.1. Weighted criterion

Before the insertion of inner nodes takes place, a weight is ascribed to each boundary node. The value of the weight at each node is calculated as an average length of boundary edges incident to this node. Next, the quality of each triangle is evaluated according to the formula [6]

$$Q(\Delta)=\frac{G(\Delta)^2}{S(\Delta)} \tag{1}$$

where:

$S(\Delta)$ – area of the triangle,

$G(\Delta)$ – geometric average of weights of vertices of this triangle.

Thus determined quality coefficient is always positive and its optimal value is approximately equal to 2. Increasing this value produces meshes with the higher density.

5.2.2. Shape criterions

One of the utilized criterions is the coefficient α evaluated according to the formula 2 [15]. The range of this coefficient for valid triangles is (0, 1]. α is equal to 1 for equilateral triangles and converges at 0 while triangles degenerate into a line segment. There are included some other shape coefficients (3), whose values grow as a triangle becomes "better shaped". Factors used in all these formulas allow to obtain 1 as the value of each coefficient for the equilateral triangle:

$$Q(\Delta)=\alpha(ABC)=2\sqrt{3}\frac{\|CA\times CB\|}{\|CA\|^2+\|AB\|^2+\|BC\|^2} \tag{2}$$

$$Q(\Delta)=\frac{r}{R},\ \frac{r}{p},\ \frac{\pi r^2}{S},\ \frac{(p-a)(p-b)(p-c)}{p^3},\ \frac{r}{d} \tag{3}$$

where:

r – radius of the inscribed circle,

R – radius of the circumscribed circle,

S – area of the triangle,

p – half of the circumference of the triangle,

a, b, c – lengths of edges of the triangle,

d – longest edge of the triangle.

5.2.3. Size criterion

The quality of triangles is evaluated as a ratio of the requested size of elements throughout the mesh to their current areas. The background spacing grid [15] is used in order to approximate expected sizes of elements within the mesh on the basis of the distribution of boundary nodes. Determined sizes of elements along the boundary are equal to the area of equilateral triangles based upon boundary edges (Fig. 20). The spacing function within the domain is interpolated in order to obtain gradual transitions between triangles with various areas. This spacing function is represented as a regular square grid enclosing the whole triangulated domain. The values of the function within each square are interpolated on the grounds of values at grid nodes (forming vertices of this square). The density of the space grid depends upon the ratio of the shortest boundary edge to the size of the bounding box of the domain.

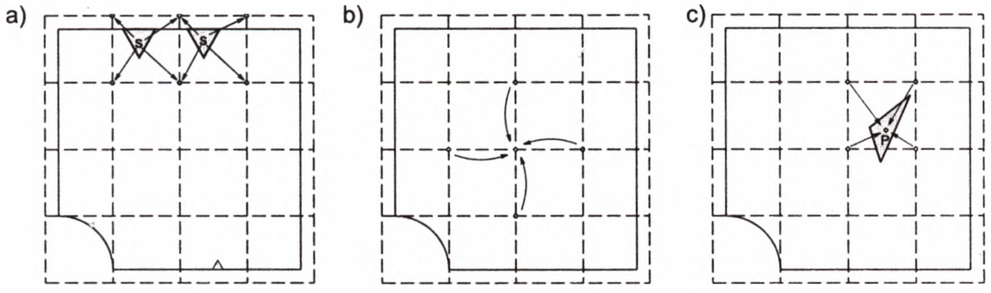

Fig. 20. Background spacing grid: a) determining values of the spacing function along boundaries; b) interpolating the spacing function within the domain; c) obtaining the expected size for a triangle

5.2.4. Quality threshold

The significant parameter of the algorithm (Fig. 19) of the insertion of the insertion of inner nodes is the threshold value of the quality coefficient, up to which triangles are still being improved (as long as the maximum number of inserted nodes isn't reached – if such limitation exists). Increasing the threshold value causes increasing the number of inner nodes and usually a better quality of the resulting mesh. For criterions based upon the shape of triangles, the threshold value should be less than 1.

5.3. Determining the location of inner nodes

Three strategies of the location of a new node for the triangle being refined are presented. Analogously to the choice of the quality criterion, the method of introducing new nodes has a large impact upon the structure of the resulting mesh.

5.3.1. Midpoint of the longest edge

The new point is inserted in the middle of the longest edge of the triangle. The weight of this point (if the weighted criterion is utilized) is calculated as an average value of weights of both nodes of this edge.

5.3.2. Weighted centroid of the triangle

The new point is inserted according to the formula 4 [19]

$$\vec{x} = \frac{W_1\vec{x}_1 + W_2\vec{x}_2 + W_3\vec{x}_3}{W_1 + W_2 + W_3} \tag{4}$$

The weight of this point is calculated as

$$W = \alpha_w \frac{W_1 + W_2 + W_3}{3} \tag{5}$$

Increasing the value of factor α_w lessens the density of the resulting mesh. The recommended range of this factor is [1, 1.3].

5.3.3. Circumcentre of the triangle

Coordinates of the new node are equal to the centre of the circle circumscribed on the triangle being improved [9]. While the circumcentre of a triangle may or may not belong to this triangle, a search for a containing triangle must be performed. If such triangle can't be found (the point lies outside the domain or there is a boundary edge between the initial triangle and the point) the operation is cancelled. In such case the quality coefficient of the triangle is set high enough to prohibit further processing of this triangle (until any modification of neighbouring triangles takes place).

The weight of the new point is evaluated according to the formula 5 already mentioned.

If the quality of triangles is assessed on the basis of criterions different from the weighted criterion, weights of all points are set to 1 (and have no impact upon the created mesh).

6. Improvement of the mesh

The quality of the constructed mesh can usually be improved by employing some optimization techniques. Methods presented below can be divided into two categories based upon smoothing and clean-up. Smoothing adjusts node locations while clean-up generally changes the element connectivity. In order to achieve best results both types of methods should be used alternatively. The process of improving the quality of the mesh is finished, if the quality is good enough or the efficiency of methods falls below some threshold.

6.1. Laplacian smoothing

This well-known simple method (Fig. 21) [7] relocates nodes of the mesh in order to locally even out lengths of inner edges. During every pass of the procedure each internal node in the mesh is placed at the average location of any node connected to it by an edge

$$\vec{x} = \frac{1}{n}\sum_{i=1}^{n} \vec{x}_i \tag{6}$$

where n – number of nodes connected to the given point by an edge.

```
for each node in the mesh:
if the node doesn't belong to the boundary:
    change the location of this node according to the formula 6,
    for each triangle incident to the node:
        if the triangle is inverted (its area is negative):
            restore the previous location of the node,
```

Fig. 21. Algorithm of Laplacian smoothing

Unfortunately, for some configurations of the mesh, the Laplacian formula can result in a corrupted mesh (Fig. 22). The presented algorithm includes an inspection of the correctness of the mesh after each relocation of the node. If the resulting mesh is invalid, the last modification is cancelled.

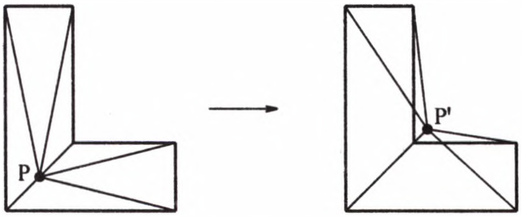

Fig. 22. Failure of Laplacian smoothing

6.2. Edges swapping

This method is based upon the already mentioned strategy of swapping the common edge in two adjacent triangles, if such modification is possible and if the criterion of inner angles is not fulfilled (i.e. if the sum of inner angles adjacent to the common edge is less than the sum of other inner angles). The algorithm (Fig. 23) iterates through all triangles in the mesh. For each pair of adjacent triangles a check can be made to determine whether swap of diagonal edge would improve quality of these triangles [13]. If the test is affirmative, the edge is swapped.

```
for each triangle in the mesh:
    for each adjacent triangle:
        if the common edge doesn't belong to the boundary
        and triangles form a convex quadrilateral
        and the sum of inner angles adjacent to the common edge
        is less than the sum of residual inner angles:
            swap the edge (Figure 24),
            update parameters of triangles,
```

Fig. 23. Algorithm of edges swapping

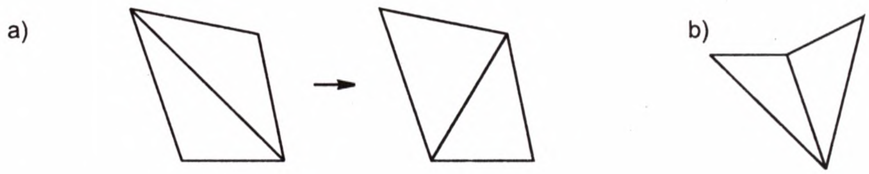

Fig. 24. Clean-up of the mesh by edges swapping: a) edge swapping; b) edge can't be flipped in this concave quadrialateral

6.3. Topological clean-up

For a triangular mesh there should be optimally 6 edges at any inner node. Whenever there is an inner node that does not have an ideal degree, the quality of elements surrounding it will also be less than optimal. While for meshes with varying density, nodes with degree different from 6 are indispensable, one should avoid creating nodes with too low or too high a degree. Topological clean-up is oriented solely to improve the connectivity of nodes, and the quality of elements in the mesh may actually decrease. Therefore, it should be coupled with a method directly improving the quality of elements, such as Laplacian smoothing.

6.3.1. Adjusting local spacing of nodes

First of the presented methods is based upon the insertion and the deletion of some inner nodes, in order to balance locally the density of the mesh. The scheme for processing sparse nodes (with a degree less or equal to 4) is described in the algorithm (Fig. 25).

```
for each node in the mesh:
    if the node doesn't belong to the boundary:
        if the degree of the node is equal to 4:
            remove the node together with four adjacent triangles,
            add two new triangles inside the cavity (Fig. 26),
        if the degree of the node is equal to 3:
            remove the node together with three adjacent triangles,
            add a new triangle inside the cavity,
```

Fig. 25. Algorithm of correction of sparse nodes

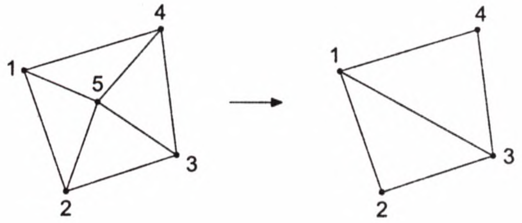

Fig. 26. Deletion of a sparse node

Dense nodes (with a degree greater or equal to 8) are improved by the insertion of additional one or two nodes in theirs vicinity (Fig. 27).

```
for each node in the mesh:
    if the node doesn't belong to the boundary:
        if the degree of the node is equal to 8 (Fig. 28):
            choose an incident edge with the greatest degree of the other node,
            move the node along this edge by $\frac{1}{3}$ its length,
            insert a new node in the middle of the opposite edge,
            perform the retriangulation (with the criterion of inner angles),
        if the degree of the node is greater or equal to 9 (Fig. 29):
            choose an incident edge with the greatest degree of the other node,
            move the node along this edge by $\frac{1}{3}$ its length,
            select edges $E_1$ and $E_2$
            (which divide all incident edges into three even groups),
            insert a new node in the middle of the edge $E_1$,
            perform the retriangulation,
            insert a new node in the middle of the edge $E_2$,
            perform the retriangulation,
```

Fig. 27. Algorithm of correction of dense nodes

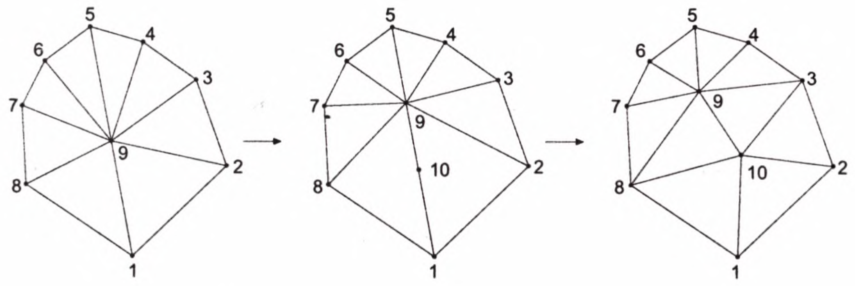

Fig. 28. Insertion of one new node

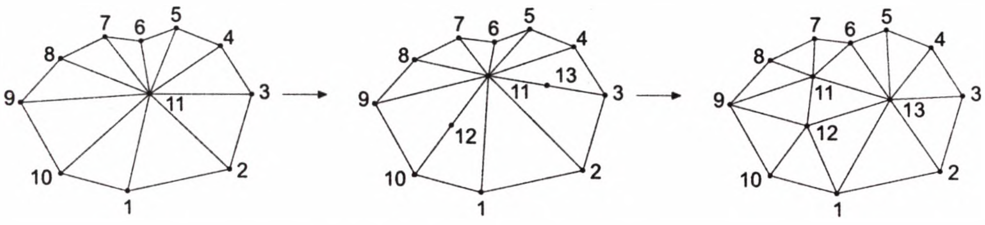

Fig. 29. Insertion of two new nodes

6.3.2. Edges swapping

The second of presented methods is based upon swapping edges in order to even out degrees of vertices in adjacent triangles (algorithm – Fig. 30) [4, 10]. While this method allows to reduce considerably the number of sparse and dense nodes, it leaves the number of all inner nodes unchanged.

```
for each edge:
    if for the edge 2–4 (Fig. 31)
    none of the vertices 1, 2, 3 and 4 belongs to the boundary
    and degrees of these vertices comply to $n_2 + n_4 - n_1 - n_3 > 2$:
        swap the edge 2–4 into 1–3, if it is possible,
        if the quadrilateral 1–2–3–4 is convex),
```

Fig. 30. Algorithm of balancing degrees of nodes by edges swapping

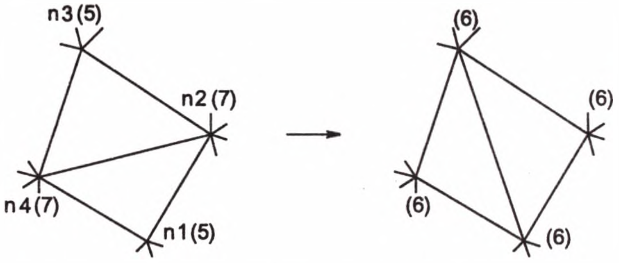

Fig. 31. Swapping an edge

7. Renumbering

The purpose of the algoritm (Fig. 32) of renumbering nodes in the mesh is to accelerate computations, for which the mesh is created. The main idea is to minimize the difference of numbers of vertices in elements in the mesh in order to decrease the bandwidth in the stiffness matrix for the finite element method (FEM). The scheme is based upon the broad-first scanning of the graph – nodes of the mesh are numbered from the first one, through its neighbours, and their neighbours, etc.

Since the choice of the first (starting) node has significant impact on the effectiveness of the entire scheme, three passes are performed (in alternated directions) with the last node being the first in each successive pass.

The algorithm (Fig. 32) describes forward renumbering. The backward scheme (iterating points from n to 1) is identical but for some slight changes reversing the process.

Another variation of the algorithm described above is renumbering with the correction for distances. This procedure starts with the renumbering algorithm (Fig. 32) and then all vertices are sorted according to their distance from the starting point (for each disjoint area separately). For some cases of meshes, especially with curvilinear edges, this additional operation allows to improve the quality of renumbering.

```
$k := 2$, ($k$ — expected number of the successive neighbour of the point $P[i]$),
for $i = 1 \ldots n$:
    if $k \le i$:
        $k := i + 1$, (important for incoherent areas)
        for each neighbour ($P[j]$) of the point $P[i]$:
            if $j > k$:
                swap points $P[j]$ and $P[k]$,
                $k := k + 1$,
            else if $j = k$:
                $k := k + 1$,
```

Fig. 32. Algorithm of renumbering of mesh nodes

8. Results

Figure 33 shows the process of triangulation for a complex domain. After generation of 100 boundary vertices (according to weights ascribed to nodes in the domain) the Delaunay triangulation is performed. Next, the boundary is recovered and obsolete triangles are removed. The resulting mesh is completed by the Delaunay refinement with the weighted criterion and introducing new nodes in the circumcentre of triangles being improved. The final mesh is then improved by three passes of described optimization procedures.

Criterions of assessing the quality of triangles and strategies of locating new nodes during the refinement procedure allow to obtain meshes with a various structure and quality.

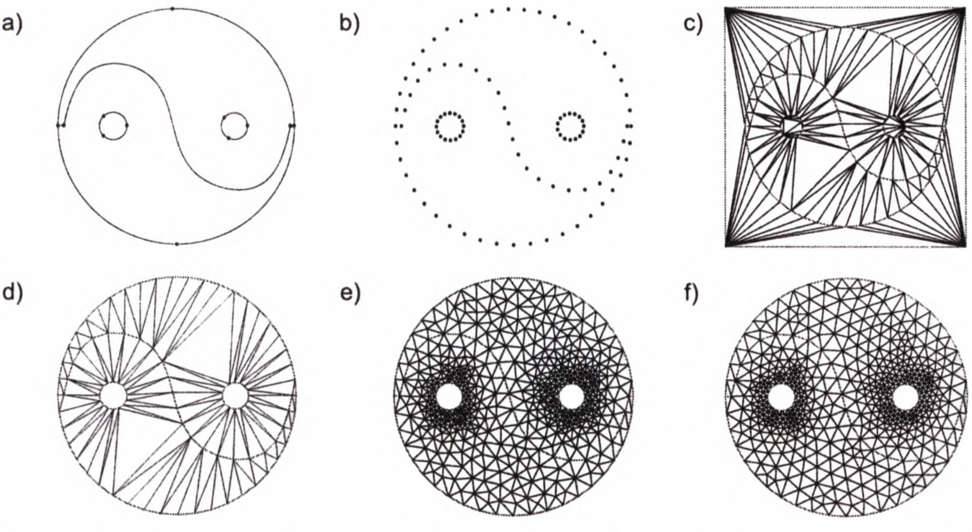

Fig. 33. Successive stages of the scheme of the mesh generation: a) specified domain; b) generated boundary nodes; c) triangulation of a set of boundary nodes; d) deletion of obsolete triangles; e) insertion of inner nodes; f) smoothing and clean-up

Figure 34 presents meshes created for various quality criterions (new nodes are introduced in circumcentres of triangles), while in Figure 35 results for different strategies of locating new inner nodes (and the weighted criterion) are demonstrated.

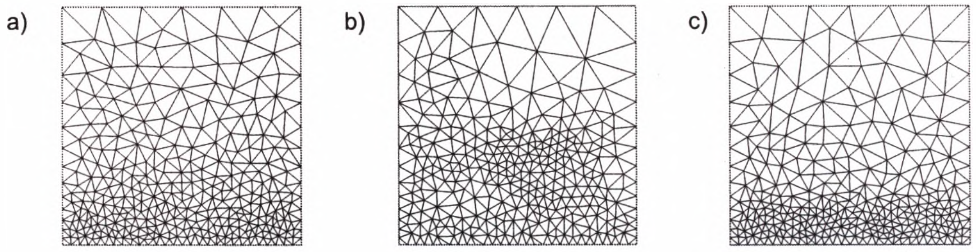

Fig. 34. Meshes generated with different quality criterions: a) weighted criterion; b) shape criterion; c) size criterion

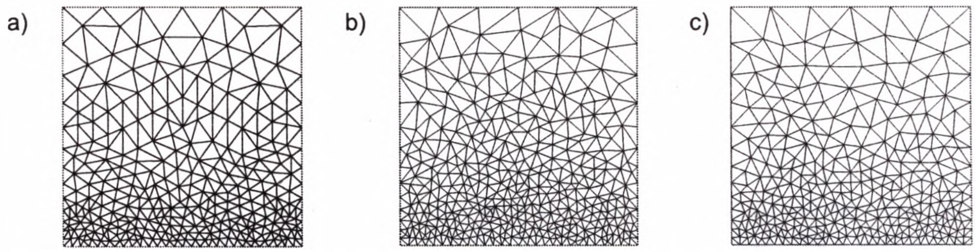

Fig. 35. Meshes generated with different strategies of locating new inner nodes: a) midpoint of the longest edge; b) centroid of the triangle; c) circumcentre of the triangle

The detailed analysis of the quality of meshes obtained for different variations of triangulation scheme as well as the study of the expected efficiency of particular algorithms is presented in [14].

9. Conclusions

In presented paper a complete scheme of the generation of triangular, unstructured meshes for 2D domains is described. The scheme is based upon the Delaunay triangulation. A serious stress was put on the ability to perform a fully automatic triangulation of complex domains while meeting requirements regarding the quality and size of elements throughout the mesh. Described algorithms were optimized with respect to running time and an empirical efficiency close to linear was achieved for the whole process.

The shape of the resulting mesh can be influenced by means such as weights of vertices in the domain, the choice of parameters and strategies employed during the refinement procedure, or the selection of improving methods. Elements in the mesh can be converted into curvilinear triangles with consideration for the complex shape of the boundary (in order to increase the matching of the mesh and the physical domain being modeled).

Future works will focus on extending the mesh generation scheme into a three-dimensional and anisotropic space, and on the generation of all-quadrilateral meshes.

References

[1] Baker T.J.: *Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation*. Engineering with Computers, 5, 1989, 161–175

[2] Banachowski L., Diks K., Rytter W.: *Algorytmy i struktury danych*. Warszawa, WNT 1996

[3] Bowyer A.: *Computing Dirichlet tesselllations*. The Computer Journal, 24, 1981, 162–166

[4] Cannan S.A., Muthukrishnan S.N., Phillips R.K.: *Topological refinement procedures for triangular finite element meshes*. Engineering with Computers, 12 (3&4), 1996, 243–255

[5] Chew L.P.: *Constrained Delaunay triangulations*. Proceedings of the 3rd Symposium on Computational Geometry, ACM Press, 1987, 215–222

[6] Coulomb J.L.: *Expérimentation de la triangulation de Delaunay*. Conf. Proc. Automated mesh generation and adaptation, Grenoble, IMAG 1987

[7] Field D.A.: *Laplacian smoothing and Delaunay triangulations*. Communications in Applied Numerical Methods, 4, 1988, 709–712

[8] Filipiak M.: *Mesh generation*. The University of Edinburgh, 1996 (technical report)

[9] Frey W.H.: *Selective refinement: A new strategy for automatic node placement in graded triangular meshes*. International Journal for Numerical Methods in Engineering, 24(11), November 1987, 2183–2200

[10] Frey W.H., Field D.A.: *Mesh relaxation: A new technique for improving triangulations*. International Journal for Numerical Methods in Engineering, 31, 1991, 1121–1133

[11] George P.L.: *Automatic mesh generation: application to finite element methods*. Chichester, Wiley 1991

[12] George P.L., Hecht F., Saltel E.: *Automatic mesh generator with specified boundary*. Computer Methods in Applied Mechanics and Engineering, 92(3), November 1991, 269–288

[13] Głut B.: *Generowanie i adaptacja siatek w modelowaniu procesów metodą elementów skończonych*. Kraków, Akademia Górniczo-Hutnicza, 1996(Ph.D. thesis)

[14] Głut B., Boryczko K., Jurczyk T., Alda W.: *Comparison of algorithm efficiency and mesh quality in Delaunay triangulation for complex 2D domains*. Accepted for 7th International Conference on Numerical Grid Generation in Computational Field Simulations, Whistler, BC, Canada, 25–28 September 2000

[15] Lo S.H., Lee C.K.: *Generation of gradation meshes by the background grid technique*. Computers & Structures, 50(1), 1994, 21–32

[16] Löhner R.: *Automatic unstructured grid generators*. Finite Elements in Analysis and Design, 25, 1997, 111–134

[17] Müller J.D.: *On Triangles and Flow*. The University of Michigan, 1996 (Ph.D. thesis)

[18] Watson D.F.: *Computing the n-Dimensional Delaunay tesselation with application to Voronoi polytopes*. The Computer Journal, 24, 1981, 167–172

[19] Weatherill N.P.: *Delaunay triangulation in computational fluid dynamics*. Computers and Mathematics Applications, 24(5/6), September 1992, 129–150