If after this stage there was a need to introduce a new module, changes in the interface or having trouble with the volume and flow of necessary information, there is an opportunity to return to the stage of vehicle development block in the user interface or database structure.

After completion of the work over the project PP working version comes, that took all the necessary revision and testing, and the project seems to continue to have support.

1. Евсеев В. В. Разработка новой модели жизненного цикла на ранней стадии проектирования программного обеспечения / В. В. Евсеев // Экономика, наука, производство : сборник научных трудов № 26. – М.: Издательство "Московский государственный открытый университет имени В. С. Черномырдина", 2013. – 192 с. – С. 61–62. 2. Евсеев В. В. Параметрическая модель декомпозиции структурированного языка SQL для решения задач расчета трудоемкости / В. В. Евсеев, В. О. Бортникова // 17 международный молодежный форум "Радиоэлектроника и молодежь в XXI веке" Сб. материалов форума. Том 2. – Харьков: ХНУРЭ, 2013 с. – С. 119–120. 3. Невлюдов И. Ш. Разработка графа параметрической зависимости для кис тпп на базе языков высокого уровня программирования / И. Ш. Невлюдов, В. В. Евсеев, С. С. Милютина, В. О. Бортникова // Вестник национального технического университета "ХПИ". – Харьков, 2012. – Вып. 66. – С. 67–73. 4. Невлюдов И. Ш. Модели жизненного цикла программного обеспечения при разработке корпоративных информационных систем технологичекой подготовки производства / И. Ш. Невлюдов, В. В. Евсеев, В. О. Бортникова // Вестник национального технического университета "ХПИ". – Харьков, 2011. – Вып. 2. – С. 94–101. 5. Невлюдов И. Ш. Анализ жизненного цикла разработки программного обеспечения для корпоративных информационных систем / И. Ш. Невлюдов, В. В. Евсеев, А. А. Андрусевич // Восточно-европейский журнал передовых технологий. – Харьков, 2010.– Вып. 6/8(48). – С. 25–27. 6. ISO/IEC 12207:2008. Информационные технологии. Процессы жизненного цикла программного обеспечения [Электронный ресурс]. – Режим доступа: \www/ URL: http://www.iso.org/iso/ru/catalogue_detail?csnumber=43447/ – 23.10.2013 г. – Загл. с экрана. 7. Каталог стандартов [Электронный ресурс]. – Режим доступа: \www/ URL: http://www.iso.org/iso/ru/home/store/catalogue_ics.htm/ – 15.09.2014 г. – Загл. с экрана.

UDC 004.021

K. Kolesnyk, N. Kretovych, M. Ziaei
Lviv Polytechnic National University,
West Saxon University of Applied Sciences of Zwickau

AUTOMATIC QUADRILATERAL MESH GENERATION FOR NON-CIRCULAR SHAFT PROFILES

АВТОМАТИЗОВАНЕ ГЕНЕРУВАННЯ ЧОТИРИКУТНОЇ СІТКИ ДЛЯ НЕКРУГЛИХ ПРОФІЛІВ ВАЛА

This paper is devoted to a technique for automatic generation of quadrilateral meshes for shafts and hubs with non-circular profiles.

Key words: quadrilateral mesh, mesh optimization, mesh smoothing.

Запропоновано методику для автоматичного створення чотирикутної сітки для валів і втулок з некруглими профілями.

Ключові слова: чотирикутна сітка, оптимізація сітки, згладження сітки.

1. Introduction

In order to conduct simulation modeling and research for the stress-strain state of the connection of a shaft and a hub, it is necessary to generate a high-quality quadrilateral mesh for these two connection components. There is a lot of algorithms for generating and smoothing the mesh: automated triangulation of surfaces via bubble packing [1], a non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm [2], an angle-based approach to two-dimensional mesh smoothing [3], quadrilateral meshing with directionality control through the packing of square cells [4], isotropic 2D quadrangle meshing with size and orientation control [5], Delaunay refinement mesh generation [6], etc. However, for effective automated mesh generation, automated triangulation of surfaces via bubble packing algorithm, a non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm, an angle-based approach to two-dimensional mesh smoothing can be used. A technique for automatic generation of quadrangular mesh, which is presented in this paper, can be applied to the shaft and the hub with non-circular profiles. Using this technique, a rectangular grid in the cross section of the connection of the shaft and the hub can be generated. First of all, it is necessary to form a rectangular elements in an area where there is a connection between shaft and hub. Then the algorithms for generating the grid of bubbles, Delaunay triangulation algorithm, Blossom-Quad algorithm, an angle-based approach to two-dimensional mesh smoothing are applied.

2. Mesh generation methods

Automated triangulation of surfaces and volumes via bubble packing. The algorithm of the automated triangulation of surfaces and volumes via bubble packing, described in [1], is used in order to obtain well-shaped triangular mesh elements in some polygon. The algorithm is based on so called bubble packing. At the beginning, the border of some domain (polygon) is filled with bubbles (or circles in 2D-space), which touch each other (they do not intersect and are not far away from each other). Then the center point of each bubble is used during the process of Delaunay triangulation: each point is inserted into the domain, creating new triangular elements. And, at the end of the algorithm, a high-quality triangular mesh is obtained.

Blossom-Quad algorithm. In [2], it is described how a triangular mesh can be converted to a quadrilateral one. At the beginning of the algorithm, a weighted graph is created according to a set of triangles of the triangular mesh. Every vertex of the graph is a triangle of the mesh. A graph is built using triangle adjacencies in the mesh. Every edge of the graph is an internal edge of the mesh that connects two neighboring triangles. After the creation of the graph, a perfect matching of edges is found in this graph. Edges that have been marked during the algorithm mean that two neighbor triangles can be joined to the quadrilateral. After all steps of this algorithm, a quadrilateral mesh is obtained.

An angle-based approach to two-dimensional mesh smoothing. The algorithm, described in [3], is used in order to make a triangular or quadrilateral mesh better, that is, to improve the quality of mesh elements (triangles or quadrilaterals).

The algorithm is easy for implementing and does not require a lot of computer calculations. The location of each internal node of the mesh is improved according to angles, which are near this node.

3. Quadrilateral mesh generation for non-circular shaft profiles

The problem we are interested in is the automated high-quality quadrilateral mesh generation for non-circular profiles of shafts or hubs. The input data is a profile of the shaft (or the hub) that must be meshed with quadrilaterals. The output data is a quadrilateral mesh for the profile.

Main requirements for the algorithm are: mesh finite elements must be quadrilaterals and have high quality; the time for building the mesh using computers must be small.

In this work, an algorithm for building a high-quality mesh for non-circular shaft profiles is proposed. It includes following steps of mesh generation: creating good-quality 2D mesh, applying the model extrusion and creating 3D mesh from 2D mesh created.

In order to create the mesh for the shaft profile, we:

- find elements and nodes in important region (in region of interest) of the model, calculate their coordinates;
- calculate the element size field in order to control mesh size over the domain;
- pack bubbles on domain border vertexes (on edges);

- pack bubbles inside the domain;
- optimize bubbles' population on the domain;
- find force-balanced configuration of bubbles;
- do basic Delaunay's triangulation of border nodes using the circumscribed rectangle;
- remove this circumscribed rectangle;
- apply Delaunay's triangulation algorithm for nodes inside the domain (these nodes are centers of bubbles, i.e., circles);
- convert triangular mesh to quadrilateral one;
- apply mesh smoothing using an angle-based approach;
- apply topological optimization.

During the first stage, we calculate coordinates of nodes of quadrilateral elements, which are near the contact domain and must have very high quality.

Bubble packing.

We need to obtain perfect triangular mesh in order to convert it to quadrilateral mesh during further stages.

Bubbles (or circles for 2D-dimensional space) are very useful for determining points (i.e., how many points and what are their coordinates), which must be put into triangulation for high-quality triangular mesh.

The algorithm of automated triangulation of surfaces via bubble packing is described in [1].

First of all, we put a bubble on each node of domain border lines. The center of a bubble has coordinates according to coordinates of an appropriate border node. The value of diameter of each bubble is obtained using the element size field (see Fig. 1).

In order to place bubbles in the domain, we use 2D-space subdivision with oblique cells (see Fig. 2).

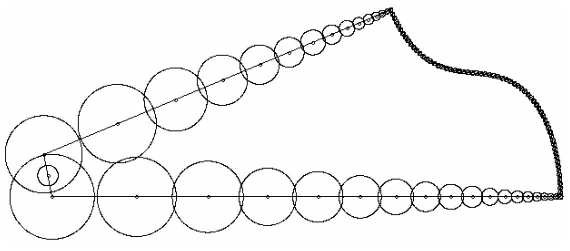
Fig. 1. Bubbles on border nodes

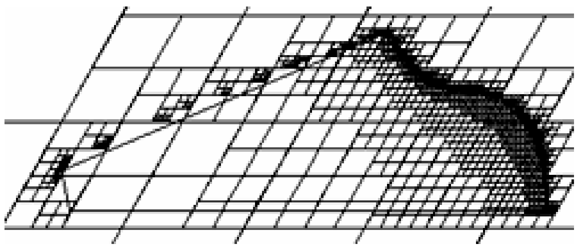
Fig. 2. Subdivision with oblique cells

Oblique cells are better for using than square cells because they let us to obtain better touchings between all bubbles, i.e, to obtain better start positions of bubbles for further algorithms.

First of all, we create a big oblique cell that includes entire domain. If we have coordinates of the center of an oblique cell, then we can get coordinates of the left bottom point, the right bottom point, the right top point, the left top point.

After creation of an oblique cell, we place four new bubbles on four corners of a cell using the field of mesh sizes for calculating diameters of new bubbles. Then, if bubbles don't intersect each other, we perform dividing the cell into four smaller oblique cells and run described above procedure for each oblique cell.

Thus, we will obtain bubbles on domain border points and bubbles inside the domain. But the number of bubbles can be too high. That's why, after the space subdivision procedure, we check the population of bubbles and intersection coefficients and, if it's needed, remove some bubbles, which have too high levels of intersection with other bubbles. Population of bubbles is controlled by using the ratio of the square occupied by bubbles to the square of the domain.

Fig. 3 says that positions of bubbles aren't optimal, i.e., not force-balanced. We use equations of motion of each bubble in order to find the best location for each bubble.

There are two types of force between two bubbles: attracting force and repelling force. If two bubbles are too close to each other, there is repelling force between them. If two bubbles are too far from each other, there is attracting force between them.

Also, we use following force properties: the force is saturated, where two bubbles are located extremely close, so that forces do not grow infinitely large; the force interaction is active only within a specified distance to reduce computation by neglecting force effects at large distances, i.e., when two bubbles are not neighbors.

Every bubble has two d.o.f. For one d.o.f. of a bubble the equation of motion is Eq. (1):

$$m \cdot \frac{d^2 x}{dt^2} + c \cdot \frac{dx}{dt} = f. \qquad (1)$$

We use simple not effective equation of motion with $m = c = 1$. f is a total summary force between one bubble and others.

In order to solve differential equations, we use integrating fourth order method Renge-Kutta.

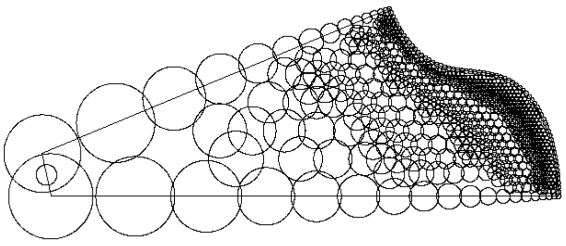
Fig. 3. Initial bubbles placement

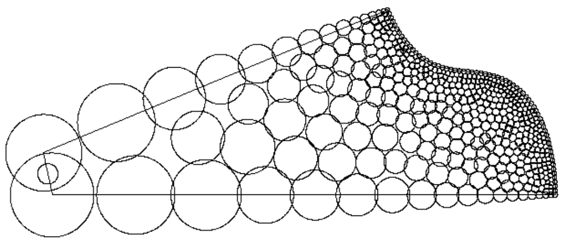
Fig. 4. Bubbles with force-balanced configuration

Applying Delaunay's triangulation. We use Delaunay's Triangulation in order to create 2D-mesh. At each iteration of Delaunay's Triangulation, a new point is added to the domain in a specified location. Points, which are added to the triangulation, are centers of bubbles on the border lines of the domain.

We create additional circumscribed rectangle (see Fig. 5) that is used in order to simplify the triangulation process at further steps.

After basic triangulation, we remove the circu

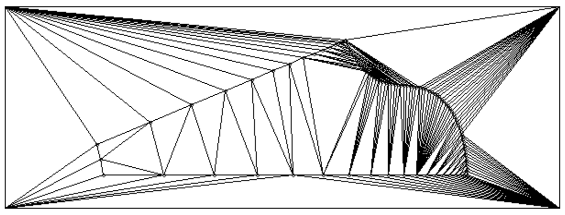
Fig. 5. Delaunay's triangulation of border nodes using the circumscribed rectangle

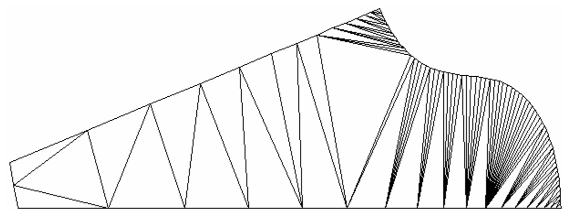
Fig. 6. Triangulation after removing geometry that is not needed anymore n

After removing geometry that is not needed anymore, we add the center point of each bubble, that is inside the domain (see Fig. 7).

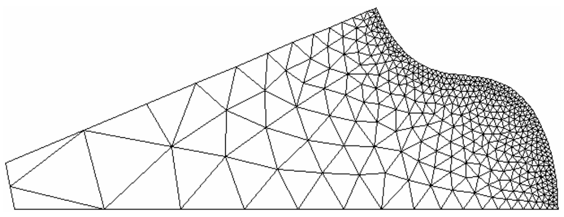
Fig. 7. Center of each bubble is added to triangulation

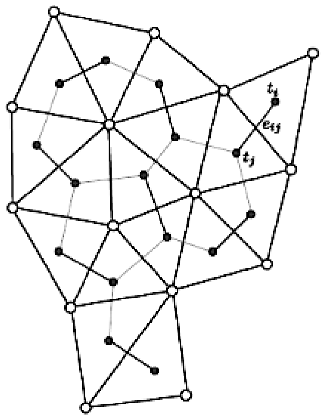

Fig. 8. A mesh and its graph

Conversion of triangles to quadrilaterals. We use of a famous algorithm of the theory of graphs: the Blossom algorithm, proposed by Edmonds in 1965 [2].

After full triangulation, we create a graph, where vertexes are triangles and edge between two vertexes, *i*, *j*, exists if a triangle of vertex *i* and a triangle of vertex *j* are adjacent and there is possibility to join two triangles by this edge in order to create a quad for triangles (see Fig. 8).

After creation a graph, we calculate cost of joining two adjacent triangles on each graph edge using Eq. (2):

$$c(e_{ij}) = 1 - h(q_{ij}), \tag{2}$$

where e_{ij} is the edge between triangle *i* and triangle *j*, q_{ij} is the quad from joining two triangles.

$$h(q_{ij}) = \max\left(1 - \frac{2}{p} \cdot \max_k\left(\left|\frac{p}{2} - a_k\right|\right),\ 0\right), \tag{3}$$

where α_k is an angle of the quad.

We also must add extra edges to the graph of the triangulation in a way that maximizes the chance of existence of a perfect matching. We add an extra-edge for every pair of triangles that do not have a general edge and have those edges that are successive in the boundary of the domain.

After adding extra edges, we unmark all nodes and edges of the graph. Then we calculate costs of joinings $c(e_{ij})$ (the cost on extra edges is 1000) and mark edges beginning from the smallest cost of joining on graph. If an edge is marked that means that two nodes (triangles) of this edge are marked, too. Then we run blossom algorithm in order to join all pairs of triangles.

If after algorithm working an extra edge has been marked, we need to make some topological changings in triangles near this extra edge (see Fig. 9). We make vertex duplication and some modification in order to unmark extra edges and mark normal edges.

When all marked extra edges are solved using vertex duplication, we obtain a quadrilateral mesh (see Fig. 10).

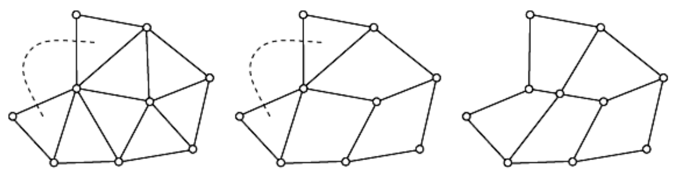

Fig. 9. Vertex duplication algorithm

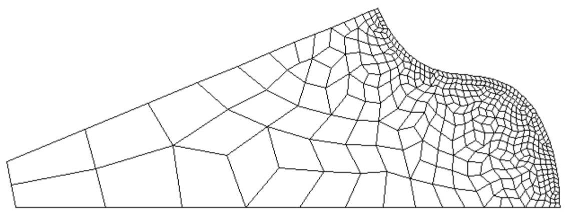

Fig. 10. Quadrilateral mesh

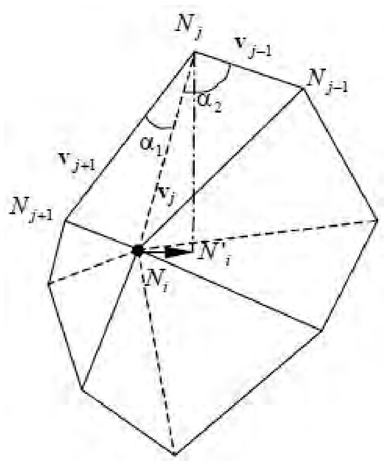

Fig. 11. Angle-based method for quadrilateral mesh

Applying mesh smoothing. In order to make mesh better, we use the algorithm of angle-based smoothing [3]. During this algorithm, the position of each internal node of the mesh (positions of external nodes are fixed) is optimized for better quality of elements, which are adjacent to the node.

The position of each internal node is optimized according to angles, which are around this node on elements, which are adjacent to this node (see Fig. 11).

There are four quadrilateral elements on Fig. 11. They contain the node that is optimized. We have also 8 nodes, which are around the node to modify. We have also 16 angles (2 angles per one node). We need to move the node in the center of the subregion in order to get the same angles in each pair of angles.

On Fig. 11, v_{j+1} is the vector from the point N_j to the point N_{j+1}, v_{j-1} is the vector from the point N_j to the point N_{j-1}.

As shown in Fig. 11, for each node N_i, there are *k* pairs of angles around it, where *k* is the number of neighboring nodes. The two adjacent angles are calculated by using Eqs. (4) and (5):

$$a_1 = \arccos\left(\frac{v_j \cdot v_{j+1}}{\|v_j\| \cdot \|v_{j+1}\|}\right), \tag{4}$$

$$a_2 = \arccos\left(\frac{v_j \cdot v_{j-1}}{\|v_j\| \cdot \|v_{j-1}\|}\right). \tag{5}$$

Then we calculate the difference between two adjacent angles by using Eq. (6):

$$b_j = \frac{a_2 - a_1}{2}. \tag{6}$$

where β_j is the angle, by which vector v_j will be moved.

Then we calculate coordinates of the new position of the node (we rotate vector v_j by angle β_j about N_j) by using Eqs. (7) and (8):

$$x' = x_0 + (x - x_0)\cdot\cos(b_j) - (y - y_0)\cdot\sin(b_j), \tag{7}$$

$$y' = y_0 + (x - x_0)\cdot\sin(b_j) + (y - y_0)\cdot\cos(b_j), \tag{8}$$

where (x_0, y_0) is the location of node N_j , (x,y) is the old location of node N_i, and (x', y') is the new location of node N_i.

By going through all the neighboring nodes, there are *k* sets of new locations for the same node N_i. We compute the final new location of node N_i by taking average of (x', y') computed from all the neighboring nodes.

Applying topological optimization. After the mesh smoothing, we apply the algorithm of topological optimization [2] and just remove some elements which must be deleted for making mesh better (see Fig. 12).

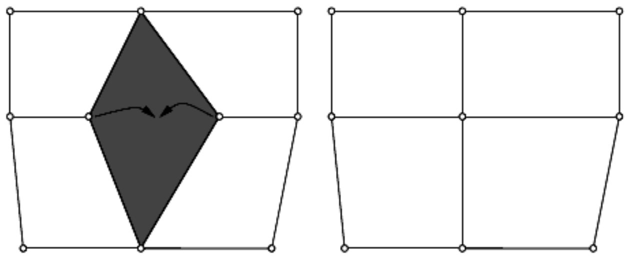

Fig. 12. Illustration of quad-vertex-merge optimization operation

4. Conclusions

In this paper, a technique for high-quality quadrilateral mesh generation that is used for shafts and hubs with non-circular profiles is proposed.

We described how we create a triangular mesh using the algorithm of automated triangulation of surfaces via bubble packing, how we convert this triangular mesh to quadrilateral mesh using the Blossom-Quad algorithm and how we optimize this mesh.

1. Shimada K. Physically-based mesh generation: automated triangulation of surfaces and volumes via bubble packing, Massachusetts Institute of Technology, 1993. 2. Remacle J.-F., Lambrechts J., Seny B., Marchandise E., A. Johnen and C. Geuzaine Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm, International Journal for Numerical Methods in Engineering, Vol. 89, Number 9, 2 March 2012. – P. 1102–1119(18). 3. Zhou T., Shimada K. An angle-based approach to two-dimensional mesh smoothing // Proceedings of the 9th International Meshing Roundtable, 2000. 4. Shimada K., Liao J.-H., Itoh T. Quadrilateral Meshing with Directionality Control through the Packing of Square Cells // Proceedings of 7th International Meshing Roundtable, 1998. 5. Pellenard B., Alliez P., Morvan J.-M. Isotropic 2D Quadrangle Meshing with Size and Orientation Control // Proceedings of the 20th International Meshing Roundtable, 2012. 6. Shewchuk J. R. Delaunay Refinement Mesh Generation, 1997.