ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2016, Vol. 56, No. 11, pp. 1901–1918. © Pleiades Publishing, Ltd., 2016.
Original Russian Text © A.I. Belokrys-Fedotov, V.A. Garanzha, L.N. Kudryavtseva, 2016, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2016, Vol. 56, No. 11, pp. 1931–1948.

Generation of Delaunay Meshes in Implicit Domains with Edge Sharpening

A. I. Belokrys-Fedotov, V. A. Garanzha*, and L. N. Kudryavtseva**

Federal Research Center "Computer Science and Control", Russian Academy of Sciences, ul. Vavilova 40, Moscow, 119333 Russia

Moscow Institute of Physics and Technology, Institutskii per. 9, Dolgoprudnyi, Mocow oblast, 141700 Russia

*e-mail: *garan@ccas.ru; arnir@rambler.ru; **belokrys.fedotov@gmail.com*

Received December 21, 2015; in final form, April 26, 2016

Abstract—A variational algorithm for the construction of 3D Delaunay meshes in implicit domains with a nonsmooth boundary is proposed. The algorithm is based on the self-organization of an elastic network in which each Delaunay edge is interpreted as an elastic strut. The elastic potential is constructed as a combination of the repulsion potential and the sharpening potential. The sharpening potential is applied only on the boundary and is used to minimize the deviation of the outward normals to the boundary faces from the direction of the gradient of the implicit function. Numerical experiments showed that in the case when the implicit function specifying the domain is considerably different from the signed distance function, the use of the sharpening potential proposed by Belyaev and Ohtake in 2002 leads to the mesh instability. A stable version of the sharpening potential is proposed. The numerical experiments showed that acceptable Delaunay meshes for complex shaped domains with sharp curved boundary edges can be constructed.

Keywords: Delaunay meshes, elastic networks, sharp edge sharpening, variational method, surface reconstruction, implicitly specified domains.

DOI: 10.1134/S096554251611004X

INTRODUCTION

There is a large class of problems in which the computational domain is specified implicitly as a isolevel surface of a function. There are various methods for the generation of such functions. In particular, *R*-functions (see [1]) make it possible to construct domains of a complex shape by combining implicit functions of elementary domains using Boolean operations.

There exist efficient algorithms for constructing tetrahedral meshes in implicit domains. Typically, instead of specifying the domain boundary explicitly, these algorithms use a fast algorithm for the calculation of the intersection point of this boundary with an arbitrary straight segment. In [2], a fast algorithm for the construction of tetrahedral meshes with a theoretical guarantee of quality in the case of domains with a smooth boundary was proposed. In [3, 4], an algorithm for constructing meshes in implicit domains was described. In this algorithm, the vertices of the 3D Delaunay mesh are lifted to the surface of a paraboloid of rotation forming a convex polyhedron in the 4D space inscribed in the paraboloid of rotation. The difference between the volume of the paraboloid of rotation and the volume of the inscribed polyhedron is used as a functional describing the distortion of the Delaunay mesh. When this functional is minimized, the Delaunay mesh is self-organized both due to the displacement of the mesh vertices and rebuilding the mesh connectivity structure. In [5], the implicit representation of the domain is used for the construction of hexahedral meshes in a domain and quadrilateral meshes on a surface.

The starting point for the present work is the paper [6], where a simple algorithm for the construction of 2D and 3D Delaunay meshes was proposed. This algorithm was implemented as a one page long code in MATLAB. The idea of the algorithm in [6] is as follows. All the mesh edges are interpreted as elastic struts with a given distribution of length, and the mesh construction is reduced to the construction of an equilibrium state of the elastic network. The target edge lengths are specified such that the struts are slightly compressed in the equilibrium state. The mesh vertices that move out of the domain in the process of relaxation are projected back onto the domain boundary. If the Delaunay property is violated in the process of relaxation, the mesh is rebuilt to restore it. The implicit domain is represented by signed dis-

tance from the boundary. A remarkable property of this algorithm is the topological regularity of the resulting meshes even in the case of chaotic initial approximations.

In [7, 8], an algorithm that can be used for domains with a nonsmooth boundary was proposed. This algorithm makes it possible to automatically reproduce sharp boundary edges as a subset of the mesh boundary edges using special sharpening forces, which were proposed for the visualization of implicit domains in [9]. No a priori sharp boundary edge specification or detection is necessary.

In this paper, we show that the self-organization algorithm can be implemented as a solution of a variational problem and propose a modification of the sharpening force that can be used for implicit functions that strongly deviate from the signed distance functions. In this case, the original sharpening terms from [9] lead to numerical instability.

Note that there is an alternative approach to the generation of tetrahedral meshes in implicit domains with nonsmooth boundaries. In this approach, the boundary surface is first reconstructed by building a polyhedron that approximates the boundary using edge sharpening algorithms, such as those used in [9, 10]. Then, the tetrahedral mesh can be obtained by successively applying surface mesh generation algorithms and then 3D mesh generation algorithms (a review of such methods can be found in [11]). For this purpose, algorithms that guarantee the construction of a tetrahedral mesh given a surface triangulation as described in [12] can be used. In [13], this problem was solved using the advancing front technique, while the tetrahedral mesh generation algorithm with a given boundary triangulation is used only in the subdomains where the fronts meet.

In this paper, we do not consider the comparison with such a class of algorithms. The objective is to demonstrate the ability of the self-organization algorithm to generate Delaunay meshes with automatic edge sharpening for complex shaped implicit domains.

In various Delaunay mesh generation algorithms, typically a certain number of flat tetrahedra called slivers appear. The removal of flat tetrahedra usually results in the violation of the Delaunay property. In [8], a variational approach was used to remove slivers; it produces very good meshes that, however, do not satisfy the Delaunay property. A review of the variational mesh optimization methods can be found in [14].

1. DESCRIPTION OF THE SELF-ORGANIZATION ALGORITHM FOR THE CONSTRUCTION OF MESHES IN IMPLICIT DOMAINS

Implicit domain specification. Consider a bounded domain $\Omega \subset \mathbb{R}^3$. We assume that the boundary of Ω is Lipschitz continuous and piecewise regular. Let a function $u(x)$: $\mathbb{R}^3 \to \mathbb{R}$ be given such that the condition $u(x) < 0$ is satisfied at the interior points of the domain, and $u(x) > 0$ outside the closure of the domain. We assume that $u(x)$ is Lipschitz continuous, piecewise smooth, and its derivatives along a nonsingular vector field that is transversal to $\partial\Omega$ exist and are bounded away from zero in a finite layer S near $\partial\Omega$. In fact, this assumption means that the behavior of the implicit function near the boundary is qualitatively similar to the behavior of the signed distance to the boundary. This assumption can be formalized by requiring, e.g., that there exists a quasi-isometric mapping $y(x)$: $\mathbb{R}^3 \to \mathbb{R}^3$ such that $y(\Omega) = \Omega_y$ and $u(x) = d_s(y(x))$, where $d_s(y)$ is the signed distance function for the domain Ω_y. The Lipschitz constant for the signed distance function does not exceed unity, while the range of the Lipschitz constants for the function $u(x)$ is defined by the quasi-isometry constant of the mapping $y(x)$. The case when the local Lipschitz constant changes abruptly across a sharp edge is the most difficult case from the computational point of view because it can lead to the computational instability as shown in this paper.

Since the mesh generation technique proposed in this paper is heuristic, we do not use rigorous formulations of the requirements for the domain.

Tetrahedral mesh in the implicit domain. Let us discuss what kind of 3D computational meshes generated by a mesh generation algorithm can be considered to be acceptable. The 3D domain Ω specified by the inequality $u(x) < 0$ is approximated by a polyhedron Ω_h with the boundary $\partial\Omega_h$ consisting of planar triangles glued along their whole edges. Inside Ω_h, a normal tetrahedral partition $\mathcal{T}$ is constructed such that the set of its boundary triangles is a subset of the faces of this partition. Furthermore, it is required that the surface $\partial\Omega_h$ be pointwise convergent to $\partial\Omega$ as the size of the boundary mesh edges tends to zero. In addition, it is required that the piecewise constant field ν_h of the unit normal vectors to $\partial\Omega_h$ be convergent to the piecewise continuous field of the unit normal vectors ν to the surface $\partial\Omega$. The parameter h plays the role of cells size; i.e., it is assumed that the maximum edge length in the mesh $\mathcal{T}$ does not exceed Ch, where C is a constant.

Thus, it is assumed that there exists a homeomorphism ψ_h: $\mathbb{R}^3 \to \mathbb{R}^3$ such that $\psi_h(\Omega_h) = \Omega$, $\psi_h(\partial\Omega_h) = \partial\Omega$, and (a) for all $p \in \partial\Omega_h$, it holds that $|p - \psi_h(p)| \to 0$ as $h \to 0$ and (b) for all $p \in \Omega_h$ that do not belong to the set of edges of $\mathcal{T}$, it holds that $|\nu_h(p) - \nu(\psi_h(p))| \to 0$ as $h \to 0$.

However, the numerical construction of such a homeomorphism is too computationally expensive; for this reason, heuristic proximity measures are used.

The most popular approach to the computational mesh generation is as follows. First, a triangular surface mesh approximating the boundary of the domain Ω is constructed, and then the 3D mesh decomposing the resulting polyhedron is generated. In the algorithm proposed in this paper, we use another approach—the 3D mesh inside the domain and the boundary mesh are generated simultaneously.

Self-organization algorithm. The idea behind the self-organization algorithm of a point system is as follows. Using a mechanical analogy, assume that the mesh vertices are point masses that repulse from each other thus modeling a hypothetical elastic medium. As a result of repulsion, a point can go beyond the domain Ω. The points that have been moved out of Ω are projected back onto its boundary. As a result, the algorithm produces a hypothetical "equilibrium" mesh in which all the forces acting on the point mass are balanced. The repulsion forces must be chosen to make the elastic medium compressed in the equilibrium state.

The spatial distribution of edge lengths of the computational mesh is given by the scale function $f_h(x)$: $\mathbb{R}^3 \to \mathbb{R}$, $f_h(x) > 0$, which can be interpreted as the target edge length at the point x. It was shown in [6] that it is more convenient to use the relative scale function $h(x) \geq 1$ that specifies the ratio of the edge lengths at different points of the domain. Then, the absolute size is given by the function $f_h(x) = Mh(x)$, where the coefficient M depends on the volume of the domain Ω (which is typically not known a priori) and on the fixed mesh size. The last dependence can be replaced by a dependence on the total number of mesh vertices n.

Assume that a set of points $\mathcal{E} = \{p_1, p_2, ..., p_n\}$ in $\mathbb{R}^3$ is given. Denote by $\mathcal{T}(\mathcal{E})$ a tetrahedral Delaunay partition without tetrahedra that do not belong to the computational domain. The union of all tetrahedra in $\mathcal{T}$ makes up the polyhedron Ω_h, which in general may or may not approximate the domain Ω. Denote by $\mathcal{T}_e$ the set of edges of the tetrahedral mesh and by $\mathcal{F}_b$ the set of boundary faces. Denote by P the $3 \times n$ matrix where the ith column is the vector p_i.

Let us associate with each mesh the elastic potential

$$W(P) = W_e(P) + W_s(P),$$

where $W_e(P)$ is the vertex repulsion potential and $W_s(P)$ is the sharpening potential used to align the boundary faces to make their normals codirectional with the gradient of the implicit function.

The repulsion potential is written as

$$W_e(P) = \sum_{e \in \mathcal{T}_e} w_e(e),$$

$$w_e(e) = \begin{cases} \frac{1}{2}|L - L_0|^2, & L < L_0, \\ 0, & L \geq L_0, \end{cases}$$

where

$$L = |p_i - p_j|$$

is the length of the edge e and $L_0(e)$ is the target length of e, which is given by the formula

$$L_0(e) = Mh\left(\frac{1}{2}(p_i + p_j)\right).$$

As an alternative, the logarithmic repulsion potential

$$w_e(e) = \begin{cases} L_0^2\left(\frac{L}{L_0} - 1 - \log\left(\frac{L}{L_0}\right)\right), & L < L_0, \\ 0, & L \geq L_0 \end{cases}$$

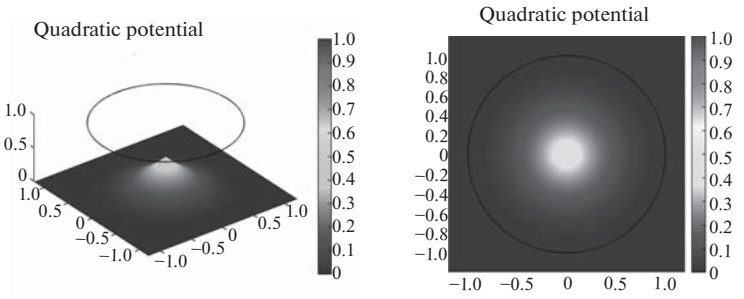

Fig. 1. The quadratic potential for a single vertex.

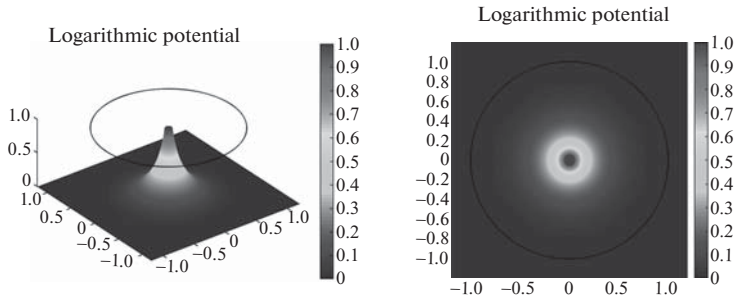

Fig. 2. The logarithmic potential for a single vertex.

may be used. The sharpening potential is written as

$$W_s(P) = \sum_{f \in \mathcal{F}_b} w_s(f),$$

where the contribution of the boundary face f has the form

$$w_s(f) = \frac{1}{2} \frac{\text{area}(f)}{|\nabla u(c)| L_0(c)^2} \sum_{i=1}^{3} ((p_i - c)^{\mathrm{T}} \nabla u(c))^2.$$

Here p_1, p_2, and p_3 are the vertices of the face f, $c = (p_1 + p_2 + p_3)/3$ is the centroid of this face, and $L_0(c) = Mh(c)$.

The behavior of the quadratic and logarithmic potentials is illustrated in Figs. 1 and 2, respectively. In these examples, the elastic energy is a function of two 2D vectors—the constant vector $p = 0$ and the variable vector $(x_1, x_2)^{\mathrm{T}}$. The figures show the resulting function of two variables. The qualitative behavior of the functionals is similar, but the logarithmic potential is singular and does not admit edges of zero length. In the 3D case, the behavior of the functionals is qualitatively similar.

Figures 3 and 4 illustrate the behavior of the functionals in the case when six 2D vertices are given and the seventh vertex is a variable vector x_1, x_2 that is an interior vertex of the triangulation consisting of six triangles. The bold lines show the domain where the Hessian matrices are positive definite.

The fixed points are situated at the vertices of the regular hexagon with the side equal to the target length L divided by a factor of 1.2.

This simple illustration explains why the elastic mesh must be in a slightly compressed state. It is also clear that, for the configuration consisting of six points, there exists an interior convexity zone around the geometric center at which the minimum is attained.

An elastic mesh is said to be in the equilibrium state if, at the current value of the vertex matrix P, the elastic energy $W(P)$ attains the local minimum provided that the vertices of the faces in $\mathcal{F}_b$ lie on the surface $u(x) = 0$.

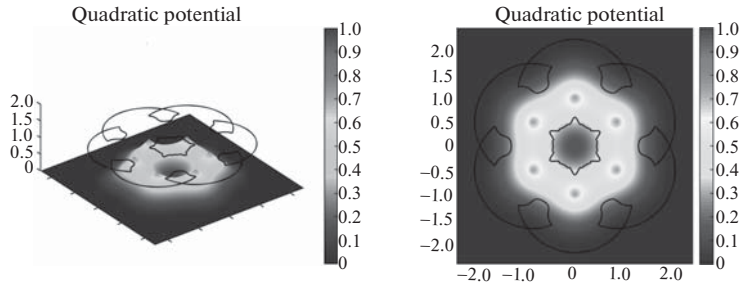

Fig. 3. The quadratic potential for a configuration consisting of seven vertices and the domain where the Hessian matrix is positive definite.

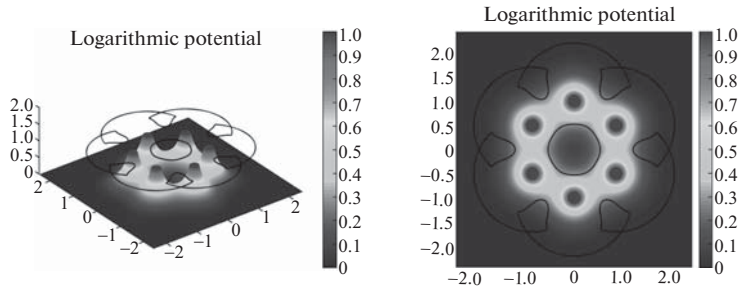

Fig. 4. The logarithmic potential for a configuration consisting of seven vertices and the domain where the Hessian matrix is positive definite.

Therefore, the equilibrium state of the elastic mesh is a solution $\overline{P}$ to the variational problem

$$\min W(P)$$

subject to the conditions

$$u(p_i) \le 0, \quad i = 1,\ldots,n.$$

The necessary minimum condition is the well-known Karush–Kuhn–Tucker (KKT) condition: there exists a vector $\Lambda = (\lambda_1, \ldots, \lambda_n)^{\mathrm{T}}$ such that

$$L_W(\overline{P}) = \min L_W(P),$$

where

$$L_W(P) = W(P) + \sum_{i=1}^{n} \lambda_i u(p_i)$$

is the Lagrange function and λ_i are the Lagrange multipliers satisfying the complementary slackness condition $\lambda_i u(\overline{p}_i) = 0$ and the dual feasibility condition $\lambda_i \ge 0$. The solution $\overline{P}$ also satisfies the constraints of the original problem $u(\overline{p}_i) \le 0$. If the functions W and u are smooth, the solution should satisfy the set of differential equations

$$\frac{\partial W}{\partial p_i} + \lambda_i \frac{\partial u}{\partial p_i} = 0.$$

For the interior vertices, we have $u(p_i) < 0$ and $\lambda_i = 0$, so that the standard stationarity conditions

$$\frac{\partial W}{\partial p_i} = 0$$

are satisfied. The boundary vertices p_i satisfy the equality $u(p_i) = 0$, and the Lagrange multipliers satisfy the inequality $\lambda_i > 0$. Here we assume that, due to the "compressed state", the reaction force on the boundary is nonzero.

If the elastic energy and the function $u(x)$ that specifies the constraints are not smooth, the subgradient form of the KKT conditions should be considered. Taking into account the heuristic nature of the proposed algorithm, we will not consider this version of the problem statement. In addition, the computation of the tangent cones for irregular functions is too computationally expensive. In practice, the approximation of the gradient is calculated using a finite difference method. The coordinates of the boundary vertices are found using the gradient projection technique: the points are moved along the antigradient of W, and if they travel beyond the domain boundary, they are projected back onto this boundary. The fairly complicated set of heuristic rules described below can be interpreted as a subgradient version of the minimization algorithm.

As a result, the self-organization algorithm is formulated as follows. Let the initial matrix of vertex positions P^0 be given.

For $k = 0, 1, \ldots$

(a) for the vertex matrix P^k, the tetrahedral Delaunay partition is constructed;

(b) the tetrahedra that are not in Ω are removed using a special set of criteria, the resulting mesh is denoted by $\mathcal{T}^k$, and the set of its boundary faces by $\mathcal{F}_b^k$;

(c) one step of the algorithm of conditional minimization of the elastic energy by the displacement of vertices is made:

$$\tau^k = \arg\min_{\tau}(W(V(P^k + \tau\delta P^k))), \quad P^{k+1} = V(P^k + \tau^k\delta P^k),$$

where δP^k is the matrix of the minimization direction.

The operator V projects each vertex p lying outside the domain onto the surface $u(x) = 0$ using the simple iterative algorithm

$$p^{m+1} = p^m - \tau_1 \frac{u(p^m)\nabla u(p^m)}{\left|\nabla u(p^m)\right|^2}. \tag{1}$$

Here the parameter m is the index of the local iteration. If $u(x)$ is a linear function and $\tau_1 = 1$, formula (1) describes the normal projection on the plane $u(x) = 0$. In the general case, iterations (1) should be repeated until the deviation of the point p^m from $\partial\Omega$ becomes less than a threshold ϵ_b. In practice, this is reduced to checking the inequality

$$\left|u(p^m)\right| \le \epsilon\left|\nabla u(p^m)\right|, \tag{2}$$

where $\epsilon = 10^{-8}$ can be considered as the accuracy of projection.

The elastic relaxation and the mesh connectivity reconstruction are alternated until the minimum of the functional is attained on a Delaunay mesh or a stopping criterion (the quality of the mesh and the quality of the boundary approximation are sufficiently good) is fulfilled.

The energy minimization procedure seems to be conventional; however, the computation of the minimization directions is based on intricate empirical rules that are responsible for the stable reconstruction of sharp edges.

In order to calculate the gradient of the elastic potential, the contribution from all edges and all boundary faces of the mesh must be summarized:

$$\frac{\partial W}{\partial p_i} = \sum_{e\in\mathcal{T}_e:\, p_i\in\mathcal{V}(e)} \frac{\partial w_e(e)}{\partial p_i} + \sum_{f\in\mathcal{F}_b:\, p_i\in\mathcal{V}(f)} \frac{\partial w_s(f)}{\partial p_i}.$$

Here $\mathcal{V}(Q)$ is the set of vertices of the geometric object Q.

Let us write out the derivative of the local potential for the boundary face:

$$\frac{\partial w_s}{\partial p_i} = \frac{\mathrm{area}(f)}{(L_0(c))^2} g(p_i - c)^{\mathrm{T}} g + \frac{\sum_{j=1}^{3}((p_j - c)^{\mathrm{T}} g)^2}{2(L_0(c))^2} \frac{\partial \mathrm{area}(f)}{\partial p_i}.$$

For simplicity, the dependence of the gradient direction $g(c) = \nabla u(c)/|\nabla u(c)|$ on p_i is neglected. Furthermore, as will be shown below, the second term on the right-hand side of this formula is dropped as well when the minimization direction is calculated.

The derivative of the quadratic potential on the edge $e = (p_l, p_m)$ with respect to the vector p_1 has the form

$$\frac{\partial w_e}{\partial p_l} = \begin{cases} \left(1 - \frac{L_0}{L}\right)(p_l - p_m), & L < L_0, \\ 0, & L \geq L_0, \end{cases}$$

while the derivative of the logarithmic potential is

$$\frac{\partial w_e}{\partial p_l} = \begin{cases} \left(1 - \frac{L_0}{L}\right)\frac{L_0}{L}(p_l - p_m), & L < L_0, \\ 0, & L \geq L_0. \end{cases}$$

Numerical experiments showed a certain advantage of the logarithmic potential over the quadratic potential. In particular, the use of the logarithmic potential seems to lead to a more efficient self-organization process, especially when the initial mesh vertices are concentrated in a wrong region.

It is convenient to introduce the so-called "repulsion forces" and "sharpening forces," which simply indicate the contribution of the repulsion and sharpening potentials, respectively, to the minimization direction vector.

Dropping details, these "forces" can be defined by

$$\delta p_i^k = -\frac{1}{d_{e_i}^k}\frac{\partial W_e}{\partial p_i}(P^k) - \frac{1}{d_{s_i}^k}\frac{\partial W_s}{\partial p_i}(P^k) = F_e(p_i) + F_s(p_i).$$

This formula implies that the gradient of the repulsion potential and the gradient of the sharpening potential are scaled independently of each other. This seems strange, but in fact this is a consequence of the assumption that the repulsion and sharpening gradients are orthogonal. Therefore, this formula is a simplified version of matrix scaling. The scaling factors are calculated using estimates of the second-order derivatives of the potentials. In practice, these gradients are not orthogonal, but the formula above turned out to be effective.

In our elastic model, the motion of the vertices is governed by the relaxation model rather than by Newton's law; therefore, the forces defined above are speculative. However, their use helps understand the algorithm's behavior.

To write out the final expressions for the calculation of the forces, it is useful to introduce the following notation.

Let $\mathrm{star}_e(p_i)$ denote the set of the mesh edges outgoing from the vertex p_i, and $\mathrm{star}(p_i)$ denote the set of vertices of these edges without p_i. It is also useful to introduce the concept of the vertex star of the surface: $\mathrm{star}_f^s(p_i)$ is the set of all boundary faces adjacent to the boundary vertex p_i, $\mathrm{star}_e^s(p_i)$ is the set of edges, and $\mathrm{star}^s(p_i)$ is the set of boundary vertices of the star. In all cases, it is assumed that, in each boundary star, its elements are ordered around p_i counterclockwise as seen from the outside of the domain.

Then, for the interior vertex p_i, we have

$$F_e(p_i) = -\frac{\sum\limits_{p_j \in \mathrm{star}\, p_i} \phi_e(p_i, p_j)(p_i - p_j)}{\sum\limits_{p_j \in \mathrm{star}\, p_i} \phi_e(p_i, p_j)},$$

where

$$\phi_e(p_i, p_j) = \left(\frac{L_0}{L} - 1\right)\frac{L_0}{L}, \quad L = |p_i - p_j|, \quad L_0 = Mh\left(\frac{1}{2}(p_i + p_j)\right).$$

On the boundary, this formula is modified as

$$F_e(p_i) = -\frac{\sum\limits_{p_j \in \mathrm{star}\, p_i} \phi_e(p_i, p_j)\Pi_{ij}(p_i - p_j)}{\sum\limits_{p_j \in \mathrm{star}\, p_i} \phi_e(p_i, p_j)}.$$

The operator Π_{ij} in this formula is defined by

$$\Pi_{ij}(p_i - p_j) = \begin{cases} p_i - p_j, & p_j \notin \mathrm{star}^s(p_i), \\ (I - \nu_i \nu_i^{\mathrm{T}})(p_i - p_j) + \max(0, \nu_i^{\mathrm{T}}(p_i - p_j))\nu_i, \end{cases}$$

where the vector ν_i is the discrete unit outward normal to the boundary mesh at the vertex p_i. It is calculated by the angular averaging of the normals to the adjacent triangles:

$$\nu_i = -\frac{\sum\limits_{f \in \mathrm{star}_f^s(p_i)} \alpha_i(f)\nu(f)}{\left|\sum\limits_{f \in \mathrm{star}_f^s(p_i)} \alpha_i(f)\nu(f)\right|},$$

where $\nu(f)$ is the unit outward normal to the face f and $\alpha_i(f)$ is the angle of f at the vertex p_i. This representation of the operator Π_{ij} eliminates the possibility that the repulsion force will push the boundary vertex inside the domain, which otherwise would be possible on a concave or saddle-like surface.

Another modification of the repulsion force is used when an edge of a Delaunay tetrahedron is positioned as an internal strut between the opposite sides of the acute wedge near its tip. For each acute wedge, a dilatation is specified that "opens" this wedge, i.e., makes it less acute. Consider an internal Delaunay edge e belonging to the Delaunay tetrahedron T, and assume that both vertices p_1 and p_2 of this edge lie on the domain boundary. Let the gradients u at these vertices be contradirectional in the following sense. For the vertices p_1 and p_2, we specify a set of directions of the gradients G_1 and G_2 that includes $\nabla u(p_i)/|\nabla u(p_i)|$ and $\nabla u(c(f))/|\nabla u(c(f))|, f \in \mathrm{star}_f^s(p_i)$; i.e., in addition to the direction at the vertex itself, the directions of the gradients at the centroids of the adjacent faces are used.

Now, we calculate the directions g_1^* and g_2^* as the most contradirectional ones:

$$\angle(g_1^*, g_2^*) = \max \angle(g_1, g_2), \quad g_1 \in G_1, \quad g_2 \in G_2, \tag{3}$$

$\alpha(T) = \pi - \angle(g_1^*, g_2^*)$. If the maximal angle is greater than $3\pi/4$, then a special stretching coefficient k is calculated. Otherwise, we set $k = 1$. Consider two planes $g_i^{*\mathrm{T}}(x - p_i) = 0$, $i = 1, 2$. These planes intersect on a line, which is orthogonal to the vectors g_1^* and g_2^*. Let us construct the circular cylinder with the axis coinciding with this line and that passes through the centroid $c(T)$ of T. As a result, we obtain two quantities—the cylinder radius R and the length $D = R\alpha$ of the part of the cylinder that is inside the hypothetical acute wedge between two planes. If it holds that

$$D < Mh(c(T)),$$

then

$$k = \frac{D}{Mh(c(T))}. \tag{4}$$

Otherwise, we set $k = 1$. The geometric meaning of the parameters R, α, and D is illustrated in Fig. 5.

If $k < 1$, then the length L of the edge e should be recalculated in the stretched system of coordinates

$$x' = \left((I - \nu\nu^{\mathrm{T}}) + \frac{1}{k}\nu\nu^{\mathrm{T}}\right)x, \quad \nu = \frac{g_1^* - g_2^*}{|g_1^* - g_2^*|}. \tag{5}$$

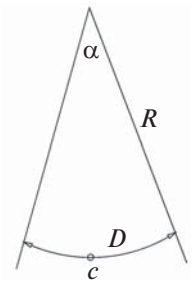

Fig. 5. Characteristics of the acute angle.

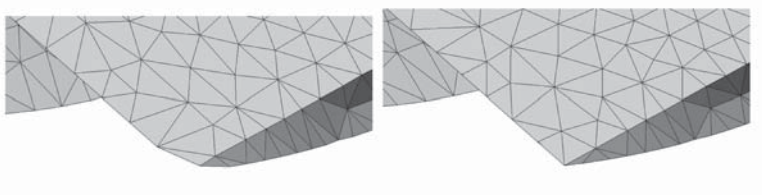

Fig. 6. Angle correction.

This transformation implies that the edge is stretched along the direction ν and L in the expression for ϕ_e is replaced with L':

$$L' = \left(L^2 + \left(\frac{1}{k^2} - 1 \right) (\nu^{\mathrm{T}}(p_1 - p_2))^2 \right)^{\frac{1}{2}}.$$

This correction decreases the degree of compression of the of the transversal edge near the tip of an acute wedge and makes it possible to decrease the inclination of the interior edges from the direction ν.

Numerical experiments (see Fig. 6) showed that this correction improves the quality of approximation of sharp edges.

Consider the expression for the sharpening force

$$F_s(p_i) = -\frac{\sum\limits_{f \in \mathrm{star}_f^s\, p_i} \psi(p_i, f)\tilde{g}}{\left| \sum\limits_{f \in \mathrm{star}_f^s\, p_i} \Psi(p_i, f) \right|}, \quad \tilde{g} = \tilde{g}(p_i, f) = \frac{\nabla u(\tilde{c}(f))}{|\nabla u(\tilde{c}(f))|},$$

where

$$\Psi(p_i, f) = \frac{\mathrm{area}(f)}{(Mh(\tilde{c}(f)))^2}, \quad \psi(p_i, f) = \Psi(p_i, f)(c - p_i)^{\mathrm{T}}\tilde{g}.$$

Here $\tilde{c}(f)$ is the point of intersection of the implicit surface

$$u(x) = \frac{1}{3}(u(p_1(f)) + u(p_2(f)) + u(p_3(f))),$$

where p_i are the vertices of the face f, with the straight line

$$x(t) = c(f) + \nu(f)t,$$

which is orthogonal to the face f and passes through its centroid. Thus, the proposed sharpening force differs from that used in [9] due to the additional projection on the boundary.

It turned out that this modification plays the key role in the case when the implicit function $u(x)$ is significantly different from the signed distance function; in particular, when the gradients of this functions on different banks of the sharp edge differ by an order of magnitude. In this case, the original sharpening

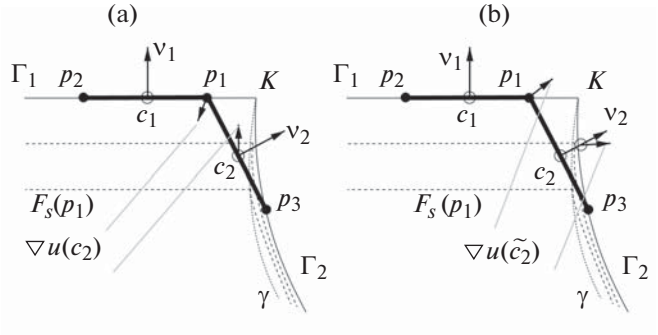

Fig. 7. Unstable configuration for the sharpening force (a) and the stabilized sharpening (b).

force can lead to the computational instability as shown in Fig. 7a. The isolines of the implicit function are shown by dashed lines. It is seen that the function is significantly anisotropic.

The boundary vertices of the mesh are divided into "smooth" and "nonsmooth", depending on the absolute value of the discrete Laplace–Beltrami operator at the mesh vertices; this operator acts on the coordinate representation of the surface. The Laplace–Beltrami operator is approximated using the Floater scheme, in which the integral of the Laplace–Beltrami operator over a closed circle around the vertex and the theorem on the mean value of a harmonic function over a circle are used. The expression for the discrete operator is

$$Bx(p_i) = \sum_{f \in \mathrm{star}_f^s(p_i)} \tan\left(\frac{1}{2}\alpha_{j+\frac{1}{2}}\right)\left(\frac{p_j - p_i}{|p_j - p_i|} + \frac{p_{j+1} - p_i}{|p_{j+1} - p_i|}\right). \tag{6}$$

Here p_i, p_j, and p_{j+1} are the vertices of the triangle f, and $\alpha_{j+1/2}$ is the angle of f at the vertex p_i. In order to label nonsmooth vertices, the threshold

$$|Bx(p_i)| > 1/2$$

is used. Note that the expression for $Bx(p_i)$ is dimensionless and is used as an angular measure of deviation from flatness at the boundary vertex p_i of the mesh. It is clear that this measure may tend to mark as nonsmooth the vertices of the coarse mesh lying on smooth surfaces.

Consider how the forces for nonsmooth vertices are modified. Let us calculate the orthogonal decomposition of the repulsion and sharpening forces at the point p_i with respect to the direction of the implicit function gradient $\nabla u(p_i)$:

$$F_e = F_{en} + F_{e\tau}, \quad F_{en} = \frac{1}{|\nabla u|}\nabla u^{\mathrm{T}} F_e, \quad F_{e\tau} = F_e - F_{en},$$

$$F_s = F_{sn} + F_{s\tau}, \quad F_{sn} = \frac{1}{|\nabla u|}\nabla u^{\mathrm{T}} F_s, \quad F_{s\tau} = F_s - F_{sn}.$$

The tangent component of the repulsion force is orthogonalized with respect to the tangent component of the sharpening force

$$\hat{F}_{e\tau} = F_{e\tau} - \frac{w}{|F_{s\tau}|^2} F_{s\tau} F_{e\tau}^{\mathrm{T}} F_{s\tau},$$

and the final formula for the repulsion force takes the form

$$F_e = F_{en} + \hat{F}_{e\tau}.$$

The parameter w depends on the angle between the vectors ∇u and F_s:

$$w = \min\left(\frac{\sin \angle(\nabla u, F_s)}{\sin\frac{\pi}{9}}, 1\right).$$

It is introduced to take into account the case when the tangent component of the sharpening force vanishes or is very small.

The decomposition described above allows the mesh vertices to move to sharp edges under the sharpening forces, and the repulsion force is responsible for the distribution of the vertices along the sharp edges and near them.

For smooth boundary vertices, the tangent component of the sharpening force is set to zero, and the repulsion force is not modified so that

$$F = F_e + F_{sn}.$$

This decomposition makes it possible to smooth the deviation of the surface mesh from the isosurface of the implicit function, while the shape of the triangles on smooth fragments of the boundary is determined by the repulsion force and is close to regular.

Below, we write the final expressions for the recalculation of the mesh vertex coordinates. First, the maximal relative displacement

$$D_{\max} = \max_i \frac{|F(p_i)|}{Mh(p_i)}, \quad \tau = \min(0.45, 0.45/D_{\max})$$

is computed, and the intermediate vertex positions are found:

$$\tilde{p}_i = p_i^k + \tau F(p_i^k), \quad i = 1,\ldots,n.$$

If the angle between $\nabla u(p_i^k)$ and $\nabla u(\tilde{p}_i)$ is greater than $\pi/18$ at a certain i, then the step size is divided by two:

$$\tilde{p}_i = p_i^k + \frac{1}{2}\tau F(p_i^k), \quad i = 1,\ldots,n.$$

Next, we find

$$p_i^{k+1} = V(\tilde{p}_i),$$

where V is the projector defined in (1).

Consider practical techniques for the computation of the error measures related to the approximation of the boundary. Let f be a boundary face. Denote by $\tilde{c}(f)$ the intersection point of the surface

$$u(x) = 0$$

with the straight line

$$x(t) = c(f) + \nu(f)t,$$

which is orthogonal to the plane of f and passes through its centroid. Here $p_i(f)$ are the vertices of f. Consider the tetrahedron $T_b(f)$ with the vertices $\tilde{c}(f)$, $p_1(f)$, $p_2(f)$, and $p_3(f)$. Denote its height over the plane of f by $H(f)$. The height of this auxiliary tetrahedron can be used as the maximum error of the deviation of $\partial\Omega_h$ from $\partial\Omega$:

$$E_d^\infty = \max_{f\in\partial\Omega_h} H(f).$$

Then, the integral deviation is the sum of the volumes of all auxiliary tetrahedra:

$$E_d^1 = \sum_{f\in F_b} \mathrm{vol}(T_b(f)).$$

The ratio $E_d^1/\mathrm{vol}(\Omega_h)$ can be used as a relative error measure. To check the quality of the sharp edge reconstruction, the maximal measure of the domain shape error should also be used:

$$E_s^\infty = \max_{f\in\partial\Omega_h} \frac{H(f)}{L_{\max}(f)};$$

here $L_{\max}(f)$ is the maximal length of the edge of the boundary face f. By integrating the shape error over the domain boundary, we can obtain the mean error of the shape approximation E_s^1.

These error measures are heuristic, but in practice they are close to the true errors if the meshes are sufficiently fine.

Thus the iterative self-organization process can be stopped when the shape proximity threshold is reached and the distribution of lengths of the mesh edges is close to the prescribed one.

Note that the declared aim of the algorithm—to construct an equilibrium mesh—is often not achieved in practice. Geometric oscillations can occur when small displacements of the vertices result in changes in the mesh connectivity, which is accompanied by the appearance of sliver tetrahedra, which turn out to be far from equilibrium. For this reason, the use of the stopping rule that the forces (displacements) should be small turned out to be ineffective. Note, however, that the shape of the computational domain remains stable.

Specification of the initial distribution of vertices. A simple way of specifying the initial distribution of mesh vertices is to specify the set $\mathscr{C}$ of the vertices of the dense hexagonal ball packing with the size h. Next, the points satisfying the inequality $u(p) \geq 0$ are removed from this set, and the set $\mathscr{C}$ is sparsified by removing the points with a probability proportional to the value of the function $1 - 1/h(x)^3$ at these points (see [6]). This method allows one to make the initial distribution roughly consistent with the mesh size distribution. A more efficient algorithm based on the construction of the octree of the domain Ω can also be used.

Note that the initial distribution of the vertices may also be specified randomly. Such a distribution is convenient for testing purposes because the aim of this algorithm is to generate meshes that weakly depend on the initial distribution of vertices. The numerical results confirm such a behavior of the proposed algorithm but, as could be expected, the construction of the mesh in the case of a bad initial approximation requires much more iterations of the vertex displacement algorithm and reconstruction of the mesh connectivity structure.

2. ELIMINATION OF TETRAHEDRA

The Delaunay mesh generation algorithm creates the convex hull of the set of vertices. Therefore, a special "shelling" procedure for the elimination of the tetrahedra that lie outside the domain is required.

These tetrahedra are removed using the following algorithm:

1. All the tetrahedra T whose centroids $c(T)$ are outside the domain, namely,

$$u(c) > 0$$

are removed.

Next the following sequence of operations is repeated while at least one tetrahedron is removed by these operations.

2. It is checked whether the tetrahedron can be removed; such a tetrahedron is called admissible for removal. If a tetrahedron has two or more boundary faces, then it is admissible. If it has only one boundary face, then the orthogonal projection of the opposite vertex on this face is examined. The feasibility condition in this case is that the barycentric coordinates μ_1, μ_2, μ_3 of the projection satisfy the condition $\mu_i > -1/10$.

For all admissible tetrahedra, operations 3, 4, and 5 are performed.

3. The tetrahedra T that lie slightly inside the domain for which

$$u(c(T)) > -k\varepsilon_b |\nabla u(c(T))|.$$

are removed. Here $k = k(T)$ is the local stretching coefficient for the acute wedge defined in (4) and ε_b is the given threshold.

4. The near boundary poorly shaped tetrahedra are removed.

Let $l_{\max}(T)$ be the result of maximization

$$l_{\max} = \max_{e \in \mathscr{E}(T)} \frac{|e|}{Mh(c(e))},$$

where $\mathscr{E}(T)$ is the set of edges of the tetrahedron T. The shape of T is considered to be poor if

$$H_{\min}(T) > \frac{1}{20} Mh(c(T)) l_{\max}.$$

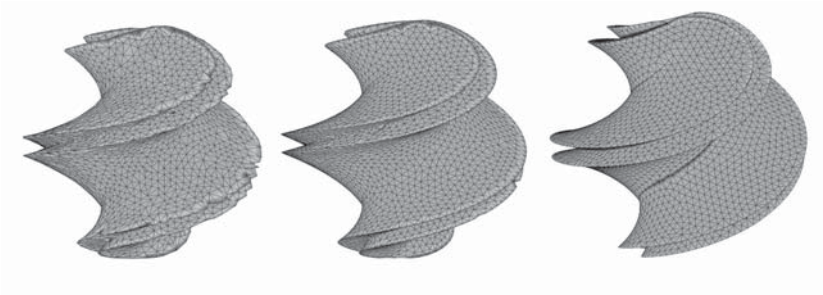

Fig. 8. Smooth and nonsmooth mesh vertices in the process of self-organization.

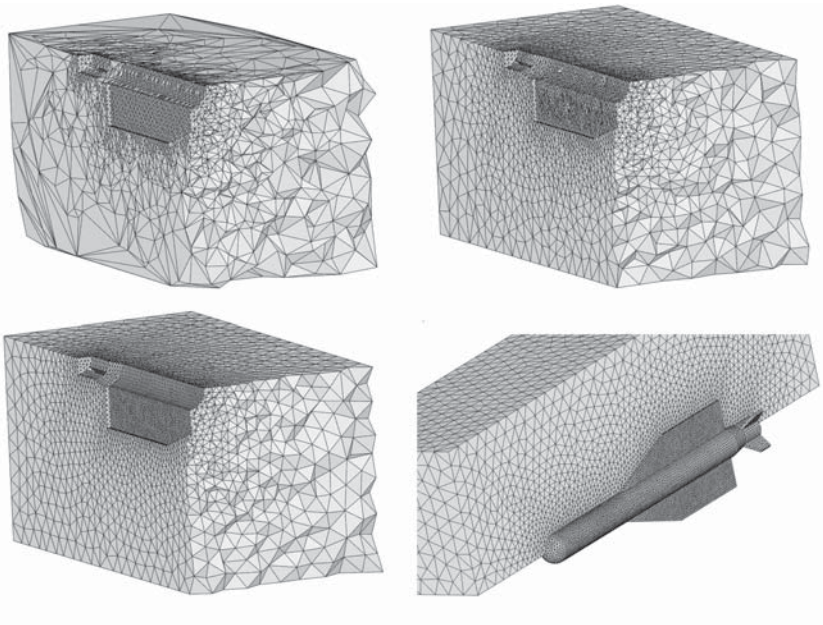

Fig. 9. The stages of the computational mesh generation: initial approximation, intermediate mesh, and the final mesh.

In this case, it is marked for removal. Here $H_{\min}(T)$ is the minimum height of the tetrahedron. If a tetrahedron is marked for removal but $k(T) < 1$, then one can apply to T the affine mapping (5). If the tetrahedron shape is still poor after the stretching, then it is removed.

5. Removing tetrahedra by the dihedral angle test.

We set the threshold for the dihedral angle $\beta = \pi/18$. If the tetrahedron T is inside an acute wedge, then the reduced threshold

$$\beta = \frac{1}{4}\max\left(\frac{1}{36}\pi, \alpha_c(T)\right),$$

is used, where $\alpha_c(T)$ is the size of the local acute wedge angle determined in (3). The tetrahedron is removed if

$$\alpha_{\min}(T) < \beta \quad \text{or} \quad \alpha_{\max}(T) > \pi - \beta,$$

where $\alpha_{\min}(T)$ and $\alpha_{\max}(T)$ are, respectively, the minimal and the maximal dihedral angles of the tetrahedron.

If certain tetrahedra were removed at stages 3–5, then the feasibility check is made again and the removal loop is repeated.

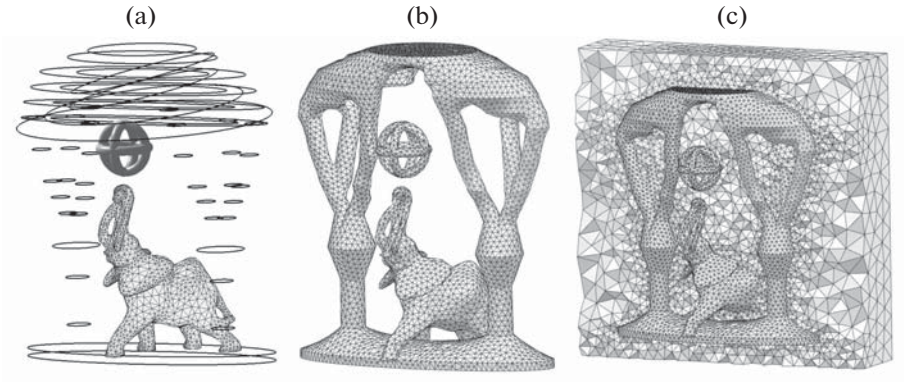

Fig. 10. Heterogeneous specification of the domain (a) and the Delaunay mesh ((b) the surface mesh and (c) the cut of the tetrahedral mesh).

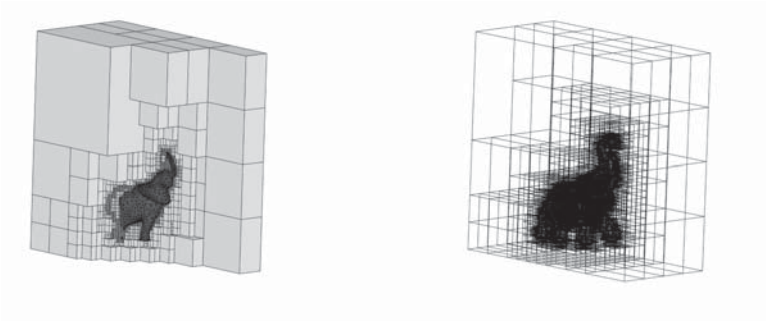

Fig. 11. Octree.

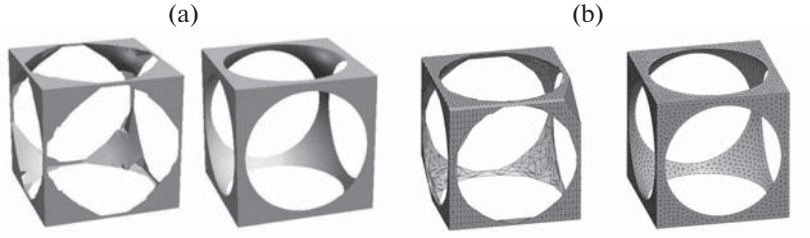

Fig. 12. Cube with a ball removed: (a) the model from [9], (b) the results obtained in this paper.

3. NUMERICAL RESULTS

Figure 8 illustrates the evolution of the set of smooth and nonsmooth boundary vertices in the process of the computational mesh construction. The values of the signed discrete Laplace–Beltrami operator $Bx(p_i)$ defined in (6) are highlighted in color. It is clear that the nonsmoothness areas are localized near sharp edges.

Figure 9 illustrates the evolution of the mesh and the approximation of the sharp edges for a body that is specified analytically.

Figure 10a illustrates the heterogeneous specification of the computational domain: the pavilion is determined by a set of planar sections, the elephant is described by a tessellation, and the openwork ball is defined analytically. The domain is assembled using Boolean operations.

Figure 11 shows the structure of an octree for the fast computation of an implicit function defined as the signed distance to the surface tessellation. A survey of techniques and algorithms for the construction of quadtrees and octrees can be found in [15, 16].

Fig. 13. The results from [9].

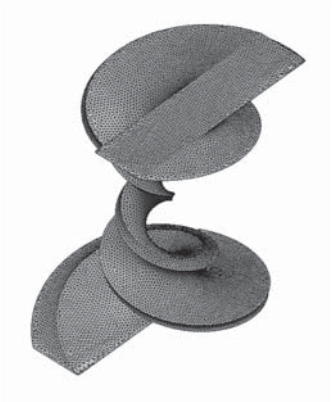

Fig. 14. The results obtained in this paper.

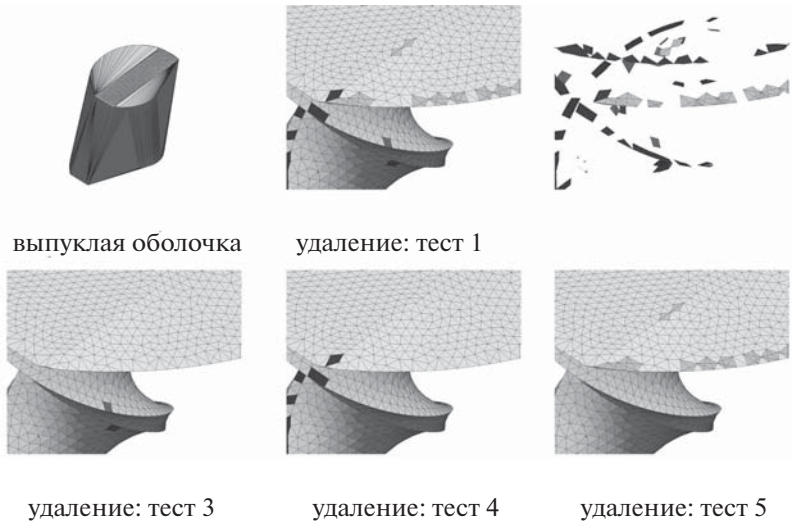

Fig. 15. Removal of tetrahedra.

In the next series of examples, we show that we solved all the tests described in [9]. Recall that in [9] only zero isosurfaces of the implicit functions were constructed, while in the present paper we construct a 3D Delaunay mesh. Since no analytical description of the models is available, we constructed their analogs. The first model is a cube with a ball removed from it (Fig. 12).

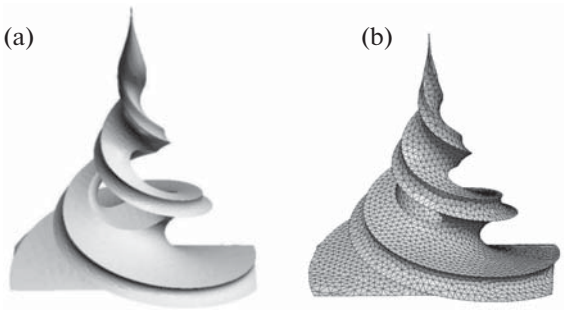

Fig. 16. Reconstruction of a surface from [9] (a), and the results obtained in this paper (b).

Fig. 17.

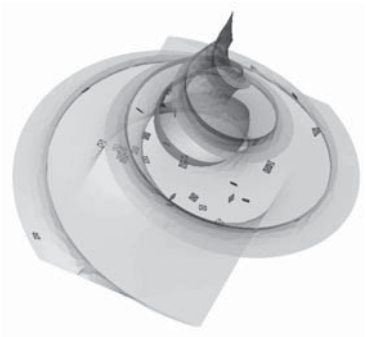

Fig. 18.

Note that the domain was not covered by the initial mesh. Due to repulsion forces and the update of the parameter M, the generated mesh filled the entire domain.

The next model is a twisted rectangular spiral (Figs. 13 and 14). It is a difficult test for the for mesh generation because it contains a boundary sharp edge with the dihedral angle close to $\pi/36$.

The removal of tetrahedra in this model is illustrated in Fig. 15, and Fig. 17 shows almost flat tetrahedra that remain in the mesh after the execution of the self-organization algorithm.

Figure 16 shows the twisted pyramid model, and Fig. 18 shows almost flat tetrahedra in the final 3D mesh. Figure 19 illustrates the results for the curvilinear icosahedron model.

After the mesh is generated, it can be optimized using the variational algorithm described in [17], which moves the mesh vertices without making topological changes to the mesh. The same optimization and untangling algorithm can be used to build a deformed 3D mesh after the boundary deformation as shown in Figs. 20 and 21.

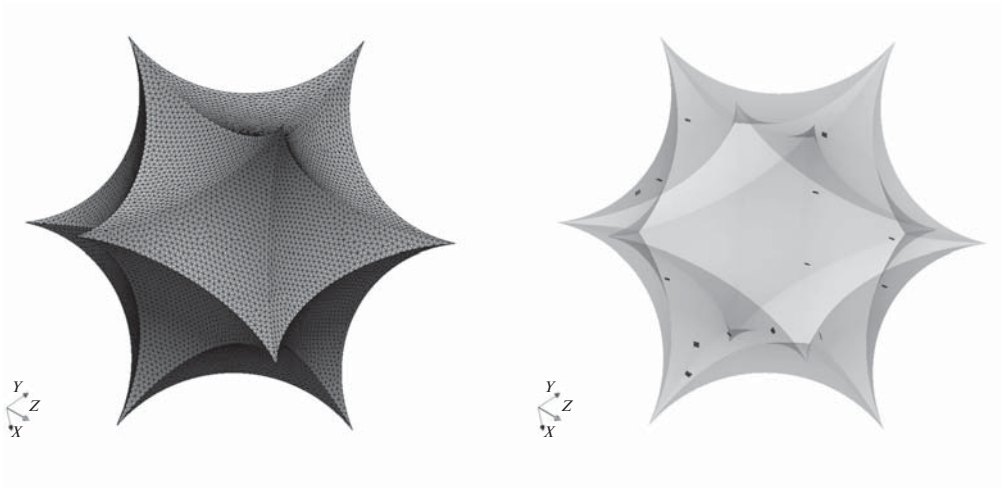

Fig. 19. A surface mesh and almost flat tetrahedra.

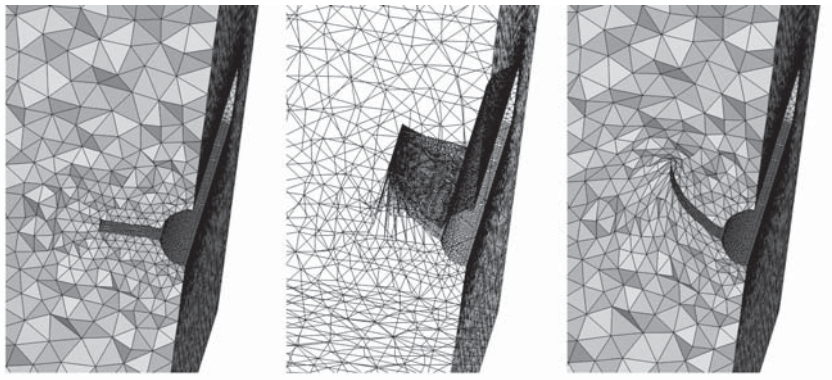

Fig. 20. Morphing of the model surface and the subsequent morphing of the mesh.

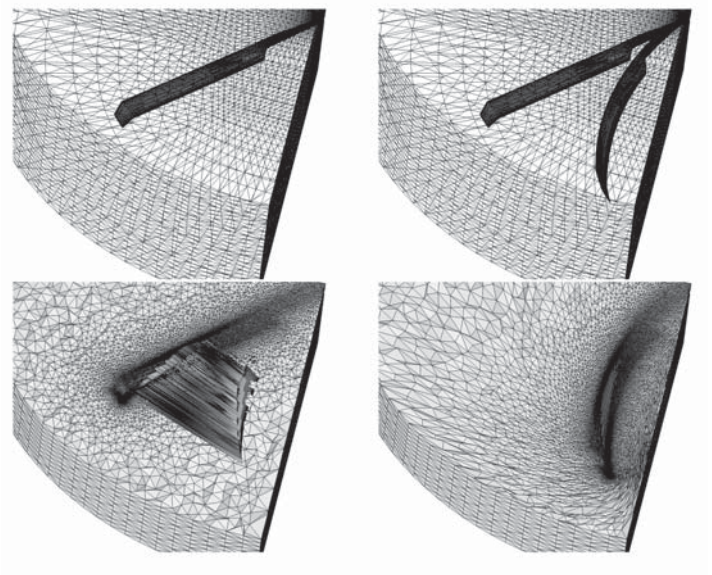

Fig. 21. Morphing of the model surface and the subsequent morphing of the mesh.

CONCLUSIONS

A variational algorithm for generating Delaunay meshes is complex shaped implicit domains that can stably reconstruct the sharp edge structure is proposed and tested.

ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research (project no. 14-07-008-05) and by the Russian Academy of Sciences (project I.33P).

REFERENCES

1. V. L. Rvachev, *The Theory of* R-*Functions and Its Applications* (Naukova dumka, Kiev, 1982) [in Russian].
2. F. Labelle and J. R. Shewchuk, "Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles." ACM Trans. Graphics **28** (3), 57:1–57:10 (2007).
3. J. -D. Boissonat, D. Cohen-Steiner, B. Mourrain, et al., "Meshing of surfaces," in *Effective Computational Geometry for Curves and Surfaces* (2007), pp. 181–230.
4. J. Tournois, C. Wormser, P. Alliez, and M. Desbrun, "Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation," ACM Trans. Graphics **28** (3), 75:1–75:9 (2009).
5. A. V. Kofanov and V. D. Liseikin, "Grid construction for discretely defined configurations," Comput. Math. Math. Phys. **53**, 759–765 (2013).
6. P.-O. Persson and G. Strang, "A simple mesh generator in MATLAB," SIAM Rev. **46**, 329–345 (2004).
7. L. N. Belousova and V. A. Garanzha, "Construction of Delaunay meshes in implicitly specified domains with a nonsmooth boundary," *Trudy 51 Nauchnoi Konferentsii Sovremennye Problemy Fundamental'nykh i Prikladnykh Nauk* (Proc. of the 51st Conf. on Problems of Fundamental and Applied Sciences), 2008, Vol. 2, pp. 98–101.
8. V. A. Garanzha and L. N. Kudryavtseva, "Generation of three-dimensional Delaunay meshes from weakly structured and inconsistent data," Comput. Math. Math. Phys. **52** (3), 427–447 (2012).
9. Y. Ohtake and A. Belyaev, "Dual/primal mesh optimization for polygonized implicit surfaces," *Proc. of the 7th ACM Symp on Solid Modeling and Applications (SMA'02)* (ACM, New York, 2002), pp. 171–178.
10. S. Schaeffer, T. Ju, and J. Warren, "Manifold dual contouring," IEEE Trans. Visualization Comput. Graphics **13**, 610–619 (2007).
11. J. L. Frey and P. L. George, *Mesh Generation: Applications to Finite Elements* (Hermes Science, Paris, 2000).
12. P. L. George and E. Saltel, ""Ultimate" robustness in meshing an arbitrary polyhedron," Int. J. Numer. Meth. Eng. **58**, 1061–1089 (2003).
13. A. A. Danilov, "Unstructured tetrahedral mesh generation technology," Comput. Math. Math. Phys. **50**, 139–156 (2010).
14. V. D. Liseikin, *Grid generation methods,* 2nd ed. (Springer, Berlin, 2010).
15. J. A. Orenstein, "Multidimensional tries used for associative data searching," Inform. Proc. Lett. **14**, 150–157 (1982).
16. H. Samet, *The Design and Analysis of Spatial Data Structures* (Addison-Wesley, Reading, 1990).
17. V. A. Garanzha, L. N. Kudryavtseva, and S. V. Utyzhnikov, "Untangling and optimization of spatial meshes," J. Comput. Appl. Math. **269**, 24–41 (2014).

Translated by A. Klimontovich

SPELL: 1. octree