Generation of Three-Dimensional Delaunay Meshes from Weakly Structured and Inconsistent Data

Article *in* Computational Mathematics and Mathematical Physics · March 2012

DOI: 10.1134/S0965542512030074

2 authors, including:

Vladimir A. Garanzha
Russian Academy of Sciences
34 PUBLICATIONS **108** CITATIONS

ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2012, Vol. 52, No. 3, pp. 427–447. © Pleiades Publishing, Ltd., 2012.
Original Russian Text © V.A. Garanzha, L.N. Kudryavtseva, 2012, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2012, Vol. 52, No. 3, pp. 499–520.

Generation of Three-Dimensional Delaunay Meshes from Weakly Structured and Inconsistent Data[1]

V. A. Garanzha and L. N. Kudryavtseva

Dorodnicyn Computing Center, Russian Academy of Sciences, ul. Vavilova 40, Moscow, 119333 Russia
e-mail: garan@ccas.ru

Received June 16, 2011

Abstract—A method is proposed for the generation of three-dimensional tetrahedral meshes from incomplete, weakly structured, and inconsistent data describing a geometric model. The method is based on the construction of a piecewise smooth scalar function defining the body so that its boundary is the zero isosurface of the function. Such implicit description of three-dimensional domains can be defined analytically or can be constructed from a cloud of points, a set of cross sections, or a "soup" of individual vertices, edges, and faces. By applying Boolean operations over domains, simple primitives can be combined with reconstruction results to produce complex geometric models without resorting to specialized software. Sharp edges and conical vertices on the domain boundary are reproduced automatically without using special algorithms. Refs. 42. Figs. 25.

DOI: 10.1134/S0965542512030074

Keywords: tetrahedral meshes, Delaunay triangulation, surface reconstruction, radial basis functions, variational method.

1. INTRODUCTION

Mesh generation plays important role in numerical simulation and arises in many fundamental problems and real-life applications. Currently methods for tetrahedral mesh generation in domains of complex geometry are fairly well developed (see [1–8]).

Software solutions and research codes in various applied fields rely on these methods. Before constructing mesh, one should build geometric model of the object under study. As a rule, the construction of such a model, for example, an airplane or an automobile, is based on geometric modeling and computer-aided design methods. Eventually, a geometric model is represented as a set of curvilinear faces defined by B-splines supplemented by the gluing rules. In biological problems and engineering analysis, a geometric model is frequently constructed using data of three-dimensional scanning, tomography, or electronic microscopy. In this case a preliminary stage of surface reconstruction is required, which produces a surface consisting of flat triangles (see, e.g., [9–13]).

As a result, geometric models may require so-called clean-up or "repair," i.e. correction of the geometric and topological errors in the definition of surfaces and three-dimensional bodies. Sophisticated algorithms and specialized software have been developed to address this problem.

As a rule, volume mesh generation is preceded by the construction of a triangular surface mesh. Next, a tetrahedral partition is constructed inside the resulting polyhedron. It is well known that a given set of boundary faces may be incompatible with Delaunay partition. Hence a number of heuristic algorithms have been proposed for tetrahedral mesh generation in polyhedral domains with prescribed boundary mesh. The least heuristic and most robust in practice is the algorithm from [7], which is used as a computational kernel in a number of mesh generation algorithms, including the one described in [14].

The purpose of this paper is to construct three-dimensional tetrahedral meshes directly from incomplete, weakly structured, and inconsistent data describing a geometric model without the reconstruction of the surface. The basic idea behind the proposed method is to find a scalar function defining the body such that its boundary is the zero isosurface of the function. Such implicit description of three-dimensional domains can be constructed analytically or using a cloud of points, a set of cross sections, or a "soup" of individual vertices, edges, and faces. By applying Boolean operations over domains, simple

[1] Partial support of FRBR grant 11-01-12136-ofi-m-2011, Presidium RAS grant P-18, Scientific School support grant NSH-5264.2012.1 is gratefully acknowledged.
The article was translated by the author.

primitives can be combined with reconstruction results to produce complex geometric models without resorting to specialized software. The basic difficulty which has to be resolved in the implementation of a mesh generation algorithm is that practically interesting implicit functions are not smooth and their zero isosurfaces contain sharp edges and conical vertices. Moreover, the boundary of resulting tetrahedral mesh should reproduce all the basic features of the domain boundary without compromising the quality of mesh cells.

For tetrahedral mesh generation in implicit domains, a number of effective algorithms have been proposed, which instead of the explicit definition of domain boundary rely on the fast procedure for computation of the intersection point of domain boundary with an arbitrary straight-line segment. In [15] a fast tetrahedral mesh generation algorithm with guaranteed mesh quality for domains with a smooth boundary was proposed. An algorithm for mesh generation in implicit domains with a piecewise smooth boundary was developed in [16, 17]; however, it required that sharp edges on the surface be defined in an explicit form.

The starting point for the development of the suggested algorithm was work [18] describing a simple algorithm for the generation of two- and three-dimensional Delaunay meshes in implicit domains implemented as a small Matlab code. The idea behind the algorithm from [18] is as follows: the edges of a mesh are interpreted as elastic struts with prescribed length distribution, and mesh generation is treated as the problem of achieving an equilibrium state of the mechanical system. The strut lengths are chosen so that they are in a compressed state for a quasi-equilibrium solution. The topology of the connections, i.e., the choice of points joined by the edges follows from the Delaunay empty ball principle. In [18] an implicit domain was defined using a signed distance function from the boundary. A remarkable property of this algorithm in the two-dimensional case is that the resulting triangular meshes are highly regular, even if the initial approximation is chaotic. Comparative analysis (see [19]) has shown that this algorithm is one of the best self-organization algorithms for two-dimensional meshes. Specifically, it is less sensitive to the initial approximation compared to Lloyd's self-organization algorithm (see [20]). Generalizations of this algorithm to three dimensions were addressed, for example, in [21, 22]. In the three-dimensional case, Lloyd's algorithm is known to face a number of difficulties. Searching for equilibrium states using Delaunay principle is not enough for the elimination of flat tetrahedra from the resulting mesh. Below, flat tetrahedra are referred to as "slivers." Thus, an additional stage is usually required to remove those tetrahedra.

In [19] a modification of the algorithm from [18] was proposed where the implicit function is not required to be a signed distance function, since the construction of such implicit functions by itself is a complicated problem of computational geometry. The resulting algorithm turned out to be very simple and undemanding with respect to input data. Moreover, in the two-dimensional case, empirical observations have shown that this algorithm, as a rule, automatically reproduces sharp angles on the boundary of the implicit domain without applying any additional techniques. However, this is not the case in 3D, and the algorithm from [18, 19] as it is may fail to reproduce sharp edges on the domain boundary. The solution of this problem is presented below. Preliminary results were published in [23].

2. ALGORITHM FOR TETRAHEDRAL MESH GENERATION IN IMPLICIT DOMAINS

Definition of an Implicit Domain

Consider a bounded domain $\Omega \subset \mathbb{R}^3$. Assume that the boundary of Ω is Lipschitz continuous and piecewise regular and the neighborhood of each boundary point in some coordinate system can be represented as the graph of a function defined as the difference of convex functions.

Let $u(x) : \mathbb{R}^3 \longrightarrow \mathbb{R}$ be a given function such that $u(x) < 0$ at interior points of the domain and $u(x) > 0$ in the complement of the domain (see Fig. 1).

Assume that $u(x)$ is a Lipschitz continuous piecewise smooth function that can be represented as the difference of convex functions and its derivatives along a nondegenerate vector field transversal to $\partial\Omega$ exist and do not vanish in a finite layer S near $\partial\Omega$. The layer is defined as follows: it is assumed that there exists a constant $r_0 > 0$ such that S contains the union of all balls of radius r_0 with centers lying on $\partial\Omega$. A domain defined by means of an implicit function is illustrated in Fig. 1.

Methods for constructing functions satisfying such conditions (at least empirically) are discussed in Section 5. In fact, we assume that the behavior of the implicit function qualitatively resembles the behavior of the signed distance function. Since the mesh generation method proposed in this paper is, in fact, heuristic, we do not present more rigorous conditions on the domain.

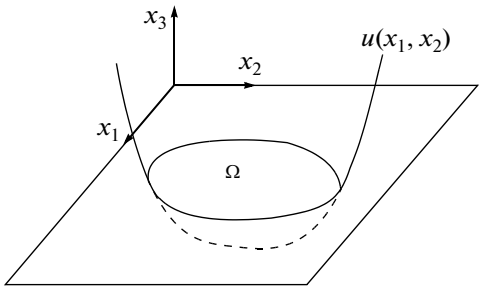

Fig. 1. Example of an implicitly defined two-dimensional domain Ω.

Tetrahedral Mesh in Implicit Domain

Let us discuss what kind of meshes are admissible as the output of a three-dimensional mesh generation algorithm. The three-dimensional domain Ω defined by the inequality $u(x) \le 0$ is approximated by a polyhedron Ω_h whose boundary $\partial\Omega_h$ consists of flat triangles glued along whole edges. Inside Ω_h, we construct a normal tetrahedral partition $\mathcal{T}$ such that the set of boundary triangles is a subset of faces of this partition. It is required that the surface Ω_h converges pointwise to $\partial\Omega_h$ as the edge length of the boundary mesh tends to zero. Moreover, the piecewise constant field ν_h of unit normals to $\partial\Omega_h$ is required to converge to the piecewise continuous field ν of unit normals to $\partial\Omega$. The parameter h here plays the role of a mesh cell size; i.e., it is assumed that the maximum edge length in $\mathcal{T}$ does not exceed Ch, where C is a constant.

Thus, we assume that there exists a homeomorphism $\psi_h : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ such that $\psi_h(\Omega_h) = \Omega$, $\psi_h(\partial\Omega_h) = \partial\Omega$, (a) $|p - \psi_h(p)| \longrightarrow 0$ when $h \longrightarrow 0$ for all $p \in \partial\Omega_h$, (b) $|\nu_h(p) - \nu(\psi_h(p))| \longrightarrow 0$ when $h \longrightarrow 0$ for all $p \in \Omega_h$ not belonging to the set of edges $\mathcal{T}$.

The most common approach to mesh generation is as follows. First, a triangular surface mesh approximating the boundary of Ω is constructed, and then a three-dimensional mesh defining a partition of the resulting polyhedron is generated. The algorithm we propose is based on a different approach. Specifically, a three-dimensional mesh in the domain and a boundary mesh are constructed simultaneously. For this purpose, we use a simple heuristic self-organization algorithm that distributes a set of points over domain according to a given distribution density. The distribution density is defined by a function $f_h(x) : \mathbb{R}^3 \longrightarrow \mathbb{R}$, $f_h(x) > 0$, which can be viewed as the target edge length at the point x.

According to [18], it is not always convenient to define $f_h(x)$ directly. In practice, the target edge length is defined using the function $f_h(x) = Mf(x)$, where $f(x)$ specifies the relative characteristic length, i.e., the ratio of the target edge lengths at different points of the computational domain, and the coefficient M depends on h and the volume of Ω. At each step, given a set of vertices, the algorithm constructs a tetrahedral mesh satisfying the Delaunay criterion (see [24–26]). The tetrahedra that according to certain criteria are recognized not to belong to the computational domain are eliminated from the mesh. These steps of the algorithm are repeated until a mesh is generated for which the measure of deviation of the polyhedral boundary from the zero isosurface does not exceed a certain threshold, while the tetrahedral mesh contains a small number of nearly flat tetrahedra. Next, we apply an optimization algorithm that minimizes some measure of mesh distortion without changing the mesh connectivity, thus improving the mesh and eliminating the nearly flat tetrahedra. Note that boundary mesh points can move along the zero isosurface. The optimization algorithm may lead to the violation of the Delaunay property.

Self-Organization Algorithm

The idea of a self-organization algorithm for a point system can be described as follows. Using a mechanical analogy, we model a hypothetical elastic medium, assuming that the mesh vertices are material points that experience repulsive forces. As a result, some points can be pushed outside Ω. These points are projected back onto the boundary of Ω. Thus, the projection onto the boundary can be interpreted as an impermeable barrier that prevents the points from moving outside the domain. As an illustration, we can use construction foam that expands and fills three-dimensional cavities, matching their boundaries. The algorithm produces a hypothetical "equilibrium" mesh in which all the forces acting on a material point are balanced. The repulsive forces are chosen so that the elastic medium in equilibrium is in a com-

pressed state. In the two-dimensional case, this simple idea was used in [18] to produce high-quality two-dimensional triangulations that were comparable in quality to those constructed with the advanced front method.

Consider a 3D point set $\mathscr{E} = \{p_1, p_2, \ldots, p_n\}$. Denote by $\mathscr{T}(\mathscr{E})$ a Delaunay tetrahedral partition from which a subset of tetrahedra violating certain criterion of affiliation to domain is removed. The union of all tetrahedra in $\mathscr{T}$ is a polyhedron Ω_h, which generally may not approximate Ω. The mesh $\mathscr{T}(\mathscr{E})$ is said to be equilibrium if it satisfies the condition

$$F(p_i) = 0, \quad 1 \le i \le n,$$

where $F(p_i)$ denotes the force acting on the point p_i. The force can be represented as

$$F(p_i) = U(w_e F_e(p_i) + w_{ee} F_{ee}(p_i) + w_{vf} F_{vf}(p_i)) + w_b F_b(p_i).$$

Here, F_e is the repulsive force directed along the edges of $\mathscr{T}$, F_{ee} is the force caused by the repulsion between the opposite edges in each tetrahedron in the mesh, and F_{vf} is the force caused by the repulsion between a tetrahedron vertex and the opposite face. The vector $F_b(p_i)$ denotes the "sharpening" boundary force, which is used to decrease the angle between the normal to a boundary face and the direction of the gradient of u. Here, $F_b(p_i) = 0$ if p_i is an interior mesh node. The operator U is defined as

$$U(q(p_i)) = \begin{cases} q, & p_i \text{ is an interior point of } \Omega_h, \\ q - \nabla u(p_i)\dfrac{q^{\mathrm{T}}\nabla u(p_i)}{|\nabla u(p_i)|^2}, & p_i \in \partial\Omega_h. \end{cases}$$

Thus we can say that U removes from q defined at the boundary node p_i the component orthogonal to the isosurface of u passing through p_i. Note that U is defined only at those points p_i where $\nabla u(p_i)$ take definite nonzero value. The conditions on u imply that at every point x of the isosurface $u(x) = 0$ a tangent cone $K(x)$ exists. If this cone is a plane, then the function is smooth at x and its gradient is uniquely determined. In the general case, the direction of the generalized gradient can be fixed using, for example, the ray joining the point x with the center of the ball that is the nearest to x among all the unit balls lying inside $K(x)$.

Below w_e, w_{ee}, w_{vf} and w_b are positive weights.

Repulsive forces are calculated as follows. First, consider repulsive forces acting along the edges. Let $F_e(p_i) = 0$ for all i. Consider the edge e_k with vertices p_i and p_j from the edge set of $\mathscr{T}$. Then the contribution of this edge to the forces acting on p_i and p_j can be written as a pseudocode

$$F_e(p_i) := F_e(p_i) + (p_i - p_j)a_e, \quad F_e(p_j) := F_e(p_j) + (p_j - p_i)a_e,$$

where

$$a_e = \max\left(\frac{L_0}{L} - 1, 0\right), \quad L = |p_i - p_j|, \quad L_0 = Mf\left(\frac{1}{2}(p_i + p_j)\right). \tag{1}$$

Summing the contributions of all the edges, we obtain the final expressions for F_e. It follows from (1) that the contribution to the repulsive force is made only by edges that are shorter than the target length. The target edge length L_0 is defined by the value of the sizing function at the center of the current edge.

The repulsive force between the opposite edges of a tetrahedron is computed as follows. Let $F_{ee}(p_i) = 0$ for all i. Let $T_k \in \mathscr{T}$ be a tetrahedron with its four vertices denoted by local indices $\{1, 2, 3, 4\}$, which correspond to global indices i_1, i_2, i_3, i_4. The force of repulsion of the edge $[p_1, p_3]$ from the edge $[p_2, p_4]$ is computed as illustrated in Fig. 2a.

Let n_e denote a vector with its initial point a lying on the edge $[p_2, p_4]$ and its terminal point b lying on the edge $[p_1, p_3]$ that is orthogonal to these edges. If there is no such vector, then we set $n_e = 0$. The given force makes the following contribution to F_{ee}:

$$F_{ee}(p_{i_1}) := F_{ee}(p_{i_1}) + (1 - \mu)n_e v_{ee}, \quad F_{ee}(p_{i_3}) := F_{ee}(p_{i_3}) + \mu n_e v_{ee},$$

$$F_{ee}(p_{i_2}) := F_{ee}(p_{i_2}) - (1 - \nu)n_e v_{ee}, \quad F_{ee}(p_{i_4}) := F_{ee}(p_{i_4}) - \nu n_e v_{ee},$$

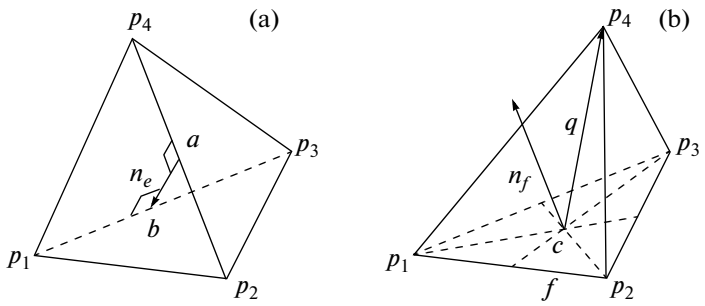

Fig. 2. Repulsive force (a) between opposite edges and (b) between a vertex and the opposite face.

where

$$\nu_{ee} = \max\left(\frac{L_0}{L} - 1, 0\right), \quad L = |a-b|, \quad L_0 = \frac{Mf(c)}{\sqrt{2}}, \quad \mu = \frac{|b-p_1|}{|p_3-p_1|}, \quad \nu = \frac{|a-p_2|}{|p_4-p_2|},$$

and $c = 1/4(p_1 + p_2 + p_3 + p_4)$ is the centroid of the tetrahedron. Summing the contributions made by all three pairs of opposite edges for each tetrahedron, we obtain F_{ee} at all mesh nodes.

Now consider the repulsive force between the face f with vertices p_1, p_2, p_3 and the opposite vertex p_4 (see Fig. 2b). The contribution of a face–vertex pair to the force F_{vf} can be written as

$$F_{vf}(p_{i_4}) := F_{vf}(p_{i_4}) + n_f|q|\,\nu_{vf}\sin^2\alpha,$$

$$F_{vf}(p_{i_m}) := F_{vf}(p_{i_m}) - \frac{1}{3}n_f|q|\,\nu_{vf}\sin^2\alpha, \quad m = 1, 2, 3,$$

where

$$\nu_{vf} = \max\left(\frac{L_0}{L} - 1, 0\right), \quad L = |q|, \quad q = p_4 - \frac{1}{3}(p_1+p_2+p_3), \quad L_0 = \sqrt{\frac{2}{3}}Mf(c).$$

Here, n_f denotes the inward unit normal to the face f, α is the angle between the vectors n_f and q, and c is the algebraic center of T_k. Summing the contributions of four vertex–face pairs for all tetrahedra yields the values F_{vf} at all mesh nodes.

The sharpening boundary force $F_b(p_i)$ at the boundary point p_i is defined by the formula (see [27])

$$F_b(p_i) = \frac{\sum\limits_{T\in \mathrm{star}(p_i)} \mathrm{area}(T)\nabla u(c)\dfrac{(c-p_i)^{\mathrm{T}}\nabla u(c)}{|\nabla u(c)|^2}}{\sum\limits_{T\in \mathrm{star}(p_i)} \mathrm{area}(T)}, \tag{2}$$

where $\mathrm{star}(p_i)$ denotes the set of boundary triangles of $\partial\Omega_h$ containing the vertex p_i, $\mathrm{area}(T)$ is area of the triangle T, and c is the centroid of the triangle T. According to Fig. 3a, if p_i does not lie on a "relatively sharp" edge of $\partial\Omega$, then the projection of the boundary force $F_b(p_i)$ onto the tangent plane to an isosurface of $u(x)$ is directed toward the edge. If p_i lies on the edge as shown in Fig. 3b, then the boundary force has nearly the same direction as the gradient u.

A quasi-equilibrium mesh is constructed as follows. Let a point set $\mathscr{E}^k$ consisting of n points p_i^k be given at the kth iteration. A step of the selforganization algorithm is formulated as follows:

(i) Construct a Delaunay tetrahedral partition $\tilde{\mathscr{T}}^k(\mathscr{E}^k)$ for $\mathscr{E}^k$ using the Quickhull algorithm (see [28]) built in Matlab.

(ii) Eliminate the tetrahedra $\tilde{\mathscr{T}}^k$ that violate the criterion of affiliation to domain. The resulting partition is denoted by $\mathscr{T}^k(\mathscr{E}^k)$.

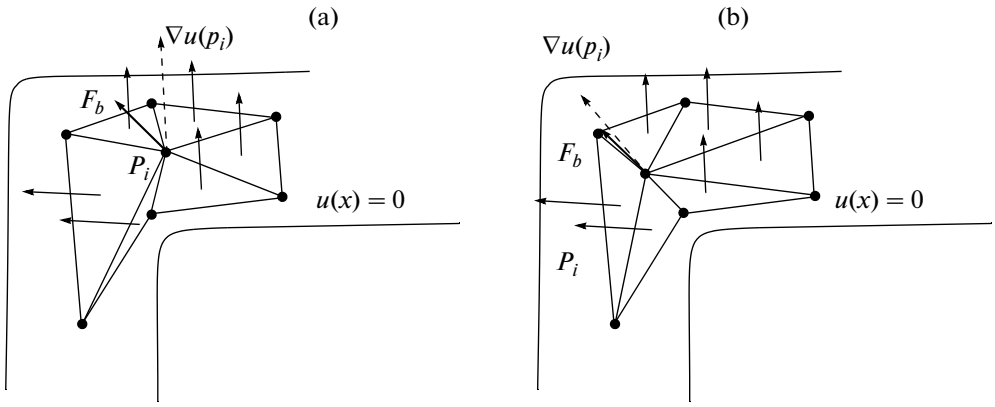

Fig. 3. Sharpening boundary force: (a) a vertex is attracted to a potential sharp edge and (b) a vertex is close to an equilibrium state.

(iii) Update the coordinates of the vertices of the point system affected by mechanical forces according to the following rule:

$$p_i^{k+1} = V(p_i^k + \tau_k F(p_i^k)), \tag{3}$$

where

$$\tau_k = \min\left(\frac{1}{2}, \frac{1}{2F_{\max}}\right), \quad F_{\max} = \max_i\left(\frac{|F(p_i^k)|}{Mf(p_i^k)}\right),$$

and V is a projector which is an identity operator for all interior mesh points and projects boundary vertices onto the surface $u(x) = 0$. The step τ_k in the relaxation process is chosen so that the maximum displacement in the course of relaxation does not exceed half the local characteristic length.

The forces acting on the mesh points are computed using the weighting coefficients $w_b = 5$, $w_v = 0.1$, $w_e = 0.1$ and $w_f = 0.1$.

The operator V projects a boundary vertex p onto the surface $u(x) = 0$ with the help of the simple iterative procedure

$$p^{m+1} = p^m - \tau_1 \frac{u(p^m)\nabla u(p^m)}{|\nabla u(p^m)|^2}. \tag{4}$$

Here, m denotes the local iteration number. If $u(x)$ is a linear function and $\tau_1 = 1$, then formula (4) defines the orthogonal projector onto the plane $u(x) = 0$. In the general case, iteration (4) is repeated until the distance of the point p^m from $\partial\Omega$ becomes less than a threshold ϵ_b. In practice, this is reduced to verifying the inequality

$$|u(p^m)| \le \epsilon_b |\nabla u(p^m)|, \tag{5}$$

where ϵ_b can be regarded as the error in the specified geometry. In [18] the threshold was specified as $\epsilon_b = h/10$. Analysis of the performance of the algorithm as applied to a number of test problems suggests the following conclusions. Due to the repulsive forces, the mesh points are eventually distributed over Ω according to the given relative local size function $f(x)$. The boundary points of $\mathcal{T}^k$ are distributed over the boundary of Ω approximately matching $f(x)$. The boundary faces of the mesh eventually approximate the smooth regions of $\partial\Omega$, while sharp edges manifest themselves on the polyhedral surface without using any special sharp edge detection algorithm. Tetrahedra with small or extremely large dihedral angles turn out to be unstable and break up.

Note that the volume repulsion mechanism may lead to degenerated tetrahedra. This property is very important, since optimal mesh generation involves continuous and discrete optimization. Continuous optimization is related to the search for the best positions of vertices. Discrete or topological optimization is related to the search for the best mesh connectivity. At present, the formal application of global optimi-

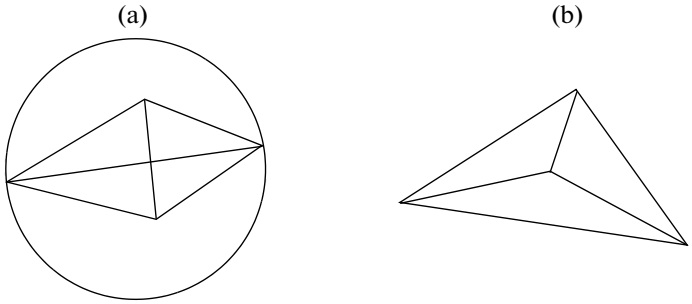

Fig. 4. (a) Sliver is a nonequilibrium tetrahedron; (b) the edge–edge and vertex-face forces are zero.

zation methods fails to produce effective mesh generation algorithms. The quality of meshes is optimized using a set of heuristic topological transformation procedures (see, e.g., [29, 30].

In fact, "loopholes" admitting mesh cell degeneration in repulsion and the subsequent connectivity change according to the Delaunay criterion can be considered as heuristic methods to jump out from local minima of a mesh quality objective function.

Figure 4a shows a flat tetrahedron (sliver), for which the size of the circumscribed sphere is relatively small. The repulsive forces are constructed in such a way that this tetrahedron is a nonequilibrium one. Figure 4b demonstrates another type of tetrahedron degeneration, when a tetrahedron vertex coincides with the centroid of the opposite face. For such a tetrahedron, the edge–edge and vertex-face forces are close to zero. Thus, tetrahedra of this type can serve as loopholes allowing for a connectivity change without a noticeable increase in the amplitude of the forces acting on the vertices.

Based on a formula of type (2), an edge-sharpening algorithm for surface triangulations approximating implicit surfaces was proposed in [27]. In fact, the development of an effective algorithm based on the interaction between a boundary edge sharpening algorithm and volume mesh generation is a major new practical result of this paper.

Iterations (i)–(iii) can be terminated only if Ω is sufficiently accurately approximated by Ω_h. Moreover, the following conditions should hold:

(i) The Euler characteristics of Ω and Ω_h coincide.

(ii) Condition (5) of proximity to $\partial\Omega$ holds for boundary vertices of $\mathcal{T}^k$.

(iii) For all boundary faces of $\mathcal{T}^k$, the following condition holds (meaning the proximity of normals to exact values):

$$n_f^{\mathrm{T}}\nabla u(c) \geq (1 - C_b\epsilon_b)|n_f||\nabla u(c)|, \tag{6}$$

where n_f is the outward normal to a face and c is the centroid of the face.

(iv) The sliver-type flat tetrahedra are few, and there are no almost degenerate tetrahedra of other types.

Note that the declared goal of the algorithm—the generation of an equilibrium mesh—is frequently not achieved in practice. Geometric oscillations may arise, when small displacements of vertices change the connectivity of the mesh leading to the formation of a new tetrahedron (sliver), being strongly nonequilibrium. For this reason, a small amplitude of the forces acting on all the mesh vertices is ineffective as a termination criterion.

In fact, conditions (5) and (6) are empirical, since they are not actual proximity measures for $\partial\Omega$ and $\partial\Omega_h$. It is easy to formulate formal conditions under which (5) and (6) become proximity measures. In fact, they are reduced to the requirement that the current mesh be fine enough to resolve the fine details of the isosurface $u(x) = 0$. From a practical point of view, we will use only conditions (5) and (6).

After iterative process (i)–(iii) is completed, variational mesh optimization (see [31]) is used without changing the connectivity of the mesh.

3. ELIMINATION OF TETRAHEDRA

Consider in more detail a procedure for eliminating from $\tilde{\mathcal{T}}^k$ the tetrahedra that violate the criterion of affiliation to domain. This procedure reminds the removal of redundant material by a sculptor, so it is

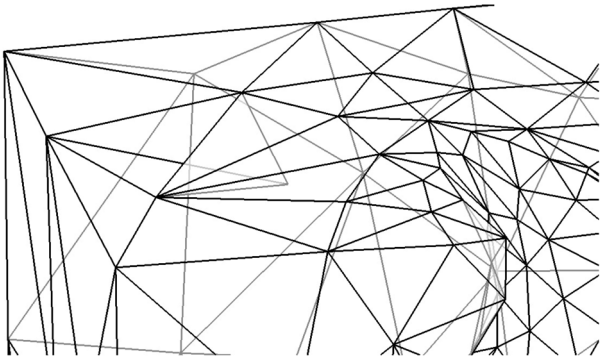

Fig. 5. Elimination of a boundary tetrahedron has led to the formation of a pocket.

sometimes called sculpting. Specifically, the tetrahedra which are guaranteed to lie outside Ω are removed first. Then we consecutively eliminate tetrahedra having some undesirable properties and lying near the domain boundary; i.e., $u(x)$ changes the sign on them, but their intersection with Ω is fairly "shallow." After eliminating such tetrahedra, the vertices and the normals to the faces of the polyhedral boundary are expected to be close to the corresponding exact values (conditions (5) and (6)). However, it is easy to construct examples of point sets and domains of simple shape with a piecewise smooth boundary such that, for some regions on the boundary of Ω, there are no Delaunay faces satisfying proximity conditions (6). Thus, whatever criteria are used for tetrahedra elimination, the resulting polyhedron looks "jagged." As a rule, it is this effect that is observed at the first iterations of the algorithm. However, these situations must be eliminated in the search for optimal vertex locations. Nevertheless, the tetrahedra elimination algorithm has to be able to construct acceptable polyhedral bodies at initial iterations as well. The following set of heuristic rules is used for this purpose.

I. Eliminate tetrahedra whose centroids satisfy the condition $u(c) > 0$.

II. For all tetrahedra with at least one boundary face, for each vertex, consider the segment joining the vertex with the center of the face in circle. If $u(x)$ is positive at a point x of the segment, then this vertex is marked as poor. If there is a vertex marked three times or two vertices marked twice, then the corresponding tetrahedron is announced to be a "bridge" between areas of the boundary of Ω and is eliminated.

III. Find a set of tetrahedra $\mathcal{T}_b$ that can be eliminated without the formation of "pockets." A pocket is characterized by the presence of a boundary mesh vertex p such that $u(p) < -\epsilon_b|\nabla u(p)|$, while the geometrically nearest point on the domain boundary is separated from p by a tetrahedron. An example of such a pocket is shown in Fig. 5. Eliminate external tetrahedra lying slightly deeper in Ω according to the criterion $u(c) > -\epsilon_b|\nabla u(c)|$.

IV. From the set $\mathcal{T}_b$, eliminate only those tetrahedra in which all the edges satisfy

$$\frac{u(c)}{|\nabla u(c)|} > -\epsilon_b Mf(c)\max\left(\frac{L}{L_0}\right),$$

where L is the edge length, $L_0 = Mf(c_l)$, c is the centroid of a tetrahedron, and c_l is the midpoint of an edge.

V. Eliminate almost flat near boundary tetrahedra where the minimal dihedral angle is smaller than 10° or the maximal dihedral angle is larger than 170° and

$$u(a) > -0.25\,Mf(a)|\nabla u(a)|,$$

where a is the centroid of the tetrahedron or its arbitrary vertex.

At several last iterations near a quasi-equilibrium state, the value of the proximity threshold ϵ_b is halved. Steps II–IV are repeated until the set of tetrahedra to be eliminated is empty.

Specification of an Initial Vertex Distribution

The simplest method for specifying an initial vertex distribution is as follows. Define a set $\mathcal{C}$ of the centers of a cubic lattice with the spacing h. The points satisfying the inequality $u(p) \geq -2\epsilon_b|\nabla u(p)|$ are removed from the set. Then $\mathcal{C}$ is decimated by removing points with a probability proportional to the value

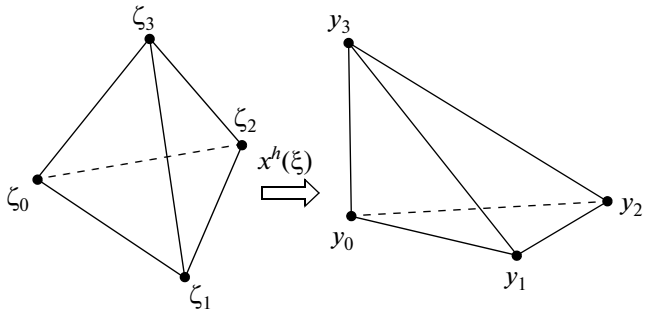

Fig. 6. Target tetrahedron T_k^t and the tetrahedron T_k of the mesh $\mathcal{T}$.

of $Mf(x)$ at these points (see [18]). As a result, the initial point distribution is roughly consistent with the sizing function. A more effective algorithm based on the construction of an octree of Ω was proposed, for example, in [15]. An overview of the theory and algorithms for constructing quadtrees and octrees can be found in [32, 33].

Note that one can define random initial vertex distribution. Such an initial guess is convenient for testing the algorithm, since one of the goals of its development was to make the final mesh weakly sensitive to the initial vertex distribution. Numerical experiments have confirmed this behavior of the algorithm, but as expected, mesh generation with a poor initial guess required much more iterations of vertex displacement and mesh reconnection.

4. VARIATIONAL MESH OPTIMIZATION

Stages (i)–(iii) in the self-organization algorithm are aimed at the reconstruction of the domain boundary, the determination of a quasi-optimal mesh connectivity, and the distribution of points over the domain according to a given sizing function. The quality of the final mesh is improved via an additional optimization stage based on vertex relocation without changing the mesh connectivity.

The shape and volume of mesh cells are optimized as described in [31]. Consider a polyhedral domain Ω_h with a tetrahedral partition $\mathcal{T}$ consisting of n_s tetrahedra with n_v vertices, of which n_b are boundary ones. The vertices of $\mathcal{T}$ are denoted by z_i, $i = 1, \ldots, n_v$, Z is $3 \times n_v$ matrix, with columns z_i. On each tetrahedron $T_k \in \mathcal{T}$ the values y_0, y_1, y_2 and y_3 are the vertices with the local numbering. For each tetrahedron of this partition, we choose an equilateral tetrahedron as a the target one. The vertices of the target tetrahedron T_k^t are denoted by ζ_0, ζ_1, ζ_2 and ζ_3, as shown in Fig. 6.

Define the matrices $H = (\zeta_1 - \zeta_0 \zeta_2 - \zeta_0 \zeta_3 - \zeta_0)$ and $Q = (y_1 - y_0 y_2 - y_0 y_3 - y_0)$. The columns of these matrices are assumed to make up a right triple in $\mathbb{R}^3$; i.e., $\det H > 0$ and $\det Q > 0$. The volumes of the tetrahedra are given by

$$\operatorname{vol} T_k^t = \frac{1}{6}\det H, \quad \operatorname{vol} T_k = \frac{1}{6}\det Q.$$

The Jacobian matrix $S = \nabla_\xi x^h$ of the affine map $x^h(\xi) : T_k^t \longrightarrow T_k$ is written as $S = QH^{-1}$.

We search for a mapping that minimizes the following functional, which can be viewed as a mesh distortion measure:

$$F(z_1, \ldots, z_{n_v}) = \sum_{k=1}^{n_s} \varphi(QH^{-1})\big|_{T_k^t} \operatorname{vol} T_k^t = \frac{1}{6}\sum_{k=1}^{n_s} \varphi(QH^{-1})\det H\big|_{T_k^t},$$

where

$$\varphi(S) = \theta\mu(S) + (1-\theta)\nu(S). \tag{7}$$

Thus, φ is the sum of the shape distortion measure

$$\nu(S) = \frac{1}{3}\frac{\mathrm{tr}S^{\mathrm{T}}S}{(\det S)^{2/3}}$$

and the volume distortion measure

$$\mu(S) = \frac{1}{2}\left(\frac{v_0}{\det S} + \frac{\det S}{v_0}\right).$$

Here, $\theta = 0.8$ and v_0 is the ratio of the volume of the target tetrahedron to the volume of a tetrahedron with unit edges: $v_0 = (Mf(c))^3$, where $Mf(c)$ is the value of the characteristic length function at the centroid of T_k.

Two types of boundary conditions are considered. The first is the Dirich-let boundary condition, for which the boundary vertex z_k is fixed. The second is the slip boundary condition, for which the point z_k can move along the surface

$$u(x) = 0.$$

during mesh optimization. It is assumed that vector $\nabla u(z_k)$ takes a definite value. If u is not differentiable at z_k in the classical sense, then the gradient can be approximately computed as described above by using the tangent cone to $\partial\Omega$ at this point. Thus, at the point z_k, we can calculate the vectors l_1 and l_2 tangent to the boundary of Ω and assume that $l_i^{\mathrm{T}}\nabla u(x_k) = 0$.

Then the stationarity condition for the functional at z_k can be written as

$$l_i^{\mathrm{T}}\frac{\partial F}{\partial z_k} = 0, \quad i = 1, 2, \tag{8}$$

$$u(z_k) = 0. \tag{9}$$

This system consists of three equations, which correspond to three variables constituting the vector z_k. Let δz_k denote the increment at z_k. Linearizing Eq. (9) gives the following equation for δz_k:

$$\delta z_k^{\mathrm{T}}\nabla u(z_k) + u(z_k) = 0.$$

Thus, if z_k lies on the boundary of Ω, then $u(z_k) = 0$ and δz_k can be represented as a linear combination of the vectors l_i:

$$\delta z_k = \beta_1 l_1 + \beta_2 l_2, \tag{10}$$

where β_i are arbitrary coefficients. In other words, equality (10) means that boundary nodes can move only along the tangent plane to the boundary.

Assume that z_k does not belong to the boundary of the domain. Denote by V the projector onto the boundary described in (4).

The gradient R of $F(z_1, \ldots, z_n)$ is composed of the three-dimensional vectors $r_k = \dfrac{\partial F}{\partial z_k}$. The Hessian H of F is made up of 3×3 matrices $H_{ij} = \dfrac{\partial^2 F}{\partial z_i \partial z_j^{\mathrm{T}}}$, where H_{ij} is placed at the intersection of the ith block row and the jth block column.

The Newton–Raphson method for finding a stationary point of a mesh functional without taking into account slip condition can be written as

$$\sum_{j=1}^{n_v} H_{ij}(Z^l)\delta z_j + r_i(Z^l) = 0, \tag{11}$$

$$z_k^{l+1} = z_k^l + \tau_l \delta z_k, \quad k = 1, \ldots, n_v. \tag{12}$$

Let L_k denote a 3×3 matrix with the first two columns being the vectors l_i calculated at the point z_k and with the last column equal to zero. If $k \notin K_s$, then we set $L_k = I$. Here K_s denotes the set indices of internal vertices.

To include the slip condition in iterative scheme (11), (12), we multiply (11) by L_i^{T} from the left and take into account the fact that δz_j satisfies (10) for $j \in K_s$; i.e.,

$$\delta z_j = L_j \begin{pmatrix} \alpha_j \\ 0 \end{pmatrix},$$

so that the two-dimensional vector α_j can be used in linear system (11) as an unknown vector instead of δz_j. Let $\delta\tilde{z}_j$ denote the increment vector equal to δz_j for $j \notin K_s$ and to $(\alpha_j^{\mathrm{T}}, 0)^{\mathrm{T}}$ for $j \in K_s$, so that $\delta z_j = L_j \delta\tilde{z}_j$. With the notation introduced, the iterative method for finding a stationary point of $F(Z)$ can be written as

$$\sum_{j=1}^{n_v} L_i^{\mathrm{T}} H_{ij} L_j \delta\tilde{z}_j + L_i^{\mathrm{T}} r_i(Z^l) = 0, \tag{13}$$

$$z_k^{l+1} = V(z_k^l + \tau_l L_k \delta\tilde{z}_k), \quad k = 1, \ldots, n_v. \tag{14}$$

Equality (14) can be rearranged into

$$Z^{l+1} = V(Z^l + \tau_l \delta\tilde{Z}),$$

where the projector V onto the domain boundary coincides with that defined by formula (3). The relaxation parameter τ_l is determined by approximately solving the one-dimensional minimization problem

$$\tau_l = \arg\min_{\tau} F(V(Z^l + \tau\delta\tilde{Z})).$$

For this purpose, we use the Armijo scheme (see [34]) or the simplest bisection method.

To obtain a method similar to the Ivanenko–Charakhch'yan iterative barrier method (see [35]), we set $H_{ij} = 0$ for $i \neq j$ in general formulas (13). Here, $\delta\tilde{z}_i$ are determined by solving independent linear systems of dimension 2 at sliding points and of dimension 3 at the remaining mesh points. To obtain the implicit method of [31] from (13), all the off-diagonal terms in the matrices $L_i^{\mathrm{T}} H_{ij} L_j$ are suppressed. In this case, linear system (13) is split into three independent linear systems for the vectors $\delta\tilde{Z}_m$ being the rows of the matrix $\delta\tilde{Z}$, which are related to $\delta\tilde{z}_i$ by the equalities

$$(\delta\tilde{Z}_m)_i = (\delta\tilde{z}_i)_m.$$

The variational method can also be used when the algebraic volume of some tetrahedra in the initial mesh is equal to zero or negative. In the presence of such negative tetrahedra, an efficient approach is to use the mesh untangling method proposed in [36]. The idea behind this method is that the determinant $\det S$ in the denominator of (7) is replaced by

$$\chi(\det S) = \frac{1}{2}(\det S + \sqrt{\varepsilon^2 + \det S^2}).$$

The mesh is corrected using continuation technique with respect to ε from large values to zero.

5. DEFINITION OF AN IMPLICIT FUNCTION

Construction of an Implicit Function Using Boolean Operations

Implicit functions defining geometric primitives are fairly easy to construct. Several simple examples are given below. For example, a sphere is defined by the function

$$u(x) = (x_1^2 + x_2^2 + x_3^2)^{\frac{1}{2}} - 1,$$

a cube is defined by the function

$$u(x) = \max(|x_1| - 1, |x_2| - 1, |x_3| - 1),$$

and a finite circular cylinder is defined by the function

$$u(x) = \max\left((x_1^2 + x_2^2)^{\frac{1}{2}} - R, |x_3| - 1\right).$$

Complicated domains can be constructed by consecutively using Boolean operations of union, intersection, and difference on simpler domains. Let $u_1(x)$ and $u_2(x)$ be implicit functions defining domains Ω_1 and Ω_2. Then, depending on the definition of Ω_3, the function $u_3(x)$ is given by

$$\Omega_3 = \Omega_1 \cup \Omega_2 \Rightarrow u_3(x) = \min(u_1(x), u_2(x)),$$

$$\Omega_3 = \Omega_1 \cap \Omega_2 \Rightarrow u_3(x) = \max(u_1(x), u_2(x)),$$

$$\Omega_3 = \Omega_1 \backslash \Omega_2 \Rightarrow u_3(x) = \max(u_1(x), -u_2(x)).$$

If the functions u_1 and u_2 and the domain Ω_3 are correctly defined, then u_3 will be correct as well.

Construction of an Implicit Function from a Set of Contours in Parallel Planar Cross Sections

Quite popular method for constructing implicit functions from incomplete data is based on the use of radial basis functions (RBF) (see [37–42]). We describe this method as applied to a simple two-dimensional example.

Let a function ϕ be given on the segment $[ab] = \{x : x = a + (b-a)t, 0 \le t \le 1\}$. The problem is to construct a function $u_\epsilon(x)$ defined for all $x \in \mathbb{R}^2$ and being close to ϕ on $[ab]$. The computational formulas are very simple:

$$u_\epsilon(x) = \frac{\sum_{i=0}^{n} w_\epsilon(|x - p_i|)\phi(p_i)}{\sum_{i=0}^{n} w_\epsilon(|x - p_i|)}, \quad p_i = a + i(b-a)/n,$$

where the weight function w_ϵ is defined, for example, as

$$w_\epsilon(r) = \frac{1}{(\epsilon^2 + r^2)^2}.$$

If n is sufficiently small, then $u_\epsilon(x)$ can oscillate in the vicinity of $[ab]$. To suppress these oscillations, n has to be chosen large enough for the distance between p_i and p_{i+1} be less than ϵ. Since the interpolant $u_0(x)$ is obtained in the limit $\epsilon \longrightarrow 0$, we see that the construction of a quality approximation requires a very detailed auxiliary extra segment partition. In what follows, the corresponding set of points p_i is referred to as the set of quadrature points.

The RBF method can be used to construct an approximant that approximates not only a function but also its gradient on a given segment. For this purpose, we define the function of two arguments

$$\tilde{\phi}(x, p_i) = \phi(p_i) + \nu^{\mathrm{T}}(x - q),$$

where q is an arbitrary interior point of the segment $[ab]$. Define the approximant

$$u_\epsilon(x) = \frac{\sum_{i=0}^{n} w_\epsilon(|x - p_i|)\tilde{\phi}(x, p_i)}{\sum_{i=0}^{n} w_\epsilon(|x - p_i|)}.$$

As $\epsilon \longrightarrow 0$, the function u_ϵ and its gradient at a quadrature point p_i converge to $\phi(p_i)$ and ν, respectively. To ensure their convergence at all points of the segment, the distance between adjacent quadrature points must decrease faster than ϵ.

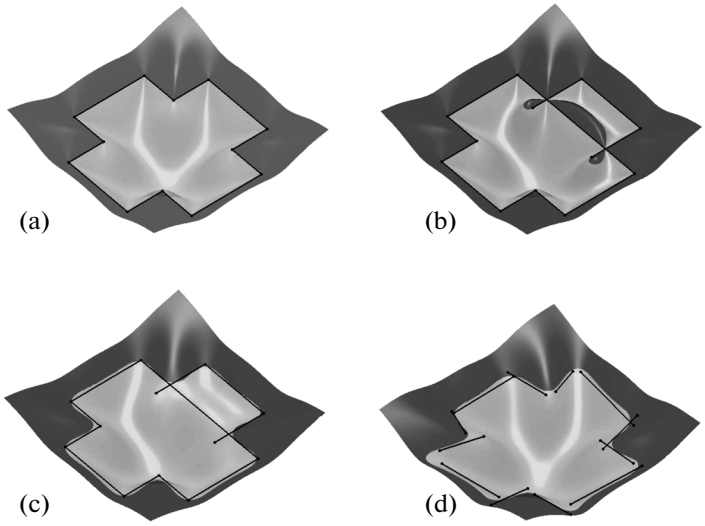

Fig. 7. (a) Exactly defined contour with an interpolating function constructed. (b) Self-intersecting contour. The exact interpolant forms undesirable loops. (c) Approximant constructed from a self-intersecting contour. The deviation of the function zero from the contour is insignificant. (d) Noisy input data. The approximant smoothes out the discontinuities and self-intersections.

In [37] it was suggested to use the analytical representation of approximant in the limit as $n \longrightarrow +\infty$. To find this analytical representation, it is assumed that as $n \longrightarrow +\infty$, the following relations hold

$$\frac{1}{n}\sum_{i=0}^{n} w_\epsilon(|x-p_i|) \longrightarrow \int_0^1 w_\epsilon(|x-a-t(b-a)|)dt = \Theta(x), \tag{15}$$

$$\frac{1}{n}\sum_{i=0}^{n} w_\epsilon(|x-p_i|)\phi(x,p_i) \longrightarrow \int_0^1 w_\epsilon(|x-a-t(b-a)|)\phi(x,a+t(b-a))dt = \theta(x). \tag{16}$$

Thus, we have the representation

$$u_\epsilon(x) = \frac{\theta(x)}{\Theta(x)}.$$

Assume now that we want to construct an approximant for a set of segments (edges) S_k, $k = 1, \ldots, K$. An approximant is sought in the form

$$u_\epsilon(x) = \frac{\sum_k l_k\theta_k(x)}{\sum_k l_k\Theta_k(x)},$$

where l_k is the length of S_k and, in view of (15) and (16), $\theta_k(x)$ and $\Theta_k(x)$ have the form

$$\theta_k(x) = \int_0^1 \frac{(\phi(b_k)-\phi(a_k))t+\phi(a_k)+(x-q_k)^{\mathrm{T}}n_k}{(|(b_k-a_k)t+a_k-x|^2+\varepsilon^2)^2}dt,$$

$$\Theta_k(x) = \int_0^1 \frac{1}{(|(b_k-a_k)t+a_k-x|^2+\varepsilon^2)^2}dt.$$

Here, n_k is the unit normal to an oriented edge which defines the target value of the implicit function gradient. The weighting factors l_k ensure that the quadrature nodes are uniformly distributed over the edges irrespective of their lengths, and this condition must hold during passage to the limit.

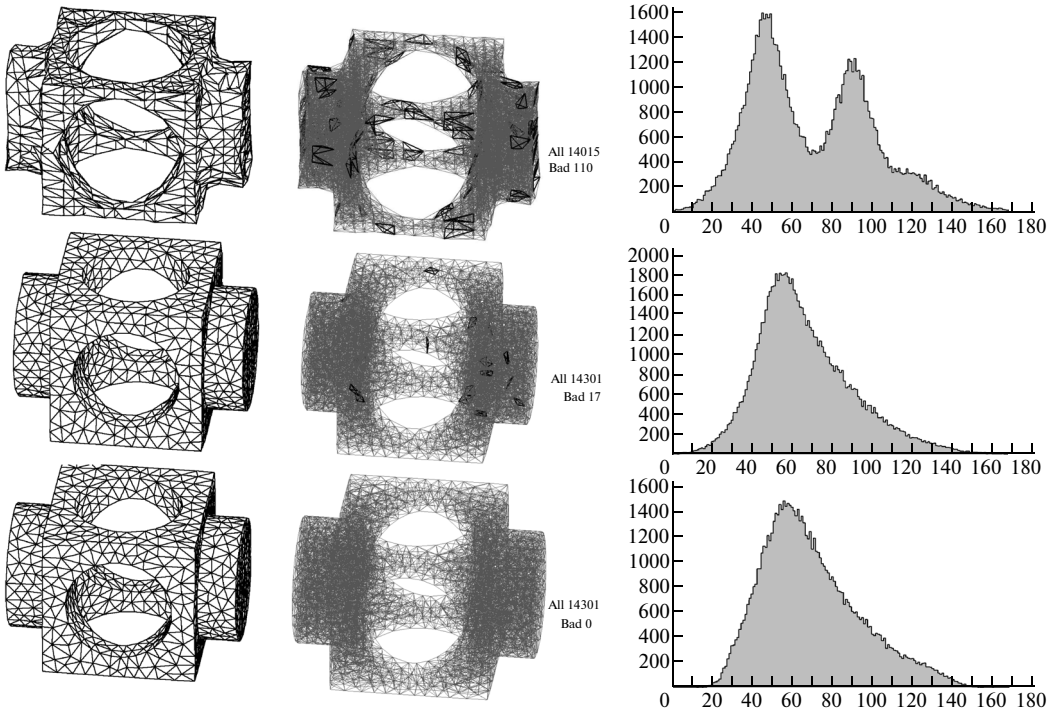

Fig. 8. Successive stages of algorithm.

Let $q_k = (a_k + b_k)/2$. The functions $\theta_k(x)$ and $\Theta_k(x)$ can be written analytically as

$$\theta_k(x) = BI_1 + CI_2, \quad \Theta_k(x) = \frac{1}{|b_k - a_k|^4} I_2,$$

where

$$I_2 = \int_0^1 \frac{1}{[(t+K_1)^2 + K_2]^2} dt = \frac{1}{2K_2\sqrt{K_2}} \left(\frac{\xi}{\xi^2 + 1} + \arctan(\xi) \right) \Bigg|_{\frac{K_1}{\sqrt{K_2}}}^{\frac{1+K_1}{\sqrt{K_2}}},$$

$$I_1 = \int_0^1 \frac{t}{[(t+K_1)^2 + K_2]^2} dt = \int_0^1 \frac{t+K_1}{[(t+K_1)^2 + K_2]^2} dt - K_1 I_2 = -\frac{1}{2K_2(\xi^2+1)} \Bigg|_{\frac{K_1}{\sqrt{K_2}}}^{\frac{1+K_1}{\sqrt{K_2}}} - K_1 I_2,$$

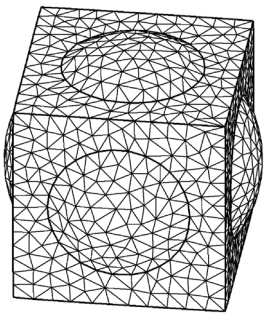

Fig. 9. Domain defined as the union of a cube and a ball.

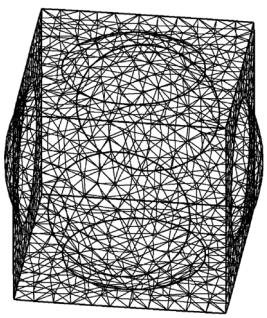

Fig. 10. Minimal dihedral angle is equal to 10.6° and a maximal dihedral angle is equal to 165.3°.

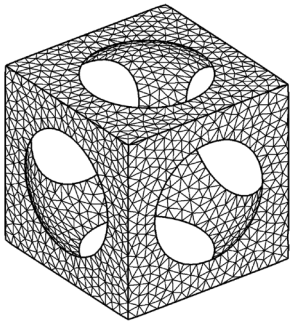

Fig. 11. Domain defined as a cube with extracted ball.

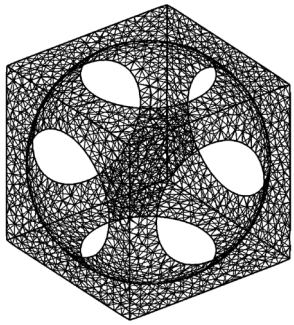

Fig. 12. Minimal dihedral angle is equal to 11.9°, and a maximal dihedral angle is equal to 159.4°.

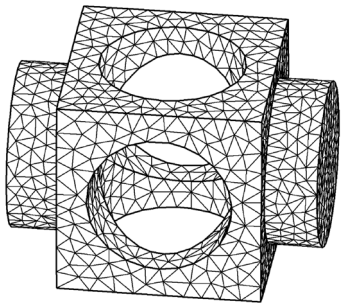

Fig. 13. Domain defined by applying Boolean operations over a cube, ball, and cylinder.

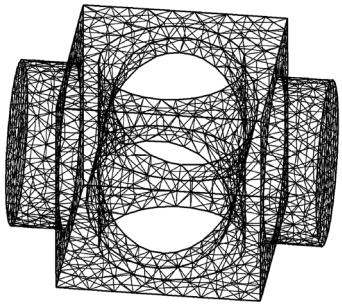

Fig. 14. Minimal dihedral angle is equal to 12.5°, and a maximal dihedral angle is equal to 159.4°.

and the coefficients A, B, K_1 and K_2 are given by the formulas

$$B = \frac{\phi(b_k) - \phi(a_k)}{|b_k - a_k|^4}, \quad C = \frac{\phi(a_k) + n_k^{\mathrm{T}}(x - q_k)}{|b_k - a_k|^4},$$

$$K_1 = \frac{(b_k - a_k)^{\mathrm{T}}(a_k - x)}{|b_k - a_k|^2}, \quad K_2 = \frac{|a_k - x|^2 + \epsilon^2}{|b_k - a_k|^2} - K_1^2.$$

In fact, ϵ serves as a smoothing parameter, namely, the function u_ϵ is almost insensitive to details of the contour whose size is less than ϵ.

Let us use the above idea to construct an implicit function from a set of contours. Consider the two-dimensional case first. Assume that an oriented contour is approximately defined by a set of segments. The goal is to construct a function of two variables $u_\epsilon(x)$ with a zero level curve approximating this contour so that $u_\epsilon(x) < 0$ to the left of the contour and $u_\epsilon(x) > 0$ to its right when the contour is traversed counterclockwise. In the case of RBF, the function $\tilde{\phi}$ is set to zero on each segment, while its gradient is equal to the value of the outward unit normal to this segment.

When an approximant is constructed according to this scheme, the zero isoline generally intersects the contour. To guarantee that the vertices on the contour lie inside the domain whose boundary is defined by the isosurface of the function, in [37] it was suggested to iteratively update the values of ϕ_k at the vertices. This approach was not implemented in this paper, since, for small ϵ, the shape of the domain is affected insignificantly. To reduce the approximation error for the boundary of the domain, the mean value of the function on the boundary is subtracted from its computed value.

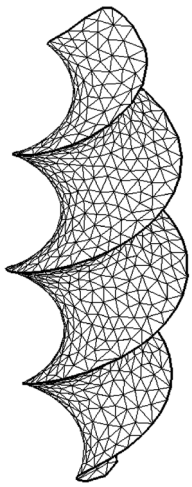

Fig. 15. Domain defined as the intersection of two ellipsoids with subsequent twisting.

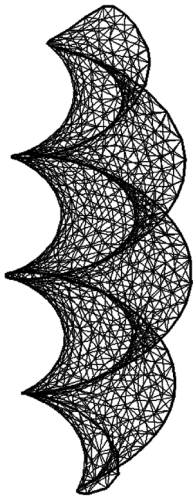

Fig. 16. Minimal dihedral angle is equal to 12.5°, and a maximal dihedral angle is equal to 159.2°.

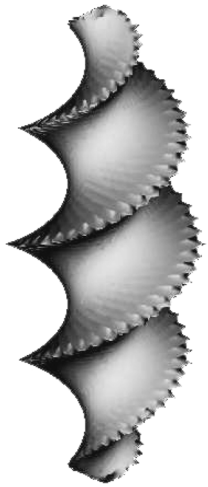

Fig. 17. Isosurface reconstruction by marching cubes.

Figure 7 illustrates the construction of interpolating and approximating functions. This is necessary to verify that the approximation results on these examples coincide with those of [37]. In Figs. 7a and 7b, an interpolant is constructed from a given contour (ϵ is negligibly small). Since the weight function has a singularity at zero, the evaluation of the integrals may face difficulty at $\epsilon = 0$. The lighter areas correspond to $u_\epsilon(x) < 0$. On the contour with a self-intersection, the zero level curve of the exact interpolant forms a loop. In Figs. 7c and 7d, approximating functions are constructed for $\epsilon = 0.2$ and the contour diameter $D = 2$. Nevertheless, after the refinement stage, the zero level of the function nearly coincides with the outer edges of the contour. The perturbation to the edges is introduced in such a way that both self-intersections and discontinuities are observed on the contour. In this case, the algorithm for constructing an approximating function performs well, smoothing the discontinuities according to the parameter ϵ.

If the contour γ_i is defined in a planar cross section $x_3 = h_i$ of three-dimensional Euclidean space, where the sequence of h_i is strictly monotonically increasing, then the function in each cross section can be constructed using the above RBF algorithm. As a result, we obtain a set of two-dimensional functions $u_i(x_1, x_2) = u(x_1, x_2, h_i)$, which are used to reconstruct the three-dimensional function $u(x)$, for example, by applying the simplest linear interpolation along a vector field when $h_i < x_3 < h_{i+1}$.

6. NUMERICAL EXPERIMENTS

In the first group of numerical experiments, meshes were constructed for domains of relatively simple shape assembled from geometric primitives. Figure 8 illustrates the successive stages of algorithm from top to bottom: (a) the initial Delaunay mesh roughly approximates the domain and contains large number of slivers; (b) an intermediate Delaunay mesh approximates the implicit domain fairly well, but still contains a few slivers; and (c) the mesh after optimization does not contain slivers.

The histograms on the right in Fig. 8 show the distribution of dihedral angles of all tetrahedra in the mesh.

Figures 9–16, 18–20 present surface meshes. The reproduced sharp edges of the boundary are marked with bold lines.

Figures 15–17 serve to compare the domain boundary reconstructions produced by the proposed algorithm and by the marching cube algorithm (see [9]). It is clearly seen that the latter algorithm fails to reconstruct sharp edges.

Figure 20 shows meshes reproducing complicated structure of intersecting sharp edges, while Fig. 21 illustrates application of the suggested method for engineering mesh generation problem.

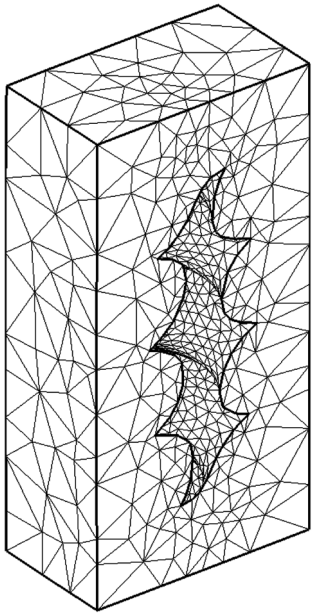

Fig. 18. Domain is defined as the complement of the spiral shown in Fig. 15.

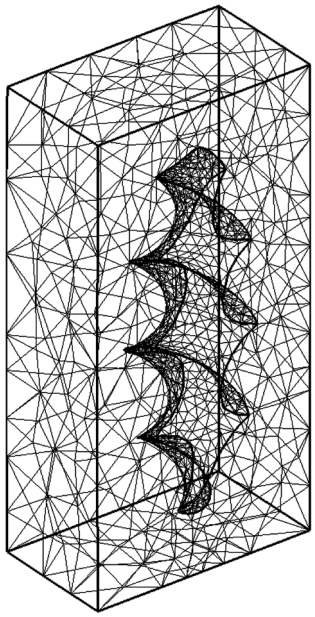

Fig. 19. Minimal dihedral angle is equal to 12.5°, and a maximal dihedral angle is equal to 159.4°.

Figure 22 gives an example of reconstructing a three-dimensional implicit function from a set of planar cross sections and presents the resulting three-dimensional mesh. Contours are shown on the left, while the boundary faces of a three-dimensional mesh constructed in the domain defined by the resulting implicit function are depicted in the center and on the right. The boundary mesh is also displayed on the right. Note that Boolean operations were also used to define the domain.

The resulting domain may depend on the prescribed vector field along which the approximation is performed. A set of contours is presented in Fig. 23 on the left. The domain obtained by approximation along a vertical vector field is shown in the center. This domain contains tunnels and bridges. In the case depicted on the right, the vector field was chosen in such a way that the resulting domain consisted of two disconnected components. The choice of a suitable vector field along which an implicit function interpo-

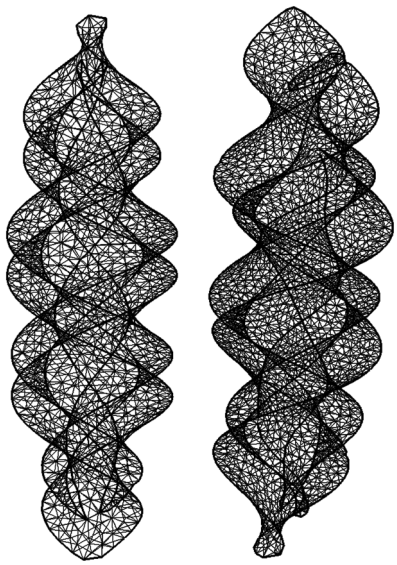

Fig. 20. Recovery of a complex structure of sharp edges on an implicit surface.

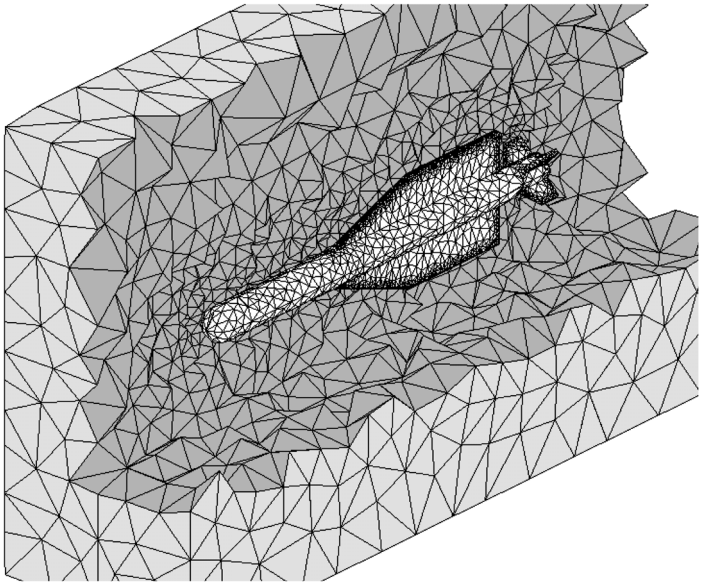

Fig. 21. Three-dimensional mesh generation for an engineering model.

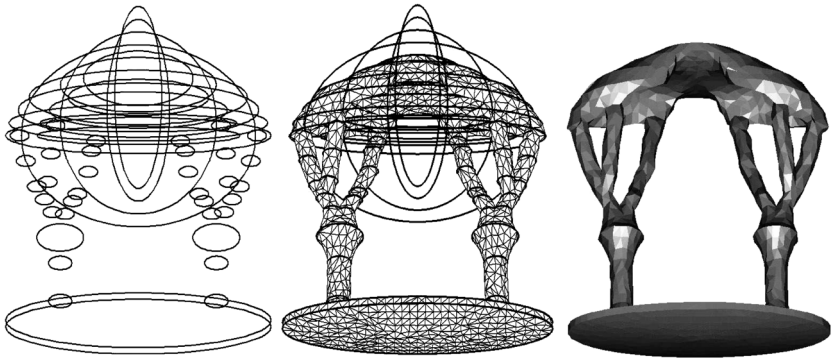

Fig. 22. Three-dimensional mesh generation from a set of cross sections.

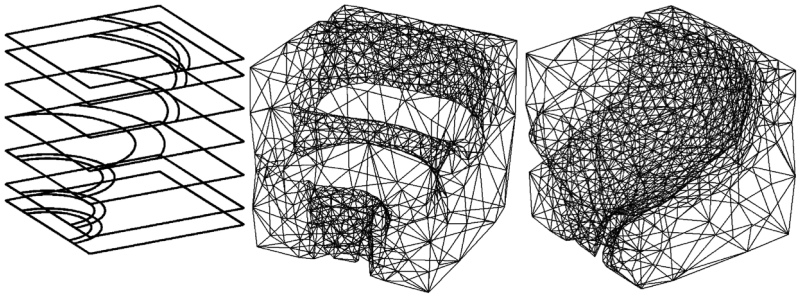

Fig. 23. A set of cross sections and reconstructions for different vector fields.

lant is constructed depends on additional assumptions about the shape of the domain and on the application field.

Figure 24 displays a set of six contours defined with errors. Their top views are shown on the left. One can see that the contours contain considerable jumps and self-intersections.

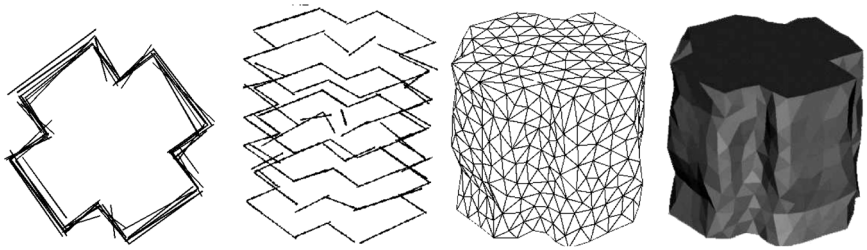

Fig. 24. Contours defined with errors, and three-dimensional mesh.

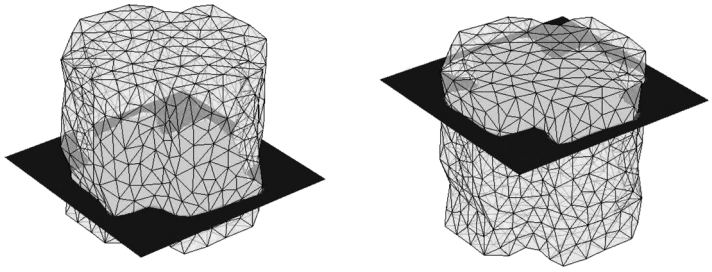

Fig. 25. Reconstruction from perturbed data.

Figure 25 displays a boundary mesh for the domain defined by the function constructed from such input data. Approximating functions are defined by the smoothing parameter $\epsilon = 0.1$. The figure demonstrates the behavior of the approximating functions corresponding to the fifth and second contours. The subdomains corresponding to negative and positive values of the approximating functions are shown by light and dark grey, respectively.

7. CONCLUSIONS

An algorithm was developed that automatically constructs three-dimensional Delaunay meshes in implicit domains with a nonsmooth boundary. Sharp edges and conical points on the boundary are reconstructed automatically without solving the problem of sharp edge detection on an isosurface. An algorithm for mesh cell optimization that eliminates almost degenerate tetrahedra and an algorithm for constructing an implicit function and a three-dimensional mesh directly from a set of contours were implemented, and their performance for noisy and inconsistent data were demonstrated. The overall algorithm was implemented as Matlab code, hence estimation of its performance on large-scale meshes and its comparative efficiency still requires more efficient realization.

ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research (project nos. 09-01-12106-ofi_m, 11-01-12136-ofi-m-2011), by the Program "Leading Scientific Schools" (project no. NSh-4096.2010.1), and by the Federal Targeted Program "Scientific and Pedagogical Staff for Innovative Russia" for 2009–2013 (state contract no. NK 408(3)).

REFERENCES

1. J. L. Prey and P. L. George, *Mesh Generation: Applications to Finite Elements* (Hermes, Paris, 2000).
2. G. F. Carey, *Computational Meshes: Generation, Adaptation, and Solution Strategies* (Taylor and Francis, Levittown, Pennsylvania, 1997).
3. S.-H. Teng, C. W. Wong, and D. T. Lee, "Unstructured Mesh Generation: Theory, Practice, and Perspectives," Int. J. Comput. Geom. Appl. **10**, 227–266 (2000).

4. V. D. Liseikin, *Mesh Generation Methods* (Springer-Verlag, Berlin, 2010).
5. N. N. Medvedev, *Voronoi–Delaunay Method in the Study of Noncrystalline System Structure* (Sib. Otd. Ross. Akad. Nauk, Novosibirsk, 2000) [in Russian].
6. H. Edelsbrunner, *Geometry and Topology for Mesh Generation* (Cambridge Univ. Press, New York, 2001).
7. P. L. George and E. Saltel, "Ultimate Robustness in Meshing an Arbitrary Polyhedron," Int. Numer. Meth. Eng. **58**, 1061–1089 (2003).
8. D. L. Marcum and N. P. Weatherill, "Unstructured Mesh Generation Using Iterative Point Insertion and Local Reconnection," AIAA J. **33**, 1619–1625 (1995).
9. W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," Comput. Graphics **21** (4), 163–170 (1987).
10. M. Levoy and K. Pulli, et al., "The Digital Michelangelo Project: 3D Scanning of Large Statues," *Proceedings of SIGGRAPH 2000* (New Orleans, Louisiana, USA, 2000), pp. 131–144.
11. H. Blum, *A Transformation for Extracting New Descriptors of Shape, Models Perception of Speech and Visual Form* (MIT, Cambridge, MA, 1967).
12. N. Amenta and M. Bern, "Surface Reconstruction by Voronoi Filtering," Discrete Comput. Geom. **22**, 481–504 (1999).
13. N. Amenta, S. Choi, and N. Leekha, "A Simple Algorithm for Homeomorphic Surface Reconstruction," *Proceedings of the 16th Annual ACM Symposium on Computational Geometry* (2000), pp. 213–222.
14. A. A. Danilov, "Unstructured Tetrahedral Mesh Generation Technology," Comput. Math. Math. Phys. **50**, 139–156 (2010).
15. F. Labelle and J. R. Shewchuk, "Isosurface Stuffing: Fast Tetrahedral Meshes with Good Dihedral Angles," ACM Trans. Graphics: Special Issue on the Proc. of ACM SIGGRAPH 2007 **26** (3), 57 (2007).
16. J.-D. Boissonat, D. Cohen-Steiner, B. Mourrain, et al., "Meshing of Surfaces," *Effective Computational Geometry for Curves and Surfaces* (Springer-Verlag, Berlin, 2007), pp. 181–230.
17. 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).
18. P.-O. Persson and G. Strang, "A Simple Mesh Generator in MATLAB," SIAM Rev. **46**, 329–345 (2004).
19. L. N. Belousova and V. A. Garanzha, "Comparison of Global Mesh Optimization Algorithms," *Proceedings of 14th International Baikal Workshop on Optimization Methods and Applications* (Inst. Syst. Energ. Melentieva Sib. Otd. Ross. Akad. Nauk, Irkutsk, 2008), Vol. 3, pp. 21–26.
20. S. P. Lloyd, "Least Squares Quantization in PCM," IEEE Trans. Inf. Theory **28** (2), 129–137 (1982).
21. Q. Du and D. Wang, "Tetrahedral Mesh Generation and Optimization Based on Centroidal Voronoi Tessellations," Int. J. Numer. Meth. Eng. **56**, 1355–1373 (2003).
22. P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun, "Variational Tetrahedral Meshing," ACM Trans. Graphics **24**, 617–625 (2005).
23. L. N. Belousova and V. A. Garanzha, "Delaunay Mesh Generation in Implicitly Defined Domains with Nonsmooth Boundaries," *Proceedings of the 51st Scientific Conference on Modern Problems in Fundamental and Applied Sciences*, Part 7, Upr. Prikl. Mat. **2**, 98–101 (2008).
24. G. F. Voronoi, "Nouveles applications des parametres continus a la theorie de formes quadratiques," J. Heine Angew. Math. **134**, 198–287 (1908).
25. G. F. Voronoi, "Study of Primitive Parallelohedra," in *Collected Works* (Akad. Nauk Ukr. SSR, Kiev, 1952), Vol. 2, pp. 239–368 [in Russian].
26. B. N. Delaunay, "On the Emptiness of the Sphere," Izv. Akad. Nauk SSSR, No. 4, 793–800 (1934).
27. Y. Ohtake, A. Belyaev, and A. Pasko, "Dynamic Mesh Optimization for Polygonized Implicit Surfaces with Sharp Features," Visual Computer. **19** (2) (2003).
28. C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," ACM Trans. Math. Software **22**, 469–483 (1996).
29. B. Joe, "Construction of Three-Dimensional Improved-Quality Triangulations Using Local Transformations," SIAM J. Sci. Comput. **16**, 1292–1307 (1995).
30. L. Freitag and C. Ollivier-Gooch, "A Comparison of Tetrahedral Mesh Improvement Techniques," *Proceedings of the 5th International Meshing Roundtable* (Pittsburgh, 1996), pp. 87–100.
31. V. A. Garanzha, "Barrier Method for Quasi-Isometric Mesh Generation," Comput. Math. Math. Phys. **40**, 1617–1637 (2000).
32. J. A. Orenstein, "Multidimensional Tries Used for Associative Data Searching," Inf. Proc. Lett. **14** (4), 150–157 (1982).

33. H. Samet, *The Design and Analysis of Spatial Data Structures* (Addison-Wesley, Reading, 1990).
34. L. Armijo, "Minimization of Functions Having Lipschitz Continuous First Partial Derivatives," Pac. J. Math. **16** (1), 1–3 (1966).
35. S. A. Ivanenko and A. A. Charakhch'yan, "Curvilinear Meshes of Convex Quadrilaterals," USSR Comput. Math. Math. Phys. **28**, 503–514 (1988).
36. V. A. Garanzha and I. E. Kaporin, "Regularization of the Barrier Variational Method of Mesh Generation," Comput. Math. Math. Phys. **39**, 1426–1440 (1999).
37. Ñ. Shen, J. F. O'Brien, and J. R. Shewchuk, "Interpolating and Approximating Implicit Surfaces from Polygon Soup," *Proceedings of the ACM SIGGRAPH Los Angeles, CA, August, 2004* (ACM, 2004), pp. 8–12.
38. E. J. Kansa, "Multiquadrics—A Scattered Data Approximation Scheme with Applications to Computational Fluid-Dynamics: II. Solutions to Parabolic, Hyperbolic and Elliptic Partial Differential Equations," Comput. Math. Appl. **19** (8–9), 147–161 (1990).
39. E. J. Kansa and R. C. Carlson, "Radial Basis Functions: A Class of Mesh-Free, Scattered Data Approximations," Comput. Fluid Dyn. J. **3** (4), 479 (1995).
40. M. J. D. Powell, "Radial Basis Functions for Multivariable Interpolation: A Review," in *Numerical Analysis* (Longman Scientific and Technical, Harlow, UK, 1987), pp. 223–241.
41. R. K. Beatson and W. A. Light, "Fast Evaluation of Radial Basis Functions: Methods for Two-Dimensional Polyharmonic Splines," IMA J. Numer. Anal. **17**, 343–372 (1997).
42. Y. Ohtake, A. Belyaev, M. Alexa, et al., "Multilevel Partition of Unity Implicits," ACM Trans. Graphics **22**, 463–470 (2003).