Chapter 1
Moving deforming mesh generation based on quasi-isometric functional ⋆

Vladimir Garanzha[1,2] and Liudmila Kudryavtseva[1,2,3]

Abstract We suggest algorithm which allows to generate moving adaptive mesh with fixed topology using time-dependent control metric in computational domain. Each cell of ideal mesh at a given time after local transformation into uniform coordinates related to metric should be congruent to the same cell at initial time level. The quasi-isometric functional [9] is used to measure and control the matching between real and ideal mesh. We have found that when global large deformations of initial mesh are necessary in order to satisfy mesh compression criteria inside thin moving layers, simple explicit methods of mesh optimization fail to follow metric precisely. Unfortunately, preconditioned gradient search algorithm for quasi-isometric functional is quite expensive, so coupling it directly to flow solver is highly inefficient. Another problem is that due to nonlinearity of the Euler-Lagrange equations we cannot assume that norm of residual of the functional is reduced to zero at each time step. Thus time continuity of the moving mesh is not guaranteed since iterative minimization of functional with changed metric may result in considerable displacements even for infinitely small time steps and space-time trajectories of mesh cells may become extremely inclined. In this work we suggest very simple and efficient algorithm which is based on direct mesh interpolation. The idea of gradient search algorithm [8] is based on the predictor, which constructs minimization direction (displacement field) for each mesh vertex for large time increment. Every intermediate mesh computed via this displacement is guaranteed to be correct. We have found that the length of the computed displacement field is very large compared to displacements allowed by flow solver, hence number of costly implicit minimizations can be sharply reduced. Assuming that time dependence of metric is smooth function, we obtain mesh deformation/adaptation algorithm which requires one linear solve per, say, 5-10 time steps, making it quite efficient component of the moving mesh

Dorodnicyn Computing Center FRC CSC RAS, Moscow 119333, Russia e-mail: garan@ccas.ru http://www.ccas.ru/gridgen/lab · Moscow Institute of Physics and Technology, Moscow, Russia · Keldysh Institute of Applied Mathematics RAS, Moscow, Russia
e-mail: liukudr@yandex.ru

⋆ ***

flow solver. We have found IC2-based linear solver [Kaporin] to be very efficient tool, especially for parallel version of algorithm. Experience with simpler algorithms, such as Conjugate Gradients algorithm IC(k) preconditioner, was quite dissapointing. We describe methods for constructing control metric allowing to implement immersed boundary conditions for a system of moving bodies defined by distance-like implicit function. Wrong specification of metric field leads to appearance of zones with sharp cell size jumps and highly skewed cells which resemble the ruptures of deformed hyperelastic material. Implicit function defines normal and tangent directions with respect to boundary, hence we specify not only normal stretching law (distribution of first eigenvalue λ_1 of metric tensor), but also distribution of "tangential" stretching law ($\lambda_{2,3}$ depending on problem dimension). Anisotropy of metric tensor is maximal near boundary and eventually reduces to zero away from the body. Near each boundary we define the layer of maximal normal stretching, and large time step of mesh predictor is computed using thickness of this layer and a expected number of cells inside layer in the normal direction. 2d and 3d results are presented.

1.1 Quasi-isometric functional

Description of quasi-isometric functional was published yearlier [9], [2]. Below we explain how it is applied for controlled mesh deformation.

Let $\xi_1,\xi_2,\ldots,\xi_d$ denote the Lagrangian coordinates associated with elastic material, and $x_1,x_2,\ldots,x_d$ denote the Eulerian coordinates of a material point. Spatial mapping $x(\xi,t):\ \mathbb{R}^d\to\mathbb{R}^d$ depending on parameter t defines time-dependent elastic deformation. We associate coordinates ξ_i with domain with initial mesh, namely, they are frozen into cell of initial mesh, while Eulerian coordinates are fixed Cartesian coordinates in the computational domain.

The Jacobian matrix of the mapping $x(\xi,\cdot)$ is denoted by C, where $c_{ij}=\partial x_i/\partial\xi_j$.

Let $G_\xi(\xi,t)$ and $G_x(x,t)$ denote the metric tensors defining linear elements and length of curves in Lagrangian and Eulerian coordinates in the domains Ω_ξ and Ω_x, respectively. Then, $x(\xi,t)$ is the mapping between metric manifolds M_ξ and M_x.

Let us define the following polyconvex elastic potential (internal energy) which serves as the distortion measure and is written as a weighted sum of the shape destortion measure and the volume distortion measure [9]:

$$W(C)=(1-\theta)\frac{\left(\frac{1}{d}\operatorname{tr}(C^TC)\right)}{\det C^{2/d}}+\frac{1}{2}\theta\left(\frac{1}{\det C}+\det C\right). \tag{1.1}$$

In most cases we set $\theta=4/5$.

We are looking for mesh deformation $x(\xi,t)$ being the solution of the following variational problem

$$F(x(\xi,t))=\int\limits_{\Omega_\xi} W(Q(x,t)\nabla_\xi x(\xi,t)H(\xi)^{-1})\det H\,d\xi, \tag{1.2}$$

In functional (1.2) time t is just a parameter.

Matrices $H(\xi)$ and $Q(x,t)$ are certain matrix factorizations of metric tensors G_ξ and G_x, respectively, defined by

$$H^T H = G_\xi, \det H > 0, \;\; Q^T Q = G_x, \det Q > 0.$$

We assume that singular values of the matrices Q and H are uniformely bounded from below and from above.

Time-dependent mesh deformation is introduced via time-dependent metric tensor $G_x(x,t) = Q^T(x,t)Q(x,t)$. Functional (1.2) attains its minimum equal to the volume of domain Ω_ξ in the metric G_ξ, when equality $Q\nabla_\xi x H^{-1} = U$ is satisfied, where U is arbitrary orthogonal matrix. It means that in the uniform coordinates $y = Qx$ associated with metric $G_x(x,t)$, when $H(\xi) = \nabla_\xi x(\xi,0)$, mapping $x(x,t)$ is locally isometric to mapping $x(\xi,0)$

Suppose that domain Ω_ξ can be partitioned into convex polyhedra D_k. We construct continuous piecewise-smooth deformation $x_h(\xi,\cdot)$, which is regular on each polyhedron. In practice we use linear and polylinear finite elements in order to assemble global deformation. We call integral over this deformation $F(x_h(\xi),t)$ by semidiscrete functional.

In order to approximate integral over a convex cell D_k one should use certain quadrature rules. As a result semi-discrete functional is replaced by the discrete functional:

$$F(x_h(\xi)) \approx \sum_k \mathrm{vol}(U_k) \sum_{q=1}^{N_k} \beta_q W(C_q) = F^h(x_h(\xi))$$

Here N_k is the number of quadrature nodes per cell D_k, C_q denotes the Jacobian matrix in q-th quadrature node of U_k, while β_q are the quadrature weights.

The following majorization property should hold

$$F(x_h(\xi)) \le F^h(x_h(\xi)) \tag{1.3}$$

This property can be used to prove that all intermediate deformations $x_h(\xi)$ providing finite values of discrete functional are homeomorphisms [2].

In order to derive discrete mesh functional special quadrature rules are applied which guarantee global invertibility of deformation mapping for finite values of discrete functional [2].

1.2 Variational predictor and mesh interpolation for stable time-dependent mesh deformation

Exact minimization of the discrete counterpart of variational problem (1.2) on each time level allows to obtain stable and accurate mesh deformation method.

Of course, in practice exact minimization is not acceptable. We have no other choice but to assume that approximate solution of the variational problem is far from exact minimizer.

Another problem is that due to nonlinearity of the Euler-Lagrange equations we cannot assume that norm of residual of the functional is reduced to zero at each time step. Thus time continuity of the moving mesh is not guaranteed since iterative minimization of functional with changed metric may result in considerable displacements even for infinitely small time steps and space-time trajectories of mesh cells may become extremely inclined. In this work we suggest very simple and efficient algorithm which is based on direct mesh interpolation. The idea of gradient search algorithm [8] is based on the predictor, which constructs minimization direction (displacement field) for each mesh vertex for large time increment. Every intermediate mesh computed via this displacement is guaranteed to be correct. We have found that the length of the computed displacement field is very large compared to displacements allowed by flow solver, hence number of costly implicit minimizations can be sharply reduced. Assuming that time dependence of metric is smooth function, we obtain mesh deformation/adaptation algorithm which requires one linear solve per, say, 5-10 time steps, making it quite efficient component of the moving mesh flow solver.

1.2.1 Preconditioned minimization algorithm and moving mesh interpolation strategy

It is convenient to write our discrete functional as a function $F(Z,Y,t)$ where spatial argument is the vector Z such that $Z^T = (z_1^T\, z_2^T\, \dots\, z_{n_v}^T)$ where $z_k \in \mathbb{R}^d$, $k = 1,\dots,n_v$ are positions of mesh vertices. Dependence on time t is introduced via time-dependent metric $G_x(y,t)$. Vector Y corresponds to the argument y of the metric G_x.

Hessian matrix $\tilde{H}$ of the function F with respect to Z is built of $d \times d$ blocks $\tilde{H}_{ij} = \frac{\partial^2 F}{\partial z_i \partial z_j^T}$. Here matrix $\tilde{H}_{ij}$ is placed on the intersection of i-th block row and j-th block column. We filter Hessian matrix during finite element assembly procedure to make it positive definite and to reduce number of nonzero elements by a factor of 2 [8]. Below we use the same notation $\tilde{H}$ for filtered Hessian matrix. Gradient of F with respect to Z is denoted by R. This vector consists of d-sized subvectors r_i. It is approximate gradient of the functional since we do not differentiate metric G_x hence dependence on argument Y is not taken into account.

One step of the Newton-type method for finding stationary point of the function can be written as follows

$$\sum_{j=1}^{n_v} \tilde{H}_{ij}(Z^l, Y^l, t+\Delta t)\delta z_j + r_i(Z^l, Y^l, t+\Delta t) = 0 \tag{1.4}$$

$$z_k^{l+1} = z_k^l + \mu_l \delta z_k,\ k = 1,\dots,n_v. \tag{1.5}$$

Here parameter μ_l is found as approximate solution of the following 1d problem

$$\mu_l = \arg\min_{\tau} F(Z^l + \mu\delta Z, Y^l, t + \Delta t). \tag{1.6}$$

The standard golden ratio algorithm is used to find approximate minimum. Obvious solution is now to assign new iteration using optimal parameter μ_l

$$Z^{l+1} = Z^l + \mu_l \delta Z \tag{1.7}$$

Preconditioned conjugate gradient technique is used for approximate solution with approximate second order Cholesky factorization [11] as a preconditioner. For parallel version of a solver we use additive version of approximate second order Cholesky factorization based on domain decomposition with overlaps [4]. The resulting nonlinear scheme was found to be very stable and efficient enough.

One can try to simplify the structure of the Hessian matrix by setting $\tilde{H}_{ij} = 0$ when $i \neq j$ [10] or even by considering only diagonal part of Hessian matrix as preconditioner. We have found that when global large deformations of initial mesh are necessary in order to satisfy mesh compression criteria inside thin moving layers, simple explicit methods of mesh optimization fail to follow metric precisely. Our observation is that "explicit" solvers are not efficient in the case of global deformations. Our linear solver is based on extraction from linear system (1.4) d independent linear systems with symmetric positive definite $n_v \times n_v$ matrices [8] closely resembling finite element approximations of scalar Laplace equations with tensor coefficient and using preconditioned conjugate gradient technique for their approximate solution.

Since our aim to use only one linear solve per time step, instead of repeating (1.4) - (1.7) we use another algorithm to account for dependence of metric G_x on the current mesh. We fix minimization direction and solve additional 1d minimization problem, namely, we start with

$$Y^0 = Z^l + \frac{1}{2}\mu_l \delta Z,$$

Coefficient $\frac{1}{2}$ is introduced to stabilize iterations and to avoid going too close to non-admissible set of deformations.

for $j = 0, \ldots, j_{\max}$

$$\tau_j = \arg\min_{\tau} F(Y^j + \tau\delta Z, Y^j, t + \Delta t). \tag{1.8}$$

$$Y^{j+1} = Y^j + \tau_j \delta Z$$

$$Z^{l+1} = Y^{j_{\max}}$$

We have found that 2-3 auxiliary 1d minimization steps are enough to essentially reduce τ_j and to stabilize mesh deformation on each time step as well as sharply improve accuracy of mesh layer positioning with respect to target position.

We observed that preconditioned gradient search technique allows for very large mesh deformation per time step. From the minimization strategy it follows that function $F(Z,Z,\cdot)$ has finite values for any vector Z_α defined by

$$Z_\alpha = \alpha Z^l + (1-\alpha)Z^{l+1},\ 0 \neq \alpha \neq 1$$

which means in turn that all intermediate mesh deformations are nondegenerate. Note that for the interpolation scheme to work, we should use single Newton-type step, i.e. $l = 0$ and single admissible increment direction δZ should be generated.

1.3 Mesh stretching and size gradation control

System of single or multiple moving and deforming bodies which are denoted as a domain $B \in \mathbb{R}^d$ defined by implicit function $d_s(x,t)$ in such a way that the function d_s is negative inside body, positive outside it, and isosurface $d_s(x,t) = 0$ defines the boundary ∂B at the time t. We assume that $d_s(x,t)$ resembles signed distance function for the instant domain boundary. In theory one can assume existence of the quasi-isometric mapping $x(y) : \mathbb{R}^d \to \mathbb{R}^d$ such that the function $d_s(x(y),t)$ is precisely the signed distance function. In practice we assume that when the vector $\nabla_x d_s$ in the vicinity of the boundary is defined its norm is bounded from below and from above by certain global constants.

Metric tensor $G(x,t)$ is defined as function of $d_s(x,t)$ in such a way that it attains largest value on the domain boundary and decreases using different laws inside and outside body. Mesh inside body is in general quite coarse since immersed boundary solver solution inside the body does not have physical meaning.

In order to attain smooth mesh size gradation we use logarithmic auxiliary mapping which allows to attain almost constant mesh size growth factor. Consider 1d auxiliary mapping $y(\xi) : [0,1] \to [0,1]$. We define uniform mesh in ξ coordinates with mesh size $h = 1/N$, where N is the number of cells. Vertices y_i are distributed assuming constant growth rate

$$y_{i+1} - y_i = (y_i - y_{i-1})(1+\varepsilon)$$

This equality can be considered as the finite difference approximation of equation

$$h\ddot{y} = \varepsilon\dot{y}$$

Interchanging dependent and independent variables we obtain

$$-\ddot{\xi}h = \varepsilon\dot{\xi}^2$$

The solution of this ODE is defined by the following formula

$$\xi = \phi(y) = \frac{h}{\varepsilon}\ln(1 + \frac{A\varepsilon}{h}(y-\delta)) + \delta A$$

We split "computational domain" into three subregions: "boundary layer" $0 <= y <= \delta$, transition zone $\delta \le y \le D < 1$ and outer zone $D \le y \le 1$. Since we apply mesh adaptation for better implementation of immersed boundary method for moving bodies, we just assign constant maximal mesh compression coefficient A inside boundary layer. For outer zone we also imply linear distribution with constant compression coefficient $\kappa = \frac{1-D}{1-\phi(D)} \le 2$. Function ϕ is used to control the transition between smallest and largest scales. The overall result is shown in Fig. 1.1, left. Note that we show here transposed graph.

The derivatives of this mapping are defined by the following equalities

$$\gamma(y,\delta,A) = \dot{\phi} = \frac{1}{\frac{1}{A} + \frac{\varepsilon}{h}(y-\delta)}, \;\; \ddot{\phi} = -\frac{\varepsilon}{h}\frac{1}{(\frac{1}{A} + \frac{\varepsilon}{h}(y-\delta))^2}$$

Mesh compression coefficient is defined by function $\dot{\phi}$. Its graph is hyperbola.

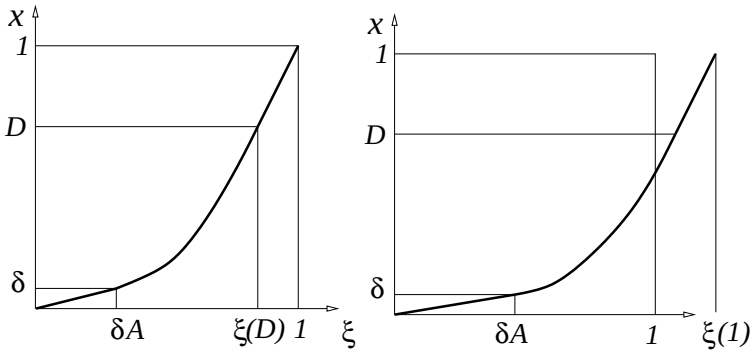

Fig. 1.1 Left - auxiliary 1d mapping defining stretching, right - target mesh compression ratio A is too large and should be reduced.

The set of control parameters δ, A, κ, h can be contradictory since for too large values of compression factor A one may get $\phi^{-1}(1) > 1$, or even $\phi^{-1}(D) > 1$. In this case we iteratively reduce the value of A with $\kappa = 2$ until equality $\phi^{-1(1)} = 1$ is satisfied.

In order to compute dimensionless control parameters of function ϕ we use $R_{\max}$ - radius of influence zone of the body, δ_R - true mesh layer thickness, average mesh size l_R in the direction normal to the body.

Then

$$h = \frac{3}{2}\frac{l_R}{R_{\max}}, \;\; \delta = \max\left(\frac{\delta_R}{R_{\max}}, \frac{h}{A}\right)$$

Suppose that the body is zero isosurface of the signed distance function $d_s(x,t)$. We compute normal compression function as follows

$$\sigma_1(x,t) = \gamma(|d_s(x,t)|/R_{\max}, \delta, A_n) \tag{1.9}$$

where A_n is compression factor in the normal direction to the boundary.

Denote by $u_1 = \nabla d_s/|\nabla d_s|$ unit vector of normal direction. We can obtain full orthonormal basis $U = (u_1, \ldots, u_d)$ using vectors $u_2, \ldots, u_d$ which define local tangent basis on the isosurface of d_s. Metric tensor $G_x(x,t)$ can be defined by equality

$$G_x = U\Sigma U^T,\ \Sigma = \mathrm{diag}(\sigma_i) \tag{1.10}$$

The choice of "tangential" stretches is not trivial. It is very important in order to obtain mesh deformation without size jumps and ruptures. "Rupture" means appearance of long and skewed mesh cells which resembles approximation of the rupture of elastic material.

The isotropic choice $\sigma_i = \sigma_1$ or $G_x = \sigma_1^2 I$ seems to be the simplest one. Unfortunately its applicability is very limited, since preimage of the boundary layer in the initial coordinates ξ is scaled by a factor A_n, as shown in Fig. 1.2 for $A_n = 6$. Enlarged preimage may cover all initial doman or even go ouside boundary meaning that most of the mesh cell will travel into boundary layer making resulting mesh unusable.

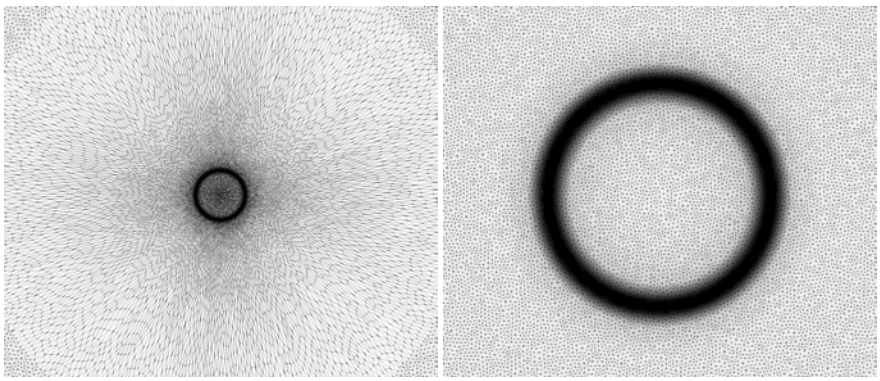

Fig. 1.2 Isotropic metric: internal layer in computational domain and its preimage in parametric domain.

If we specify maximal tangential compression A_t, then the linear size of preimage would increase by the factor A_t, while thickness of preimage layer will increase by factor A_n, as shown in Fig. 1.3 for $A_n = 6, A_t = 2$.

We assign maximal anisotropy ratio A_n/A_t in the boundary layer and build σ_i to gradually reduce anisotropy away from the layer. In some cases constant A_t factor does not allow to resolve boundary features, in particular in the presence of sharp or highly curved boundary fragment. General argument is that when tangential resolution is not enough one should use smart values of tangential stretches $A^*_{t\,i}$, $i = 2, \ldots, d$ in different directions in the range $A_t \le A^*_{t\,i} \ge A_n$. Let us denote metric with constant stretches in boundary layer by G_1, and consider metric introduced around surface features by G_2. We use the same representation (1.9), (1.10) for G_2, the main difference being the influence radius R_2, which should be smaller compared

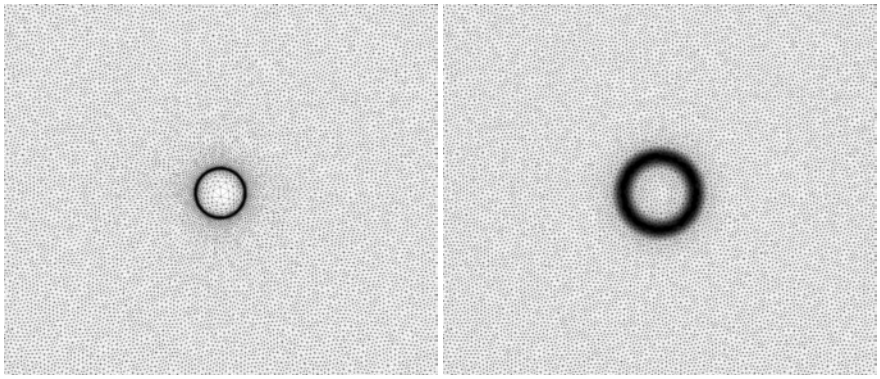

Fig. 1.3 Anisotropic metric: internal layer in computational domain and its preimage in parametric domain.

to global influence radius $R_{\max}$ for G_1. Then final metric is defined by

$$G_x = G_3 = \max(G_1, G_2)$$

The maximum operation is based on common circum-ellipsoid construction which is close to the one suggested in [7] and is shown in Fig. 1.4. Consider two concentric ellipsoids M_1 and M_2 defined by quadratic forms $x^T G_1^{-1} x = 1$ and $x^T G_2^{-1} x = 1$, respectively. Common circum-ellipsoid M_3 defines matrix G_3. Construction of ellipsoid M_3 is simple: find affine map which transforms one of the ellipsoids into sphere. In transformed coordinates circum-ellipsoid is trivially constructed and mapped back into original coordinates. Since each metric is defined by its spectral decomposition, this construction is reduced to a number of products by orthogonal and diagonal matrices.

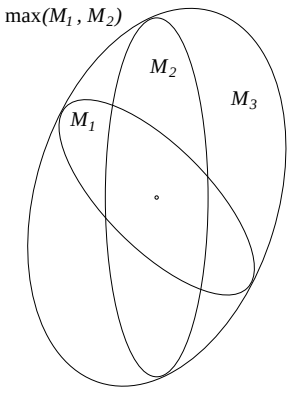

Fig. 1.4 Illustration of maximum operation for two metrics.

1.3.1 MPI-based Parallel implementation

We use spatial mesh decompostion in order to build parallel algorithm. Since degrees of freedom which define mesh deformation are mesh vertices, we build consistent partitioning of mesh cells and vertices: parametric domain $\Omega_x i$ is split into connected subdomains consisting of full mesh cells. Mesh vertices belonging to boundaries between subdomains are distributed between subdomains. We use parallel ILU2-based iterative solver [11], [4] to compute minimization direction. Input data for this solver are right hand side partitioned into blocks and sparse matrix partitioned into block rows. Each block precisely corresponds to the partitioning of mesh vertices. Our implementation of iterative scheme is based on extended subdomains defining two cell-wide subdomain overlap. At the beginning of each iteration of minimization (1.5) we use data exchange to create current guess at each extended subdomain. Such an extension makes cell-by-cell assembly of functional, its gradient and the Hessian matrix and its subsequent double scaling completely local operations. Additional data exchanges are not necessary in order to find optimal value of τ by approximate solution of 1d minimization problem (1.6), just global sums should be computed. As one can see all operations of mesh generation algorithm are fully scalable hence one can expect that overall scalability is defined by that of parallel linear solver. Note that linear solver uses its own overlaps and data exchange scheme [4].

1.4 Numerical experiments

We apply mesh deformation solver for geometric adaptation of computational mesh in order to improve the resolution of the NOISETTE flow solver [6] for numerical modelling of turbulent flow around propeller of quadrocopter. Flow solver is based on Immersed Boundary Conditions (IBC). We test the numerical technology in the 2d case using 2d projection of realistic propeller geometry [5] with the main goal to extend it to the case of real 3d propeller modeling. At this stage we run simple 3d tests in order to check initial geometric adaptation, to compare computed position of mesh layer with target moving position as well as check efficiency and scalability of 3d algorithm. The geometry of propeller is shown in Fig. 1.5 (left). Due to natural geometrical limitations mesh is deformed only inside a circle which is 10% larger compared to propeller itself, meaning that mesh adaptation near blade tip become nontrivial task. Propeller is defined by the implicit signed distance-like function. We impose compression factor 30 in the normal direction inside thin layer near the blades. In order to initiate time-dependent deformation we start from initial mesh adaptation to a fixed shape. This mesh shown in Fig. 1.5 (right).

In order to resolve corners and highly curved boundary fragment we introduce locally isotropic metric influence zones (see Fig. 1.6). Isotropic metric is defined by the same law (1.9) with different constants. Maximum operation for metrics is applied in order to compute G_x.

Fig. 1.7 shows fragments of initial mesh near influenze zones of anisotropic metric

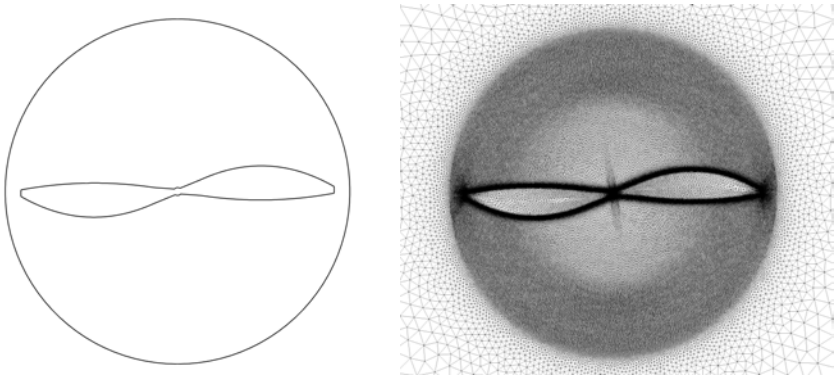

Fig. 1.5 2d propeller model and initial adapted mesh.

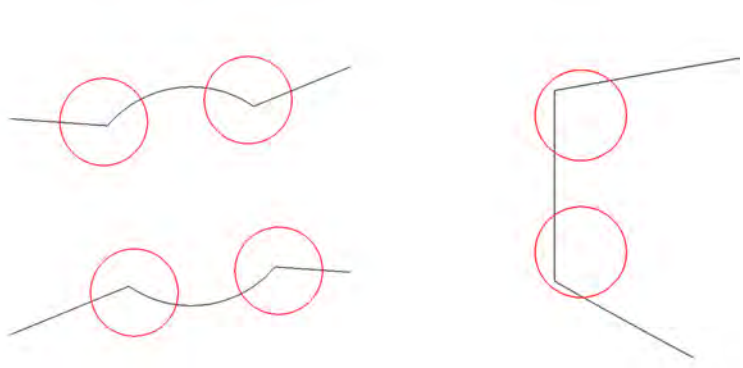

Fig. 1.6 Influence zones of isotropic metric near sharp vertices.

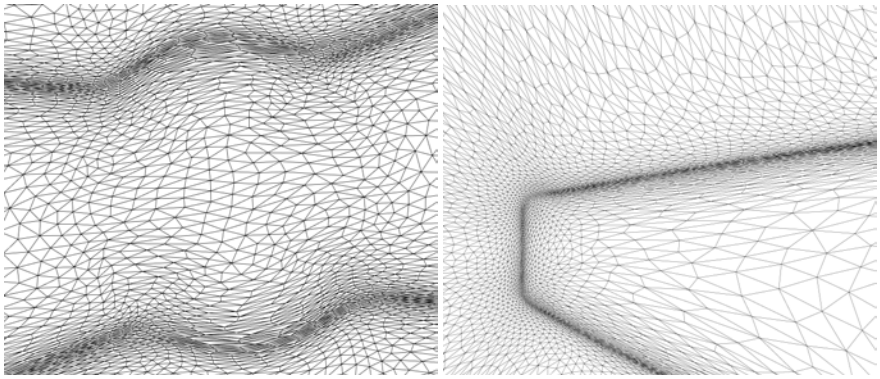

Fig. 1.7 Fragments of initial adapted mesh near sharp vertices.

Fig. 1.8 shows general mesh outline after first and second rotation of propeller.

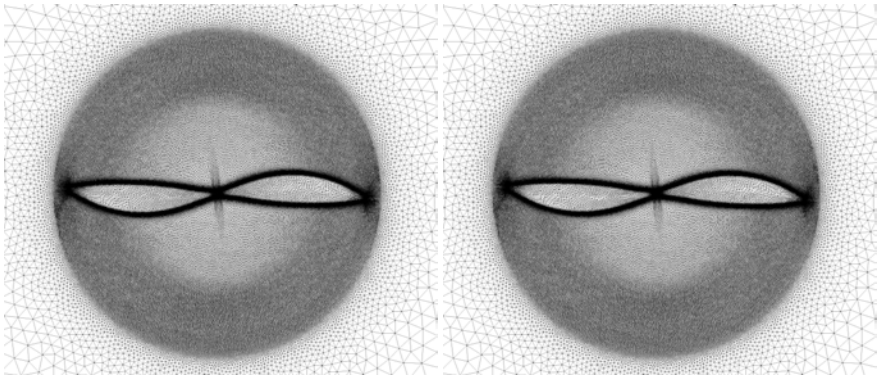

Fig. 1.8 Mesh after first and second rotation.

Figs. 1.9-1.10 show mesh fragments after first and second rotation.

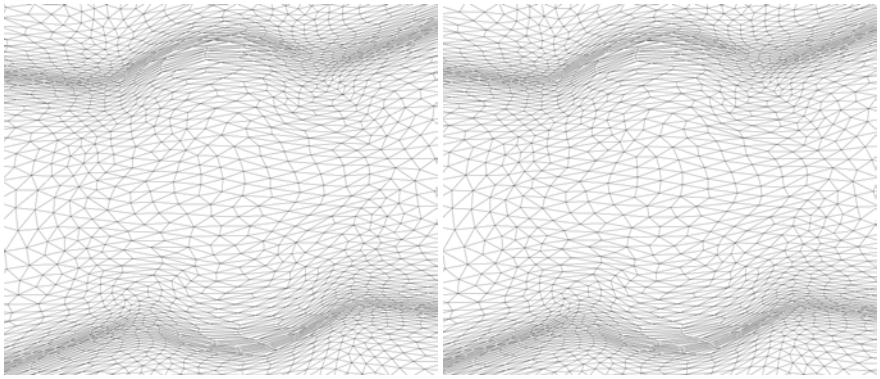

Fig. 1.9 Mesh fragment after first and second rotation.

Computation of long term mesh deformation demonstrates behaviour which is close to the periodic one. Fig. 1.11 show trajectories of selected cells saved every 500 time steps.

Fig. 1.12 shows deformation of mesh subdomains when propeller spans quarter of the rotation.

Fig. 1.13 shows mesh deformation and movement of subdomain boundaries in the propeller-related reference frame.

Fig. 1.14 shows target σ_1 distribution in the rotating reference frame in two different time levels.One can observe that position of mesh layer follows controls quite closely. Quite interesting effect is related to the bands of mesh cells which are attracted to the blade, then travel some time along the boundary layer in the

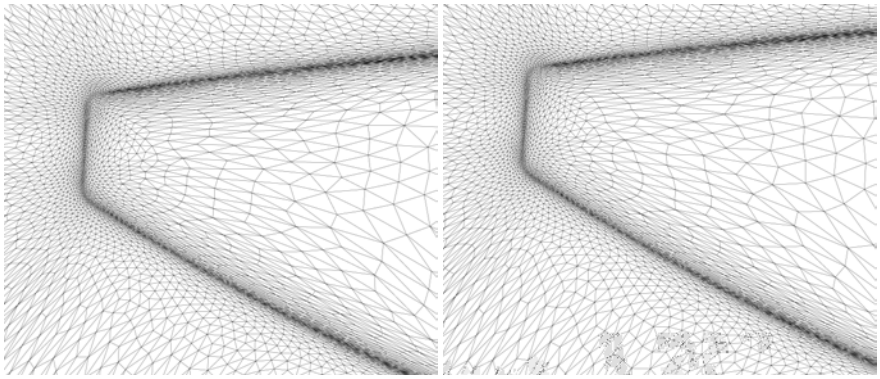

Fig. 1.10 Mesh fragment after first and second rotation.

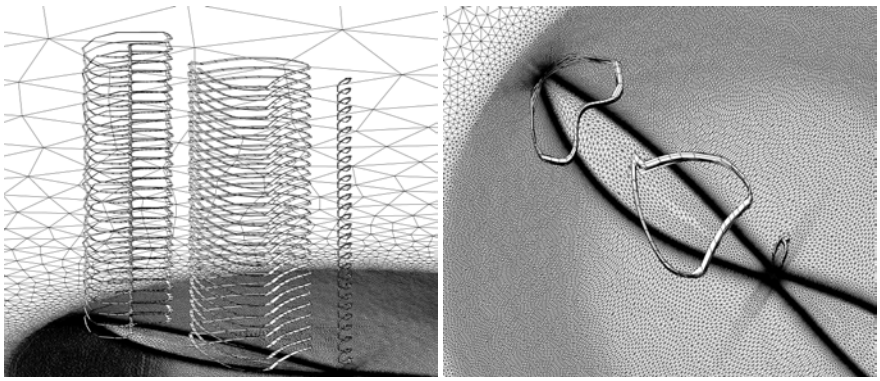

Fig. 1.11 Cell trajectories for several rotations.

compressed state and eventually leave the propeller. Another group of cell in the middle of the domain is attracted to the blades, cross them and go out at the other side. Right figure in Fig. 1.14 is slightly inclined in 3d x_1, x_2, t coordinates in order to make trajectories of cells more visible.

In order to evaluate scalability of the parallel implementation, we run several 2d and 3d test cases. In 2d we consider "small" problem with 357781 vertices and 713760 triangles and "moderate" problem with 1429321 vertices and 2855040 triangles. In the 3d case we consider deformation of tetrahedral mesh with 9586347 vertices and 56715408 tets.

Scalability experiments were ran on the parallel cluster of Moscow Institute of Physics and Technology. It is Intel CPU-based cluster with 24 cores per board and Infiniband interconnect. Fig. 1.15 illustrates scalability of the mesh deformation algorithm. Separate graph is devoted to linear solver. As one can expect, overall scalability is defined by the linear solver scalability. Note that scalability with respect to single core is not impressive, while results scale very well with respect to 24 cores.

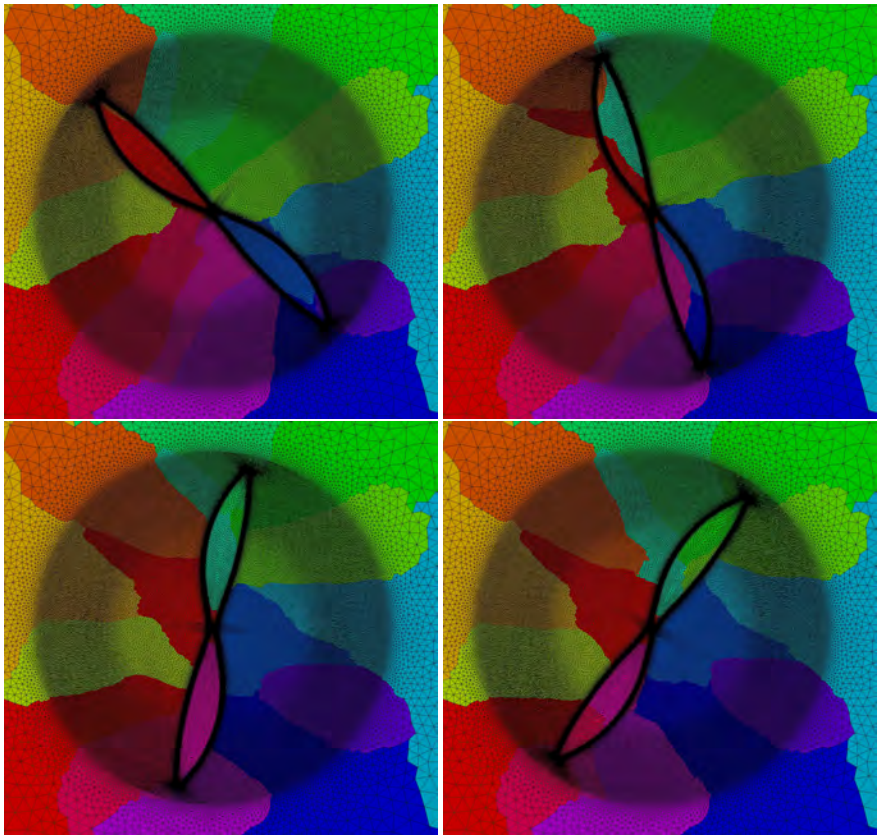

Fig. 1.12 Deformation of subdomains for quarter of rotation.

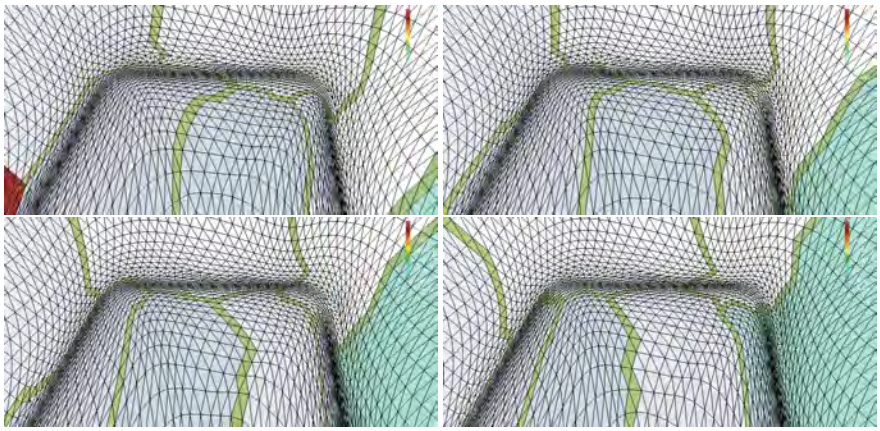

Fig. 1.13 Mesh deformation and movement subdomains in the rotating reference frame.

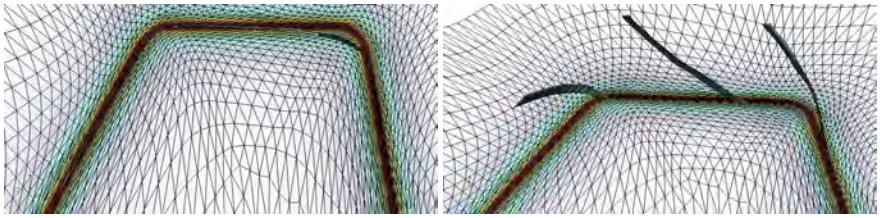

Fig. 1.14 Mesh layer localization and cell trajectories in the rotating reference frame.

We were not able to explain this observation. In principle, it may be drawback of the algorithm or cluster software/hardware misconfiguration artefact.

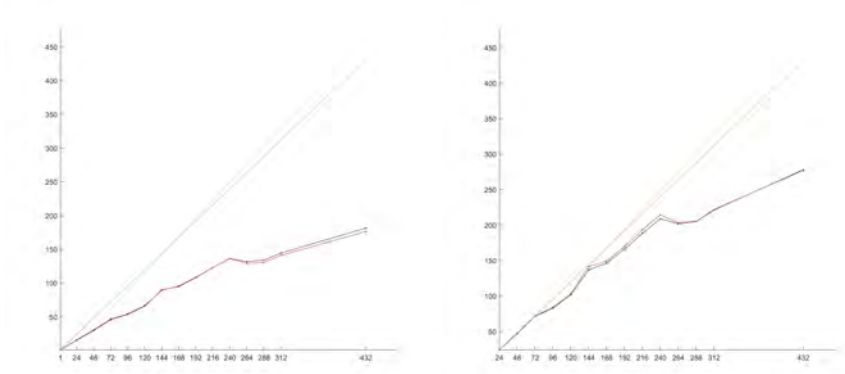

Fig. 1.15 Speed-up versus number of computational cores for moderate-sized 2d problem, (a) - speed-up with respect to single core, (b) speed-up with respect to 24 cores

Fig. 1.16(a) shows speed-up for small scale 2d problem where saturation is quite pronounced for more then 120 cores, while for 3d case 1.16(b) scalability is quite reasonable and saturation is not observed until 600 cores. We plan to run 3d algorithm on larger configurations.

Fig. 1.17 illustrates subdomains for 3d problem in the case of 144 cores. We show trace of the subdomains on the boundary, cross-section of initial uniform mesh and cross-section of mesh adapted to a moving sphere.

Fig. 1.18(a) shows initial adapted mesh and target σ_1 distribution. Fig. 1.18(b) shows preimage of the compressed boundary layer on the initial uniform 3d mesh. In this example in the boundary layer $\sigma_1 = 30$ and $\sigma_{2,3} \approx 3$.

Fig. 1.19 shows fragments of mesh with two positions of computed mesh layer in the process of sphere movement.

Fig. 1.20 show two positions of mesh layer extracted from the global meshes using threshold 20 for target σ_1 distribution.

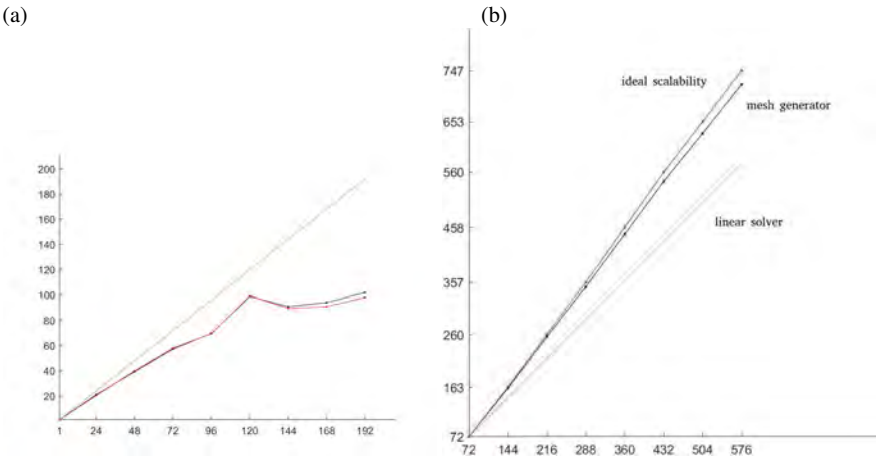

Fig. 1.16 Speed-up versus number of computational cores. (a) Small 2d problem, (b) 3d problem

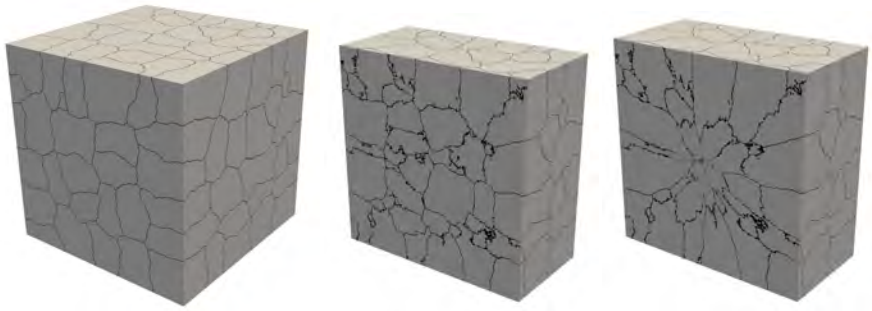

Fig. 1.17 Subdomains for initial uniform mesh and for instant moving deformed 3d mesh

1.5 Conclusions and discussions

We describe algorithm which allows to construct high quality time-dependent mesh deformations via approximate minimization of quasi-isometric functional. Presented algorithm is still slower compared to mesh solvers based on linear elliptic equations, see e.g. [3], however difference is no longer crucial since we use d linear solves with linear systems corresponding to FE approximations of scalar Laplace-like equations per several time steps. It is well known, that algorithms based on linear elliptic solver can attain reasonable mesh quality via careful choice of metric tensor/weight functions [1] so it may happen that advantage of presented method in terms of mesh quality does not overweight computational overhead. However, unlike linear mesh solvers, presented algorithm does not have any limitation on the shape of the domain and the type of the mesh elements. It can be applied in the case of multiple dimensions and for high-order elements. Due to advanced parallel linear solver quite reasonable parallel scalability of numerical algorithm was demonstrated.

(a) (b)

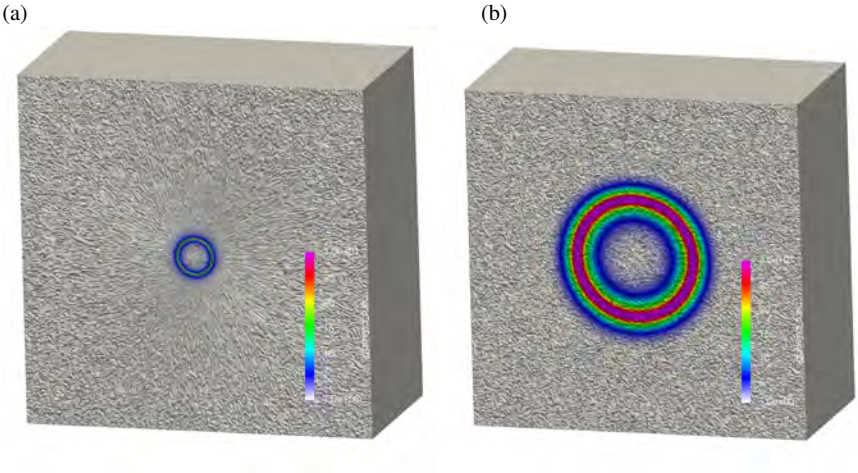

Fig. 1.18 Distribution of mesh compression factor in normal direction and its preimage on initial mesh

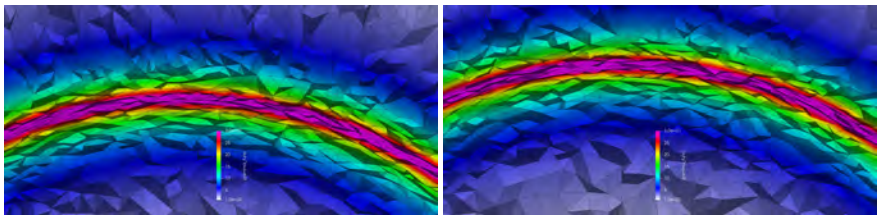

Fig. 1.19 Two positions of moving layer with imposed distribution of mesh compression factor

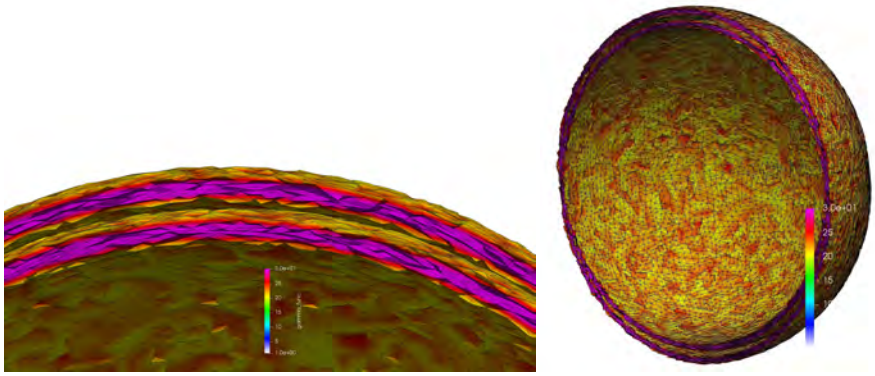

Fig. 1.20 Two positions of compressed layer extracted using threshold 20 for normal compression factor

Research is supported by Russian Science Foundation, Project 20-41-0901_ANR (acronym NORMA).

References

1. Van Dam, A., Zegeling, P.A.: Balanced monitoring of flow phenomena in moving mesh methods. Communications in Computational Physics **7**(1), 138–170 (2010)
2. Garanzha, V.A., Kudryavtseva, L.N., Utyzhnikov, S.V.: Untangling and optimization of spatial meshes. Journal of Computational and Applied Mathematics. **269**, 24–41 (2014)
3. Tang, H.Z., Tang, T. Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. SIAM J. Numer. Anal. **41**(2), 487–515 (2003)
4. Kaporin, I.E., Milyukova, O.Yu. MPI+OpenMP parallel implementation of explicitly preconditioned conjugate gradient method. Keldysh Institute preprints, **008** (Mi ipmp2369) (2018)
5. Brandt J.B. Small-scale propeller performance at low speeds : PhD thesis – University of Illinois at Urbana-Champaign (2005).
6. Tsvetkova V.O., Abalakin I.V., Bobkov V.G., Zhdanova N.S., Kozubskaya T.K., Kudryavtseva L.N. Simulation of flow near rotating propeller on adaptive unstructured meshes using immersed boundary method. Accepted for publication in Mathematical Models and Computer Simulations, 2020.
7. Castro-Díaz M.J., Hecht F., Mohammadi B., Pironneau O. Anisotropic unstructured mesh adaption for flow simulations. International Journal for Numerical Methods in Fluids, 1997, 25 (4), 475-491.
8. V.A. Garanzha, L.N. Kudryavtseva. Hyperelastic springback technique for construction of prismatic mesh layers, Procedia Engineering 203 (2017) 401–413.
9. V.A. Garanzha, The barrier method for constructing quasi-isometric grids. *Comput. Math. Math. Phys.* 40(2000) 1617–1637.
10. S.A. Ivanenko, Construction of nondegenerate grids. Comput. Math. and Math. Phys. 28(1988) 141-146.
11. Kaporin, I. E. (1998). High quality preconditioning of a general symmetric positive definite matrix based on its UTU+UTR+RTU-decomposition. Numerical linear algebra with applications, 5(6), 483-509.