MECHANICS

UDC 517.95, 519.6 https://doi.org/10.23947/2687-1653-2021-21-3-222-230

Development of algorithms for constructing two-dimensional optimal boundary-adaptive grids and their software implementation

A. E. Chistyakov [1], **V. V. Sidoryakina** [2], **S. V. Protsenko** [1]

[1] Don State Technical University (Rostov-on-Don, Russian Federation)

[2] Taganrog Institute Named after A.P. Chekhov, Rostov State University of Economics (RINH) branch, (Taganrog, Russian Federation)

✉ cheese_05@mail.ru

Introduction. It is noted that the use of adaptive grids in calculations makes it possible to improve the accuracy and efficiency of computational algorithms without increasing the number of nodes. This approach is especially efficient when calculating nonstationary problems. The objective of this study is the development, construction and software implementation of methods for constructing computational two-dimensional optimal boundary-adaptive grids for complex configuration regions while maintaining the specified features of the shape and boundary of the region. The application of such methods contributes to improving the accuracy, efficiency, and cost-effectiveness of computational algorithms.

Materials and Methods. The problem of automatic construction of an optimal boundary-adaptive grid in a simply connected region of arbitrary geometry, topologically equivalent to a rectangle, is considered. A solution is obtained for the minimum set of input information: the boundary of the region in the physical plane and the number of points on it are given. The creation of an algorithm and a mesh generation program is based on a model of particle dynamics. This provides determining the trajectories of individual particles and studying the dynamics of their pair interaction in the system under consideration. The interior and border nodes of the grid are separated through using the mask tool, and this makes it possible to determine the speed of movement of nodes, taking into account the specifics of the problem being solved.

Results. The developed methods for constructing an optimal boundary-adaptive grid of a complex geometry region provides solving the problem on automatic grid construction in two-dimensional regions of any configuration. To evaluate the results of the algorithm research, a test problem was solved, and the solution stages were visualized. The computational domain of the test problem and the operation of the function for calculating the speed of movement of interior nodes are shown in the form of figures. Visualization confirms the advantage of this meshing method, which separates the border and interior nodes.

Discussion and Conclusions. The theoretical and numerical studies results are important both for the investigation of the grids qualitative properties and for the computational grid methods that provide solving numerical modeling problems efficiently and with high accuracy.

Keywords: particle dynamics method, computational two-dimensional grid, boundary-adaptive grid, numerical simulation.

***For citation*:** A. E. Chistyakov, V. V. Sidoryakina, S. V. Protsenko. Development of algorithms for constructing two-dimensional optimal boundary-adaptive grids and their software implementation. Advanced Engineering Research, 2021, vol. 21, no. 3, pp. 222–230. https://doi.org/10.23947/2687-1653-2021-21-3-222-230

Funding information: the research is done with the financial support from RFFI (project no. 19−01−00701).

Introduction. The grid construction was initially considered as a necessary auxiliary step in solving other problems [1–3]. In particular, when solving computational fluid dynamics problems, the construction of a computational grid is a considerably labor-intensive and lengthy process [4–6]. When studying the water areas of real reservoirs, we have to deal with the areas with an objectively predetermined boundary running along the coastline[1]. The border nodes of the rectangular grid of Cartesian coordinates covering the reservoir may not exactly fall on the contour of the coastline. Therefore, a uniform grid should be very dense, so that the errors it introduces for setting the coastline are acceptable. For example, for the Azov Sea, a two-dimensional grid, as a rule, contains more than half a million nodes [7–11]. Note also that when using rectangular grids, the boundary conditions are set at points offset from the real boundary, or (when truncating the boundary grid cells) inhomogeneities associated with uneven steps in spatial directions are concentrated near the boundary[2]. Taking into account the above, as well as the need to simplify data structures and algorithms for their processing, it is advisable to conduct numerical modeling of this type of problems on an optimal boundary-adaptive grid [12–14].

This paper presents an algorithm and a program developed for constructing quadrangular optimal boundary-adaptive computational grids based on the particle dynamics method in a two-dimensional formulation. This powerful computational method provides the grid nodes as charged particles and simulates the dynamics of a system consisting of a huge number of particles (up to a million). Using the example of text problems, the efficiency of the algorithm for areas with a complex boundary is demonstrated.

Materials and methods. Description of the method for constructing a 2D optimal boundary-adaptive grid. In the Cartesian coordinate system Oxy, we introduce area D. In area D, grid $\omega = \left\{ \left(x_{i,j}, y_{i,j} \right),\ i = \overline{1, N_1},\ j = \overline{1, N_2} \right\}$ is built according to the specified coordinates of the border nodes. The grid area divides area D into elementary subdomains in the form of quadrilaterals. The grid nodes are redistributed along the coordinate lines. Then, in the needed zones where node thickening is required, variables x, y are replaced with compressing variables ξ, η c using separated transformations:

$$(\xi, \eta) \to (x, y):\ x = x(\xi, \eta),\ y = y(\xi, \eta), \tag{1}$$

$$(x, y) \to (\xi, \eta):\ \xi = \xi(x, y),\ \eta = \eta(x, y). \tag{2}$$

Thus, grid $\omega = \left\{ \left(\xi_{i,j}, \eta_{i,j} \right),\ i = \overline{1, N_1},\ j = \overline{1, N_2} \right\}$, which is determined by the functions $x = x(\xi, \eta),\ y = y(\xi, \eta)$, is constructed on the plane (ξ, η).

The construction of the computational grid ω is based on the method of particle dynamics. This modeling technique is widely presented in the literature, so we will focus on it only briefly [15–17].

We represent nodes of grid ω as a collection of particles with charges q_{ij} and mass m_{ij}, that move in the calculated area D along and near its boundary. The particles interact with each other, and the interaction forces are of an electrical nature.

[1]Sukhinov AI, Sukhinov AA. Reconstruction of 2001 ecological disaster in the Azov Sea on the basis of precise hydrophysics models. Parallel Computational Fluid Dynamics 2004, Multidisciplinary Applications. Amsterdam: Elsevier; 2005. P. 231–238. https://doi.org/10.1016/B978-044452024-1/50030-0

[2]Sukhinov A, Chistyakov A, Sidoryakina V. Investigation of nonlinear 2D bottom transportation dynamics in coastal zone on optimal curvilinear boundary adaptive grids. In: Proc. XIII Int. Sci.-Tech. Conf. on Dynamic of Technical Systems. 2017;132:04003. https://doi.org/10.1051/matecconf/201713204003

According to Coulomb's law, repulsive force $\dot{F}_{ij}$ acts on a single i-th particle from the side of j-th particle. Absolute magnitude of force $\dot{F}_{ij}$ is determined by the distance between these particles, and its vector is directed opposite to the radius-vector $\dot{r}_{ij}$, connecting the i-th and j-th charges (Fig. 1).

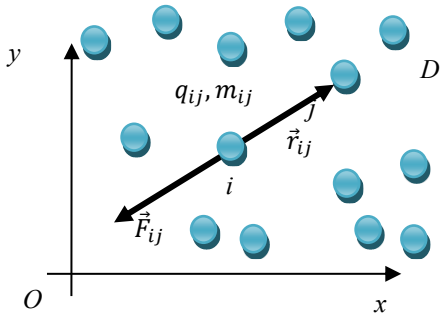


Fig. 1. Scheme of particle interaction

The trajectories of the charged particles determine the location of the grid nodes. Let us denote the coordinates of the i-th particle $\left(x_{ij}, y_{ij}\right)$, and j-th — $\left(\xi_{ij}, \eta_{ij}\right)$. The length of radius-vector $\dot{r}_{ij}$, that determines the movement of the node $\left(x_{ij}, y_{ij}\right)$ to the node $\left(\xi_{ij}, \eta_{ij}\right)$, is calculated from the formula:

$$r_{ij} = r\left(x_{ij}, y_{ij}, \xi_{ij}, \eta_{ij}\right) = \sqrt{\left(x_{ij} - \xi_{ij}\right)^2 + \left(y_{ij} - \eta_{ij}\right)^2}. \quad (3)$$

Note the specificity of the transformation (1)–(2). If the distance is $r \neq 0$, then the node $\left(\xi_{ij}, \eta_{ij}\right)$ repels from each of the neighboring ones. They, in turn, must either stand at the prescribed distance, or move away, being pulled to a neighboring node and freeing up space for the newly inserted one.

The type of transformation (1)–(2) that compresses coordinates x, y in zones of high gradients is determined by the solution to the problem. For this purpose, a model equation describing the potential is used in the direction of axes Ox, Oy:

$$F\left(x_{ij}, y_{ij}, \xi_{ij}, \eta_{ij}\right) = \frac{l}{r\left(x_{ij}, y_{ij}, \xi_{ij}, \eta_{ij}\right)^{\alpha}}, \quad (4)$$

where l — proportionality factor between potential and distance r at a given node, α — a certain parameter.

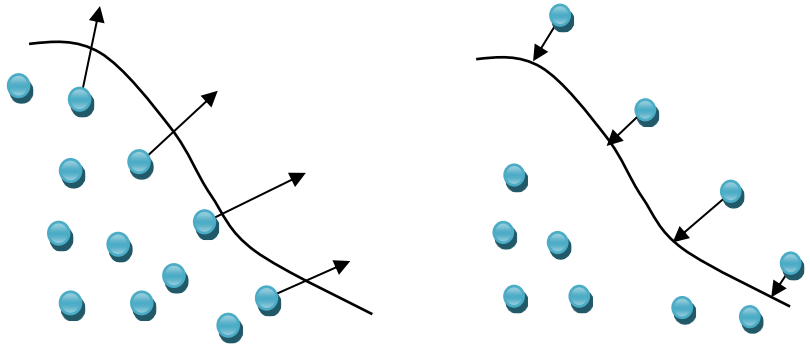
Each node seeks to reduce potential energy, namely:

$$\sum_{m,n} F\left(x_{ij}, y_{ij}, x_{nm}, y_{nm}\right) \to min,\ i = \overline{1, N_1},\ j = \overline{1, N_2},\ n = \overline{1, N_1},\ m = \overline{1, N_2}. \quad (5)$$

Force $\vec{f}\left(x_{ij}, y_{ij}\right)$ is related to potential $F = F\left(x_{ij}, y_{ij}, \xi_{ij}, \eta_{ij}\right)$ through the following relation:

$$\vec{f}\left(x_{ij}, y_{ij}\right) = grad\left(F\right) = -\sum_{m,n} \frac{\alpha l}{r\left(x_{ij}, y_{ij}, x_{nm}, y_{nm}\right)^{\alpha+2}} \vec{r}. \quad (6)$$

When modeling the process of interaction of mobile particles, we assume that the grid nodes that fall outside the calculated domain are forced to move to a point on the boundary of the domain, the distance to which is minimal. The scheme of interaction between a mobile particle and a particle at the boundary is shown in Fig. 2.



a) *b*)

Fig. 2. Scheme of particle interaction at the boundary of computational domain D :
a) direction of movement of particles inside domain *D; b*) direction of movement of particles outside domain *D*

The technology of numerical integration of the equations of motion is based on the algorithm presented in [17].

Description of the meshing software. The program consists of modules that implement five functions: a control one that calls the calculation functions a specified number of times, three calculation functions, and a visualization function. The calculation functions are the following: a function describing the speed at which nodes move; a function for moving nodes; a function for checking the exit of nodes beyond the boundary of the domain.

The data structure of the program: A — input array of sizes $[N,M]$; C — visualization array; B_x, B_y — arrays of sizes $[n,m]$, describing the location of nodes; B_m — array of sizes $[n,m]$, describing the masks of boundary conditions; u, v — components of the velocity vector of the nodes; i, j — counters; n, m — number of nodes in the directions Ox, Oy, respectively; l — proportionality coefficient between the potential and distance r at a given node; α — some parameter (degree at r). The control function resets the arrays and sets the initial location of the nodes.

Algorithm describing the node speed function.

Input arrays: B_x, B_y, B_m and parameter α (i.e., $\alpha = 3$). Output arrays: u, v.

1. Start of the loop on variables i, j. The counter values are set equal to $i = 0,...,n-1$, $j = 0,...,m-1$...

2. Separation of border and interior points of the computational domain. If mask = 1 — point on the domain boundary, go to item 3, if mask = 2 — point is inside the domain, go to item 4.

3. Algorithm describing the speed function of the border nodes

3.1. Zeroing arrays u, v.

3.2. Start of the loop on variables $i1$, $j1$. The counter values are set equal to $i1 = i-2,...,i+2$, $j1 = j-2,...,j+2$.

3.3. Calculating the distance from one node to another:

$$r \leftarrow \sqrt{\left(Bx_{i,j} - Bx_{i1,j1}\right)^2 + \left(By_{i,j} - By_{i1,j1}\right)^2}.$$

3.4. Checking the condition. If $r > 0$, then go to item 3.5, otherwise, go to item 3.6.

3.5. Calculating u, v for $\alpha = 3$:

$$u_{i,j} \leftarrow u_{i,j} + \frac{\left(Bx_{i,j} - Bx_{i1,j1}\right)}{r^{\alpha}} Bm_{i1,j1},$$

$$v_{i,j} \leftarrow v_{i,j} + \frac{\left(By_{i,j} - By_{i1,j1}\right)}{r^{\alpha}} Bm_{i1,j1}.$$

3.6. Build up counters $i1$, $j1$ and go to item 3.3.

3.7. Build up counters i, j and go to item 3.1.

4. Algorithm describing the speed function of the interior nodes

4.1. Zeroing arrays u, v.

4.2. Start of the loop on variables $i1$, $j1$. The counter values are set equal to $i1 = i-1,...,i+1$, $j1 = j-1,...,j+1$.

4.3. Checking the condition. If $(i1-i)(j1-j)=0$, then go to item 4.4, otherwise, go to item 4.8.

4.4. Calculating the distance from one node to another:

$$r \leftarrow \sqrt{\left(Bx_{i,j} - Bx_{i1,j1}\right)^2 + \left(By_{i,j} - By_{i1,j1}\right)^2}.$$

4.5. Checking the condition. If $r > 0$ is executed, then go to item 4.6, otherwise, go to item 4.7.

4.6. Calculating u, v for $k = 0.005$:

$$u_{i,j} \leftarrow u_{i,j} - k\left(Bx_{i,j} - Bx_{i1,j1}\right),$$
$$v_{i,j} \leftarrow v_{i,j} - k\left(By_{i,j} - By_{i1,j1}\right).$$

4.7. Build up counters $i1$, $j1$ and go to item 4.4.

4.8. Build up counters i, j and go to item 4.1.

Algorithm describing the function of calculating the movement of nodes

Input arrays: B_x, B_y, u, v and parameter l (i.e., $l = 30$). Output arrays: B_x, B_y.

1. Start of the loop on variables i, j. The counter values are set equal to $i = 1,...,n-2$, $j = 1,...,m-2$.

2. Calculating arrays B_x, B_y:

$$Bx_{i,j} \leftarrow Bx_{i,j} + lu_{i,j},$$
$$By_{i,j} \leftarrow By_{i,j} + lv_{i,j}.$$

3. Build up counters i, j and go to item 2.

Algorithm describing the function of checking the exit of nodes beyond the domain boundary

Input arrays: B_x, B_y, A and parameter d (i.e., $d = 3$), that describes the size of the window in which the computational domain is presented in case node B_x, B_y goes beyond the boundary of the computational domain. Output arrays: B_x, B_y.

1. Start of the loop on variables i, j. The counter values are set equal to $i = 1,...,n-2$, $j = 1,...,m-2$.

2. Finding indexes A, in the array corresponding to node $\left(Bx_{i,j}, By_{i,j}\right)$:

$$i1 \leftarrow \left|Bx_{i,j}\right|, \quad j1 \leftarrow \left|By_{i,j}\right|.$$

3. The initial value of the distance to the boundary is set by the parameter value d:

$$r \leftarrow 2d.$$

4. Checking the condition for the exit of point $(i1, j1)$ beyond the computational domain boundary: if $A_{i1,j1} > 0$ is executed; otherwise, go to item 12.

5. Start of the loop on variables $i2$, $j2$. The counter values are set equal to $i2 = -d,...,d$, $j2 = -d,...,d$.

6. Checking point $(i2+i1, j2+j1)$ for belonging to the calculated domain: if $A_{i2+i1,j2+j1} = 0$ is executed; otherwise, go to item 10.

7. The distance from node $\left(Bx_{i,j}, By_{i,j}\right)$ to point $(i2+i1, j2+j1)$ is found from the formula:

$$r1 \leftarrow \sqrt{\left(Bx_{i,j} - i2 - i1\right)^2 + \left(By_{i,j} - j2 - j1\right)^2}.$$

8. If $r1 > r$ is executed, then item 9 is performed; otherwise, go to item 10.

9. Remember the point of the computational domain closest to the node $\left(Bx_{i,j},\ By_{i,j}\right)$:

$$r \leftarrow r1, \quad i3 \leftarrow i2 + i1, \quad j3 \leftarrow j2 + j1.$$

10. Build up counters by variables $i2, j2$ and go to item 6.

11. Offset of the node $\left(Bx_{i,j},\ By_{i,j}\right)$ to the boundary of the computational domain:

$$Bx_{i,j} \leftarrow i3, \quad By_{i,j} \leftarrow j3.$$

12. Build up counters $i,\ j$ and go to item 2.

Algorithm describing the visualization function.

Input arrays: $B_x,\ B_y,\ A$. Output array — C.

1. We put array *A* in the visualization array.

2. Start of the loop on variables $i,\ j,\ k$. The counter values are set equal to $i = 1,...,n-3,\ j = 1,...,m-2$, $k = \left|Bx_{i,j}\right|...\left|Bx_{i+1,j}\right|$.

3. Draw vertical lines:

$$C\left[k, \left|By_{i,j} + \frac{By_{i+1,j} - By_{i,j}}{Bx_{i+1,j} - Bx_{i,j}} \cdot \left(k - Bx_{i,j}\right)\right|\right] \leftarrow 255.$$

4. Build up counters $i,\ j,\ k$ and go to item 3.

5. Start of the loop on variables $i,\ j,\ k$. The counter values are set equal to $i = 1,...,n-2,\ j = 1,...,m-3$, $k = \left|By_{i,j}\right|...\left|By_{i,j+1}\right|$.

6. Draw horizontal lines:

$$C\left[k, \left|By_{i,j} + \frac{By_{i+1,j} - By_{i,j}}{Bx_{i+1,j} - Bx_{i,j}} \cdot \left(k - Bx_{i,j}\right)\right|\right] \leftarrow 255.$$

7. Build up counters $i,\ j,\ k$ and go to item 6.

The source file is BMP. On it, the geometry of the domain on which the grid is being built is indicated in black. The rest of the domain is marked in white. The source BMP file is written to an array, with black being 0, and white — 255. The output information includes arrays $B_x,\ B_y$, describing the location of the grid nodes, and array C, that stores the geometry of the source domain with the applied grid.

Research Results. The results of the algorithm demonstrate the solution to the test problem.

Input data: the source domain of the type shown in Fig. 3, as well as the calculated data $n = 12,\ m = 14,$ $i = 1,...,10,\ j = 1,...,12,\ l = 30,\ d = 3,\ \alpha = 3.$

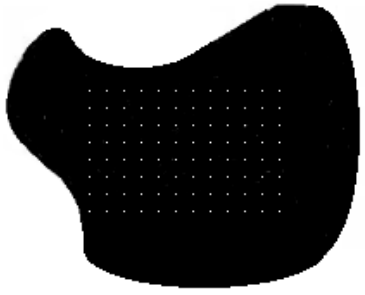


Fig. 3. Computational domain of the test problem

The initial location of the nodes is taken as the location of nodes of grid ω without adaptation to the boundary of the source domain. Visualization of the work of the function of moving interior nodes is shown in Fig. 4.

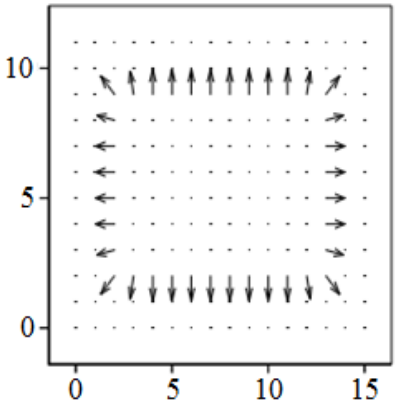
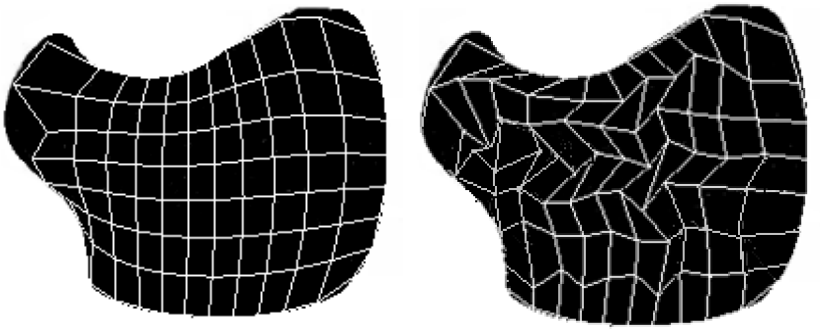


Fig. 4. Work of the function for calculating the speed of interior nodes

The result of constructing quadrangular boundary-adaptive grid ω, covering the source domain, is obtained on the basis of the presented algorithm (Fig. 5 *a*). Fig. 5 *b* shows the work of the program algorithm for the case when the border and interior nodes were not separated. A clear advantage of the grid shown in Fig. 5 *a*, consists in the fact that its cells are convex quadrilaterals. This requirement is not met for the grid of type 5 *b*.



a) *b*)

Fig. 5. Results of grid ω construction algorithm: *a*) image of grid ω, constructed when separating the border and interior grid nodes; *b*) image of grid ω, constructed without separating the border and interior nodes of grid ω

Discussion and Conclusions. A technology for constructing two-dimensional optimal boundary-adaptive grids based on the particle dynamics method is proposed. An algorithm for numerical calculation of quadrangular grids for complex configuration domains with preservation of the specified geometric features of the domain shape and boundary is developed and tested. Using the example of a test problem, the possibilities of the proposed algorithm were investigated. One of the advantages of this software implementation is the ability to automatically place nodes on the boundary of the computational domain and get convex cells. The presented approach has sufficient versatility and reliability and can be used for triangulation of the considered domains.

References

1. Downing-Kunz MA, Work PA, Schoellhamer DH. Tidal Asymmetry in Ocean-Boundary Flux and In-Estuary Trapping of Suspended Sediment Following Watershed Storms: San Francisco Estuary, California, USA. Estuaries and Coasts. 2021. URL: https://doi.org/10.1007/s12237-021-00929-y (accessed: 31.08.2021).

2. Ballent A, Pando S, Purser A, et al. Modelled transport of benthic marine microplastic pollution in the Nazaré Canyon. Biogeosciences. 2013;10(12):7957–7970. https://doi.org/10.5194/bg-10-7957-2013

3. Kirk B, Lipnikov K, Carey GF. Nested grid iteration for incompressible viscous flow and transport. International Journal of Computational Fluid Dynamics. 2003;17(4):253–262. https://doi.org/10.1080/1061856031000173635

4. Xiaoying Liu, Shi Qi, YuanHuang, et al. Predictive modeling in sediment transportation across multiple spatial scales in the Jialing River Basin of China. International Journal of Sediment Research. 2015;30(3):250–255. https://doi.org/10.1016/j.ijsrc.2015.03.013

5. Barnard PL, Jaffe BE, Schoellhamer DH (eds.). A multi-discipline approach for understanding sediment transport and geomorphic evolution in an estuarine-coastal system: San Francisco Bay. Marine Geology. 2013;345:1–326. https://doi.org/10.1016/j.margeo.2013.09.010

6. Alekseenko E, Roux B, Sukhinov A, et al. Coastal hydrodynamics in a windy lagoon. Computers & Fluids. 2013;77:24–35. https://doi.org/10.1016/j.compfluid.2013.02.003

7. Sukhinov AI. Pretsizionnye modeli gidrodinamiki i opyt ikh primeneniya v predskazanii i rekonstruktsii chrezvychainykh situatsii v Azovskom more. Izvestiya TRTU. 2006;58(3):228–235. (In Russ.)

8. Sukhinov AI, Sidoryakina VV. Development and correctness analysis of the mathematical model of transport and suspension sedimentation depending on bottom relief variation. Vestnik of Don State Technical University. 2018;18(4):350–361. https://doi.org/10.23947/1992-5980-2018-18-4-350-361 (In Russ.)

9. Sidoryakina VV, Sukhinov AI. Well-posedness analysis and numerical implementation of a linearized two-dimensional bottom sediment transport problem. Computational Mathematics and Mathematical Physics. 2017;57(6):978–994. https://doi.org/10.1134/S0965542517060124

10. Sukhinov AI, Sidoryakina VV, Sukhinov AA. Sufficient conditions for convergence of positive solutions to linearized two-dimensional sediment transport problem. Vestnik of Don State Technical University. 2017;17(1):5–17. https://doi.org/10.23947/1992-5980-2017-17-1-5-17 (In Russ.)

11. Sukhinov AI, Vasiliev VS. Precise two-dimensional models for shallow water basins. Mathematical Models and Computer Simulations. 2003;15(10):17–34. (In Russ.)

12. Carey GF, Anderson M, Carnes B, et al. Some aspects of adaptive grid technology related to boundary and interior layers. Journal of Computational and Applied Mathematics. 2004;166(1):55–86. https://doi.org/10.1016/j.cam.2003.09.036

13. Owen SJ. A Survey of Unstructured Mesh Generation Technology. In: Proc.7th Int. Meshing Roundtable. Dearborn, MI; 1998. P. 239–269.

14. Skovpen' AV. Improved algorithm for unstructured quadrilateral mesh generation. Computational Mathematics and Mathematical Physics. 2005;45(8):1506–1528. (In Russ.)

15. Krivtsov AM, Krivtsov NV. Method of particles and its application to mechanics of solids. Far Eastern Mathematical Journal. 2002;3(2):254–276. (In Russ.)

16. Belkin AA. On a modification of the method of molecular dynamics. Journal of Applied and Industrial Mathematics. 2006;9(4):27–32. (In Russ.)

17. Zheleznyakova AL, Surzhikov ST. Triangular mesh generation by molecular dynamics method. Physical-Chemical Kinetics in Gas Dynamics. 2011. Vol. 11. URL: http://chemphys.edu.ru/issues/2011-11/articles/192/ (accessed: 31.08.2021). (In Russ.)

Received 28.06.2021
Revised 19.07.2021
Accepted 27.07.2021

About the Authors:

Chistyakov, Aleksandr E., professor of the Mathematics and Informatics Department, Don State Technical University (1, Gagarin sq., Rostov-on-Don, RF, 344003) Dr.Sci. (Phys.-Math.), professor, ResearcherID: O-1507-2016, ORCID: https://orcid.org/0000-0002-8323-6005, cheese_05@mail.ru

Sidoryakina, Valentina V., Head of the Mathematics Department, Taganrog Institute Named after A. P. Chekhov, Rostov State University of Economics (RINH) branch, (48, Initsiativnaya St., Taganrog, RF, 347936) Cand. Sci. (Phys.-Math.), associate professor, ORCID: https://orcid.org/0000-0001-7744-015X, cvv9@mail.ru

Protsenko, Sofya V., postgraduate student of the Mathematics and Informatics Department, Don State Technical University (1, Gagarin sq., Rostov-on-Don, RF, 344003) ORCID: https://orcid.org/0000-0001-9656-8466, rab55555@rambler.ru

Claimed contributorship

A. E. Chistyakov: academic advising; development of the calculation program; analysis of the research results. V. V. Sidoryakina: basic concept formulation; research objectives and tasks setting; conducting a computational experiment; text preparation. S. V. Protsenko: text preparation; formulation of conclusions.

All authors have read and approved the final manuscript.