# Integrated Computational Environment for Grid Generation Parallel Technologies

Valery Il'in[1,2(✉)]

[1] Institute of Computational Mathematics and Mathematical Geophysics SB RAS, 6, Ac. Lavrentieva Avenue, Novosibirsk 630090, Russia
ilin@sscc.ru
[2] Novosibirsk State University, 1, Pirogova Street, Novosibirsk 630090, Russia

**Abstract.** This paper is devoted to the conception and general structure of the integrated computational environment for constructing multi-dimensional large grids (with $10^{10}$ nodes and more) for high-performance solutions of interdisciplinary direct and inverse mathematical modelling problems in computational domains with complicated geometrical boundaries and contrast material properties. This includes direct and inverse statements which are described by the system of differential and/or integral equations. The constructed computational grid domain consists of subdomains featuring a grid, which may be of different types (structured or non-structured); discretization at the internal boundaries can be consistent or non-consistent. The methodology of such quasi-structured meshes makes it possible to use various algorithms and codes in the subdomains, as well as different data structure formats and their conversion. The proposed technologies include grid quality control, the generation of dynamic grids adapted to singularities of input geometric data of structures and multigrid approaches with local refinements, taking into account information about the solution to be obtained. The balanced grid domain decomposition, based on hybrid programming at the heterogeneous clusters with distributed and hierarchical shared memory, supports scalable parallelization. In addition, the paper outlines the technological requirements to provide a successful long-life cycle for the proposed computational environment. In a sense, the considered development presents a stable software ecosystem (integrated grid generator DELAUNAY) for supercomputing modelling in the epoch of big data and artificial intellect.

**Keywords:** Multi-dimensional boundary value problems · Grid computational domain · Adaptive quasi-structured grids · Grid generation methods · Data structures · Scalable parallelization

V. Il'in—This work was supported by Russian Foundation of Basic Research grant 18-01-00295.


L. Sokolinsky and M. Zymbler (Eds.): PCT 2020, CCIS 1263, pp. 58–68, 2020.
https://doi.org/10.1007/978-3-030-55326-5_5

## 1 Introduction

Grid generation is an important stage of mathematical modelling of real processes and phenomena in various applications. In the ideology of the Integrated Computational Environment (ICE, [1]) for mathematical modelling, the discretization of the original problem to be solved follows after the geometrical and functional modelling and precedes the approximation and algebraic solution steps. The mesh quality defines the efficiency of the numerical methods for differential and/or integral equation systems, or corresponding variation statements. There are many papers and books on mesh constructing algorithms (see [1,9], for example). Also, many conferences, special journal issues and internet sites deal with these problems [10,11] and numerous program implementations for grid generation, commercial or free available packages, such as Netgen [18] and Gmesh [12]. Also, in gas-oil applications, PEBI (Perpendicular Bisection [20]) and Corner Point [21] are popular discretization approaches.

Grid problems include many mathematical and technological issues. Numerical approaches to the discretization of computational domains are based on the conformal and non-conformal mapping [2], variational principles [4], differential geometry methods [1], self-organizing neural networks for unsupervised learning [6], and various empirical algorithms [3,4,7,8]. Grid quality estimation and an approach to control, improve and optimize the mesh are very important questions. Also, there are many open problems in this respect, and a formulaic definition of the optimal grid has not been established yet. In a sense, the problem to construct a good mesh can be more difficult than to solve the original boundary value problem as a means of simple discretization.

From the practical point of view, the ultimate success of the grid generation consists in the mesh data structure (MDS). It must support different advanced discretization approaches: local mesh refinement based on a posteriori or/and a priori analysis of the solution to be sought, multi-scale and super-element algorithms, multigrid methodologies, as well as the domain decomposition methods (DDM), which are intended for providing the scalable parallelism and high-performance computing. Here, it is important to remark that at the further modeling stages, the parallel computations by means of synchronous solving of the auxiliary problems in subdomains with the DDM technology require to organize a distributed data structure in advance, i.e. at the step of grid generation.

We will mainly consider stationary adaptive quasi-structured grids. Their first characteristic feature means that the vertices of a computational domain coincide with the grid nodes. Also, in this case, we assume that the edges and faces of the computational domain coincide, or "almost coincide" with the grid faces, respectively. The term *quasi-structured* defines the grids which consist of the grid subdomains that each of them presents a structured or a non-structured grid. In the structured grid, for each mesh node, the number of the neighboring nodes can be computed by means of a simple formula. For each mesh point of the non-structured grid, a set of the neighboring nodes can be defined by enumeration only. Also, each grid subdomain can consist of different types of mesh elements.

We will further describe the conception of the integrated computational environment for generating a wide class of quasi-structured grids. In general, the properties of grids are defined by their MDS. Let us consider the numerical solution of the multidimensional and multi-disciplinary initial boundary value problems (IBVPs) in the computational domain with complicated contrast material properties and geometry with piecewise smooth multi-connected boundaries. We suppose that the grid generation is performed once, in advance to the general computational process. Of course, in some problems, it is necessary to construct dynamic meshes, but such computing tasks require further research.

In the inverse problems, input data are presented in the parametrized form, and we must find the solution which would provide the minimum of the prescribed objective functional, under some additional constraints. In these cases, the optimization methods include solving a set of the direct problems which use, probably, different grids. Such cases take place, for example, in the actual shape optimization problems which are connected with the geometrical and topological modifications of a computational domain.

The numerical solution of IBVPs is based on the approximation principles: finite difference, finite volume, finite element, discontinuous Galerkin methods of various order of accuracy as well as collocation and the least squares approaches. So, the grid generation stage of a mathematical simulation has an intermediate place between the geometrical modeling and approximation steps. Also, from the practical point of view, it is important to introduce a concordance between MDS and CAD products (CAE, CAM, PLM, etc.) which have an essential market and popular data formats.

In general, the grid issues include a large set of mathematical problems, algorithms, computational and program technologies which would hardly be unified in the single software product. So, we consider the concept of the integrated instrumental surrounding for the new generation of the grid constructing tool which presents not the group project but community project. The projects of FOAM, DUNE (Distributed Unified Numerical Environment), INMOST (the development of Marchuk Institute of Numerical Mathematics, RAS) and BSM [3,22–24,29] are some examples of such an integrated approach.

The considered numerical methods and technologies are implemented in the framework of the integrated grid generator DELAUNAY which is a separate part of BSM [22] and interacts with other subsystems responsible for the corresponding modeling stages just via data structures. In particular, DELAUNAY is constructed with the native CAD system HERBARIUM [25]. More exactly, the input data for grid constructing are presented by the geometrical and functional data structures (GDS and FDS) which are formed by the subsystem VORONOI responsible for constructing a continuous model of the original problem and CAD system usage. In other words, DELAUNAY presents a library of grid generators, which not only contains its own algorithms but can also efficiently use external program products. The grid construction (MDS, together with GDS and FDS) results in a discretized model for the BSM approximation stage, i.e. the CHEBYSHEV subsystem [24].

The content of this paper is as follows. Section 2 reviews a formal statement of the continuous direct and inverse problems to be solved. Section 3 describes grid objects, structures and their specifications, as well as main mesh operations which are necessary to be implemented in the numerical methods. Section 4 discusses the conception, main components and technical requirements for the integrated grid generation environment.

## 2 Formal Statement of the Interdisciplinary Direct and Inverse Problems

Since our end goal is to discretize direct and inverse IBVPs, we need to consider, from the formal point of view, specifications of the problems to be solved.

Let there be $N_\Omega$ subdomains $\Omega_k$ in the closed computational domain $\bar{\Omega}$ with the boundary $\Gamma$

$$\bar{\Omega} = \Omega \bigcup \Gamma = \bigcup_{k=1}^{N_\Omega} \bar{\Omega}_k, \tag{1}$$

and it is necessary to find the vector solution $\mathbf{u} = \{u_\mu, \quad \mu = 1, ..., \bar{m}\}$ to satisfy the differential equation

$$L\mathbf{u}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t), \ \mathbf{x} \in \Omega, \tag{2}$$

as well as the boundary and initial conditions

$$l\mathbf{u} = \mathbf{g}(\mathbf{x}, t), \ \mathbf{x} \in \Gamma, \ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}^0(\mathbf{x}), \tag{3}$$

where $\mathbf{x} = (x_1, ..., x_d)$ and $t$ are the spatial and temporal independent variables, respectively. Here $\Omega$ is an open one-connected bounded domain in the $d$-dimensional euclidean space ($d = 1, 2, 3$), $L$ and $l$ are the operators of the original equation and the boundary condition. For example, operator $L$ can have the form

$$L = A\frac{\partial}{\partial t} + \nabla B \nabla + C\nabla + D. \tag{4}$$

Here $A, B, C, D$ are the matrices, which entries depend on the time $t$ and Cartesian or some other type of the coordinate $\mathbf{x}$. In the nonlinear case, the matrix and the right-hand side $\mathbf{f}$ in (2) can also depend on the unknown solution $\mathbf{u}$. In the interdisciplinary problem, the unknown functions $\mathbf{u}_\mu$ present the fields of different physical nature: velocities, pressure, temperature, densities, etc. The boundary $\Gamma$ of the computational domain is presented by two parts, on each one of them the boundary conditions of the first (Dirichlet), the second (Neumann) or the third type (Robin) are given:

$$\mathbf{u} = \mathbf{g}_D, \ \mathbf{x} \in \Gamma_D, \quad D_N\mathbf{u} + A_N\nabla_n\mathbf{u} = \mathbf{g}_N, \ \mathbf{x} \in \Gamma_N, \tag{5}$$

where $\nabla_n$ denotes the external normal derivative on $\Gamma_N$, and $A_N, D_N$ are, in general, the matrix of coefficients, in general. Let us remark that at the joint

boundary of the contacted subdomains, some internal (interface) conditions can be given. The initial boundary value problem (IBVP) is called the interdisciplinary (or multi-physical) for the case $\bar{m} > 1$ in the sense that every scalar function $u_\mu$ corresponding to a different physical subdomain and different equations in system (1) describes various phenomena. Equation (2) can be have a different form in different subdomains $\Omega_k$. Usually it means that we have different material properties in various computational subdomains.

Equations (1)–(5) describe the direct IBVPs. But in real cases the ultimate goal of the research consists in solving inverse problems, which means, for example, the identification of the model parameters, the optimization of some processes, etc. The universal optimization approach to solving the inverse problems is formulated as minimization of the objective functional

$$\Phi_0(\mathbf{u}(\mathbf{x},\ t,\ \mathbf{p}_{opt})) = \min_{\mathbf{p}} \Phi_0(\mathbf{u}(\mathbf{x},\ t,\ \mathbf{p})), \tag{6}$$

which depends on the solution $\mathbf{u}$ and on some vector parameter $\mathbf{p}$ which is included in the input data of the direct problem. The constrained optimization is carried out under the linear conditions

$$p_k^{min} \le p_k \le p_k^{max},\ k = 1, ..., m_1, \tag{7}$$

and/or under the functional inequalities

$$\Phi_l \mathbf{u}(\mathbf{x},\ t,\ \mathbf{p})) \le \delta_l,\ l = 1, ..., m_2. \tag{8}$$

Formally, the direct problem can be considered as the state equation and can be written down as follows:

$$L\mathbf{u}(\mathbf{p}) = \mathbf{f},\ \mathbf{p} = \{p_k\} \in \mathcal{R}^m,\ m = m_1 + m_2. \tag{9}$$

There are two main kinds of optimization problems. The first one consists in the local minimization. This means that we look for a single minimum of the objective function in the vicinity of the initial guess $\mathbf{p}^0 = (p_1^0,\ ...,\ p_m^0)$. The second problem is more complicated and presents the global minimization, i. e. the search for all extremal points of $\Phi_0(\mathbf{p})$. The usual way to find the solution of the inverse problem consists in solving a set of direct problems with different parameter values.

We suppose that the subdomains $\Omega_k$ are one-connected and have no intersections. Its boundary can be presented in the form

$$\Gamma_k = \Gamma_k^{(e)} \bigcup \Gamma_k^{(h)} = \bigcup_{k' \in \omega_k} \Gamma_{k,k'}, \Gamma_{k,k'} = \bar{\Omega}_k \bigcap \bar{\Omega}_{k'} = \Gamma_{k,k'}\ k, k' = 1, ..., N_\Omega. \tag{10}$$

Here $\omega_k$ is a set of subdomain numbers which are neighboring to $\omega_k$, and $\Gamma_k^{(e)}$ and $\Gamma_k^{(h)}$ are the surface segments, which belong to external and internal parts of $\Gamma$ respectively. Formally, we can write $\Gamma_k^{(e)} = \Gamma_{k,0}$, i.e. the index "0" denotes the number of external subdomains. Each fragment $\Gamma_{k,k^0}$ presents the joint boundary

of the contacted subdomains, without any boundary condition in the statement of the original IBVP.

In general, the computational domain $\Omega$ consists of the following geometric objects: subdomains or macro-volume, $\Omega_k,\ k = k_1, ..., N_\Omega$, the vertices $V_p = (x_p, y_p, z_p),\quad p = 1, ..., N_V$, the surface segments (macro-faces) $F_q,\quad q = 1, ..., N_f$, and macro-edges (curvilinear fragments) $E_l,\ l = 1, ..., N_f$. For the analytical description of the geometric primitives the global and local coordinate systems are defined (Cartesian, cylindrical or spherical). In many cases, it is convenient to describe the line or surface segments in the parametric form:

$$x = x(\tau), y = y(\tau),\ or\ x = x(\tau^{(1)}, \tau^{(2)}), y = y(\tau^{(1)},\ \tau^{(2)}), z = z(\tau^{(1)}, \tau^{(2)}). \tag{11}$$

For particular situations, there are a lot of different approaches, to describe geometric objects. An advanced simulation system should have various possibilities for efficient representations and the converters of the input formats.

The connections between geometric objects are described by means of incident matrix with bit entries:

$$M_{VE} = \{m_{i,j}^{VE} :\ i = 1, ..., N_V;\ j = 1, ..., N_E\} - \text{(vertices – edges)},$$
$$M_{EF} = \{m_{i,j}^{EF} :\ i = 1, ..., N_E;\ j = 1, ..., N_F\} - \text{(edges – faces)},$$
$$M_{F\Omega} = \{m_{i,j}^{F\Omega} :\ i = 1, ..., N_F;\ j = 1, ..., N_\Omega\} - \text{(faces – subdomains)}.$$

In total, configuration information on the computational domains is included in geometric data structure (GDS). The functional description of the IBVPs (the system of equations, to be solved. the boundary conditions, and their coefficients) is presented in the functional data structure (FDS), see [6] for details. Generally speaking, this input (for DELAUNAY) information presents the continuous model of the problem to be solved. It is formed by the VORONOI system, which is responsible for the geometric and functional modeling in BSM.

Let us remark, that some of the input data can be given in parametrized form, in order to solve the inverse, or the set of direct problems. In such cases the additional information to organize the computational experiments should be given.

## 3 Grid Objects, Operations and Data Structure

In this section, we define the main specifications of the grids, the operations with the mesh objects. as well as principles of the constructing the mesh data structure.

### 3.1 Main Grid Notions and Specifications

The discretization of the computational domain $\Omega$ consists in the construction of the grid computational domain $\Omega^h$, which includes the following mesh objects, similar to the "macro-objects" of $\Omega$:

grid subdomain $\bar{\Omega}_k^h$: $\bar{\Omega}^h = \bigcup\limits_k \bar{\Omega}_k^h$, $\bar{\Omega}_k^h = \Omega_k^h \bigcup \Gamma_k$, $k = 1, ..., N_\Omega^h$, $\Omega_k^h \approx \Omega_k$,

grid subdomain boundaries: $\Gamma_{k,k'}^h = \bar{\Omega}_{k'}^h \bigcap \bar{\Omega}_k^h$, $\Gamma_e^h = \bigcup\limits_k \Gamma_{k,0}^h$, $\Gamma_k^h \approx \Gamma_k, k, k' = 1, ..., N_\Gamma^h$,

grid nodes: $V_p^h = V_{e,e'}^h \equiv \bar{E}_e^h \bigcap \bar{E}_{e'}^h = (x_p, y_p, z_p)$, $V_{l,l',l''} \equiv \bar{F}_l^h \bigcup \bar{F}_{l'}^h \bigcup \bar{F}_{l''}^h, p = 1, ..., N_V^h$,

grid edges: $E_s^h = E_{l,l'}^h \equiv \bar{F}_l^h \bigcap \bar{F}_{l'}^h$, $s = 1, ..., N_E^h$,

grid faces: $F_l^h = F_l^h = F_{m,m'}^h \equiv \bar{T}_m^h \bigcap \bar{T}_{m'}^h$, $l = 1, ..., N_F^h$,

grid (finite) volumes, or elements: $T_m^h, m = 1, ..., N_T^h$, $\bar{\Omega}^h = \bigcup\limits_m \bar{T}_m^h$,

where $N_\Omega^h, N_\Gamma^h, N_V^h, N_E^h, N_F^h, N_T^h$ means the total numbers of the grid subdomains, boundaries, nodes, edges, faces, and volume elements respectively.

The grid objects have the following topological connections. The mesh domain and subdomains are the unions of the corresponding elements. The mesh faces are the joint boundaries of the contacted finite elements. The grid edges present the intersection of the neighboring faces. The nodes are the joint points of the intersected edges or faces. Similar to macro-object, we can define rectangular matrice of the different types, which characterize the connections between the grid objects. For example, a node-edge matrix is $M_{V,E}^h = \{m_{i,j}^{V,E}, i = 1, ..., N_N^V, j = 1, ..., N_E^h\}$ where each bit entry $m_{i,j}^{V,E} = 1$ if the $j$-th grid edge is connected to the $i$-th grid node, and $m_{i,j}^{V,E} = 0$ otherwise. In a similar way, we can define a node-volume matrix $M_{V,T}^h = \{m_{i,j}^{V,T}, i = 1, ..., N_V^h, j = 1, ..., N_T^h\}$, an edge-force matrix $M_{E,F}^h = \{m_{i,j}^{V,T}, i = 1, ..., N_E^h, j = 1, ..., N_F^h\}$, as well as other types of the matrix: $M_{S,T}^h, M_{V,F}^h$, $M_{F,E}^h = (M_{E,F}^h)^T$, etc. All types of grid objects have their own global and local (on subdomains) numbering, as well as the functions for their remembering. The geometric and topological specification of the object can be formed via the MDS. It also includes necessary connections with the functional data structure for defining the type of an equation and the values of its coefficients in each element $T_m^h$, as well as for describing the boundary conditions at the grid faces $F_l^h$.

### 3.2 Grid Operations and Data Structure

Advanced numerical methods of mathematical modelling are based on the complicated matrix transformations, which are connected with the graph structure of a grid. The main consuming step of the numerical simulation is to solve a system of linear algebraic equations (SLAEs) which is repeated many times if the original problem is inverse or non-stationary and non-linear. The modern large SLAE can be characterized as a sparse matrix of order $10^9$ or higher and the medium number of non-zero entries which are about 100 in each row.

The arithmetic operations must be done in the standard double precision format (64 bits, or 8 bytes).

The high performance solution in such cases is provided by means of scalable parallelizm based on the domain decomposition approaches. The advanced numerical tools include the two-level multi-preconditioned iterative methods in the Krylov subspaces. At the upper level, the distributed version of the additive block Swarz algorithm is implemented with the help of the MPI (Message Passing Interface between cluster nodes) functions. The lower level consists of the synchronous solution to the auxiliary SLAEs in subdomains using the multithread computing (Open MP technologies) and vectorization of the operations by AVX instruments. Also, some auxiliary algebraic subsystems can be solved by means of special algorithms on the super-fast graphic accelerators (GP GPU or Intel Phi).

For the real mathematical problems, which are solved on a non-structured grid, an important circumstance for implementation of the algebraic methods is that the matrices are presented in the compressed sparse formats, such as CSR, because the allocation of the non-zero matrix entries in the rows corresponds to the connection of the respective grid nodes to their neighbors. In this case, only non-zero matrix entries and references between them are saved. It is obvious that such forced technologies make access to matrix values in memory too slow and expensive.

The considered approach defines the matrix portrait which is isomorphic, in a sense, to the graphic structure of a non-structured grid. So, the algebraic data structure (ADS) which is the base for iterative algorithms is fairly similar to the MDS, and the CSR format can be the base for the grid data structure because each matrix row corresponds to a mesh node or another primitive.

In some cases, the situation is more complicated. If we solve an interdisciplinary problem which is described by a system of partial differential and/or integral equations, then after discretization in each grid node, several unknown variables can be defined, and we have to generalize the CSR format to the block one (BCSR).

Because of a big volume of the grid data structure, the domain decomposition techniques must be implemented at the mesh generation stage. It should include a description of the grid computational subdomains and a description of the MDS informational arrays in the memory of the corresponding cluster node. This means that the corresponding approximation stage will be done efficiently in parallel. From the algorithmic point of view, the decomposition is performed in two steps. The first one consists in disassembling the mesh computational domain into parts without intersections. At the second stage, the grid subdomains are extended to a necessary number of mesh layers. By the end of this stage, we must obtain the distributed MDS for decomposition with parameterized overlapping of the subdomains. At the following modeling steps, it will be the base for the parallel implementation of the total computational process.

Also, at this step, the multigrid data structure is performed if needed. Theoretically, this approach provides the optimal in order algebraic solvers. The complementation technology requires forming a hierarchical data structure for

a set of embedded grids. Computational schemes in these cases are rich in various numerical operations: pre-smoothing and post-smoothing, restriction and prolongation, course grid correction, etc. Automatic construction of such algorithms is a highly intellectual problem, and the grid transformation support is very important in such a challenging task.

In recent decades, the advanced approaches have appeared for solving multi-scale problems based on super-element technologies and constructing the immersion type meshes, which are aimed at the application of a special type of non-polynomial basis functions. In such cases, a special kind of MDS is required to take into account the geometric details of the under-cell scale.

The above brief review of the grid construction problems reflects many mathematical issues. In particular, there are valuable computational geometry tasks, see [26,27]. Automatic construction of mesh algorithms and their mapping on the computer architecture present a challenging intellectual problem of great practical importance. In fact, we need to create a mathematical knowledge base with expandable sets of computational methods and technologies. A prototype of such a base is presented in [28].

## 4 ICE Structure and Technical Features

We will consider the ICE for the grid generation as a DELAUNAY subsystem of the Basic System of Modeling (BSM [15]). In fact, it presents a library for grid generation and transformation algorithms, as well as a set of system instruments for data processing and for supporting the external and internal communications. Because of the rich functionality and a large volume of the ICE code, the following technical requirements must be provided in order to ensure an efficient long life cycle of the proposed applied program product for the grid generation.

- Flexible extendability of the IBVPs to be solved by means of the BSM as well as a manifold of applicable numerical methods and technologies for the grid generation without program limitations on the degrees of freedom (d.o.f.) and the number of computer nodes, cores and other hardware.
- Adaptation to the evolution of computer architectures and platforms; automatic mapping of the algorithm structures onto hardware configuration.
- Compatibility of the flexible data structures with the conventional formats to provide the efficient re-use of the external products for grid generation, which presents great intellectual potential.
- High-performance of the developed software, scalable parallelism based on the hybrid programming tools, minimization of expensive communications, and code optimization on the heterogeneous multi-processor supercomputers with distributed and hierarchical shared memory.
- Multi-language interaction and consistency of various program components, enabling working contacts for different groups of developers, as well as creating friendly interfaces for the end users of different professional background. The considered integrated program environment must have valuable system

support, aimed at the maintenance, collective operations, information security and further high productive development. The corresponding intelligent instruments and big data support must constitute the infrastructure to support the following system procedures.

- Automatic verification and validation of the codes, as well as testing and comparative experimental analysis of the algorithms.
- Generating the multi-variant program configurations for a specific application by assembling the functional modules.
- Data structures control and transformations to provide the internal component compatibility and re-using of the external products.
- Deep learning of the grid generation issues: creating a knowledge database on mesh constructing methods, grid quality analysis, automatic selection of the available algorithms based on cognitive technologies.

## 5 Conclusion

The conception, main components and data structure of the integrated computational environment for constructing a wide class of multi-dimensional quasi-structured grids have been considered. The objective of the development is to provide high-productive program tools for the modern supercomputers with a long life cycle aimed at efficient support of the mathematical modelling in various applications.

## References

1. Il'in, V.P.: Mathematical Modeling, Part I: Continuous and Discrete Models. SBRAS Publ, Novosibirsk (2017). (in Russian)
2. Godunov, S.K., Romenski, E.L., Chamakov, G.A.: Grid generate ioncomplicated domains by qusuconformal mapping. Trudy IM SBRAS Novosibirsk **18**, 75–84 (1990). (in Russian)
3. Terekhov, K., Vassilevski, Y.: Mesh modification and adaption within INMOST programming platform. In: Proceedings of the 9th International Conference NUMGRID 2018, Moscow, LNCSE, vol. 131, pp. 243–255 (2018)
4. Garanzha, V.A.: Variational principles in grid generation and geometric modelling. Numer. Lin. Alg. **11**, 535–563 (2003)
5. Bronina, T.N., Gasilova, I.A., Ushakova, O.V.: Algorihms fort hree- dimensional structured grid generation. Zh. Vychisl. Mat. Met. Fiz. **43**, 875–883 (2003). (in Russian)
6. Fritzke, B.: Growing cell structuring - a self-organizing networks for unsupervised learning. Neural Netw. **7**(9), 1441–1460 (1994)
7. Ivanenko, S.A.: On the existence of equation for description of the classes of non-singular curvilinear coordinates on arbitrary domain. Zh. Vgchisl. Mat. Phys. **42**, 47–52 (2002). (in Russian)
8. Liseikin, V.D.: Grid Generation Methods. SC. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57846-0
9. Il'in, V.P.: DELAUNAY: technological media for grid generation. Sib. J. Industr. Appl. Math. **16**, 83–97 (2013). (in Russian)

10. Funken, S.A., Schmidt, A.: Ameshref: a Matlab-toolbox for adaptive mesh refinement in two dimensions. In: Proceedings of the 9th International Conference NUMGRID 2018, Moscow, LNCSE, vol. 131, pp. 269–279 (2018)
11. Li, S.: Mesh curving refinement based on cubic bezier surface for high-order discontinuous Galerkin methods. In: Proceedings of the 9th International Conference NUMGRID 2018, Moscow, LNCSE, vol. 131, pp. 205–216 (2018)
12. Ushakova, O.V. (ed.): Advances in Grid Generations. Nova Sci. Publ, New York (2007)
13. Zint, D., Grosso, R., Aizinger, V., Kostler, H.: Generation of block structured grids on complex domains for high performance simulation. In: Proceedings of the 9th International Conference NUMGRID 2018, Moscow, LNCSE, vol. 131, pp. 87–99 (2018)
14. Khademi, A., Korotov, S., Vatne, J.E.: On equivalence of maximum angle conditions for tetrahedral finite element meshes. Performance Simulation. In: Proceedings of the 9th International Conference NUMGRID 2018, Moscow, LNCSE, vol. 131, pp. 101–108 (2018)
15. Gartner, K., Kamenski, L.: Why do we need voronoi cells and delaunay Meshes? In: Proceedings of the 9th International Conference NUMGRID 2018, Moscow, LNCSE, vol. 131, pp. 45–60 (2018)
16. International Meshing Roundtable: www.imr.sandia.gov/18imr
17. Internat. J. for Numer. Math. in Eng., 58(2), special issue "Trends in Unstructured Mesh Generation" (2003)
18. Schoberl, J.: Netgen-an advancing front 2D/3D-mesh generator based on abstract rules. Comput. Visualizat. Sci. **1**, 41–52 (1997)
19. Geuzain, C., Remacle, J.-F.: Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Num. Methods Eng. **79**, 1309–1331 (2009)
20. Meng, X., Duan, Z., Yang, Q., Liang, X.: Local PEBI grid generation method for reverse faults. Comput. Geosci. **110**, 73–80 (2018)
21. Ponting, D.K.: Corner point geometry in reservoir simulation. In: Proceedings of the 1st European Conference on Mathematics in Oil Recovery, Cambridge, pp. 45–65 (1989)
22. DUNE.URL: http://www.dune-project.org. Accessed 15 Jan 2016
23. Il'in, V.P., Gladkih, V.S.: Basic system of modeling (BSM): the conception, architecture and methodology. In: Proceedings of the International Conference on Modern Problems of Mathematical Modeling, Image Processing and Parallel Computing. (MPMMIP & PC-2017) DSTU Publ. Rostov-Don, pp. 151–158 (2017). (in Russian)
24. Ilin, V.P.: The Conception, Requirements and Structure of the Integrated Computational Environment. In: Voevodin, Vladimir, Sobolev, Sergey (eds.) RuSCDays 2018. CCIS, vol. 965, pp. 653–665. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05807-4_56
25. HERBARIUM. http://tflex.ru/about/publications/detail/index.php?ID=3846
26. Cottrell, J., Hughes, T., Bazilevs, Y.: "Isogeometric Analysis", Towards Integration of CAD and FEA. Wiley, Singapore (2009)
27. Delfour, M., Zolesio, J.-P.: Shape and Geometries. Metrics, Analysis, Differential Calculus, and Optimization, SIAM Publication, Philadelphia (2011)
28. ALGOWIKI. https://algowiki-project.org
29. Ilin, Valery: On an Integrated Computational Environment for Numerical Algebra. In: Sokolinsky, Leonid, Zymbler, Mikhail (eds.) PCT 2019. CCIS, vol. 1063, pp. 91–106. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28163-2_7