Artificial Intelligence Problems in Mathematical Modeling

Valery Il'in[1,2(✉)]

[1] Institute of Computational Mathematics and Mathematical Geophysics SBRAS, Novosibirsk, Russia
ilin@sscc.ru
[2] Novosibirsk State University, Novosibirsk, Russia

Abstract. The artificial intelligence problems in creating the integrated computational environment (ICE) for the mathematical modeling on the modern supercomputers are considered. The mathematical tasks to be solved present the direct and inverse interdisciplinary problems which are described by the systems of partial differential and/or integral equations in the multi-dimensional computational domains with real complicated multiscale geometry and contrast material properties. The conception of ICE consists in developing the instrumental media to support all main stages of large scale computational experiments, with a long life cycle of the products. The technical requirements for ICE include the flexible extension of the mathematical models and methods, the adaption to evolution of the computer platforms, the reusing external program products, and coordinated participation in the project of the different groups of developers. The objectives of intelligent tools consist in automatic construction of algorithms as well as generating and executing the applied software packages for the end users in particular domains.

Keywords: Artificial intelligence · Integrated computational environment · Mathematical modeling · Numerical methods · Ontology of designing · Knowledge base · Automatic construction of algorithms

1 Introduction

The dramatic quantitative and qualitative growth of supercomputers, which in the coming years should mark the emergence of exaflops machine, as well as the predicted breakthrough in the advanced countries in all kinds of scientific and technological spheres means not only the transition to a principally new level of obtaining fundamental and applied knowledge, but also their application in all areas of human activity. A special role in the progress is given to mathematical modeling, which, on the one hand, holds an intermediate position between theoretical and experimental scientific studies, and on the other - becomes an indispensable attribute of the study and/or control of production, natural, and

Supported by the RFBR grants N 16-29-15122 ofi-m and N 18-01-00295.

V. Voevodin and S. Sobolev (Eds.): RuSCDays 2019, CCIS 1129, pp. 505–516, 2019.
https://doi.org/10.1007/978-3-030-36592-9_41

social processes. The principal point here is to solve on a supercomputer such supertasks, which in terms of mathematical complexity and resource intensity could not be imagined a decade ago.

As is noted in [1] the key problem of the world computing community is the development in a short time of huge amounts of the new generation of software, whose creation is possible only with the extensive cooperation. With regard to scientific application software this means the transition from conventional problem-oriented software packages of the ANSYS, FeniCS [2,3] type, libraries of universally or method-oriented algorithms (see, for example, NETLIB [4] or MKL INTEL [5]) and specialized tools, to integrated computational environments (ICE) designed to support the computational stages of a large-scale supercomputer experiment. Such examples are the projects DUNE, INMOST and BSM, see [6–8].

In general, the models of the processes and phenomena under consideration are interdisciplinary direct and inverse multidimensional initial-boundary value problems (IBVPs) described by the systems of differential and/or integral equations of different types in classical or generalized formulations, for which the real source data are characterized by complex multi-scale geometry of computational domains with piecewise smooth multi-connected boundary surfaces and contrast material properties. The current requirements to the accuracy and resolution of the applied mathematical models impose strict conditions on the stability, convergence, conservative and robustness of rapidly developing computational methods.

With all the variety of mathematical problems and algorithms, the organization of large-scale computational experiments is carried out by clearly structured technological stages: geometric and functional modeling associated with the description of the initial data of the problem to be solved, the generation of spatial-temporal adaptive grids, i.e. discretization of continuous initial statements, approximation of given differential and/or integral equations, the solution of the resulting systems of linear or nonlinear algebraic grid problems, optimization approaches for the inverse problems with constrained minimization of objective functionals, post-processing and visualization of the results of calculation, computational process control and decision-making according to the results of modeling, [9]. All the described stages are implemented by the corresponding rather autonomous method-oriented software subsystems, which can be developed independently and interact through the formed data structure. (geometric, functional, grid, algebraic, etc.) in the coordinated formats, which can be converted and interact with the external word. Each subsystem present the integrated program environment in the own methodological domain and the whole their totally represent the functional content of BSM.

In the last decades, despite the active dynamics of development of multiprocessor computing systems (MPS), their architecture can be considered fairly stable and represent heterogeneous clusters of multifunctional nodes with distributed memory, each of which contains several multi-core processors (CPU) with a shared hierarchical memory, as well as graphics accelerators such as

GPGPU or INTEL Phi. The scalable parallelization of such MPS is achieved by means of hybrid programming with the passing of messages between different MPI - processes, organization of multithread computing using OpenMP systems, as well as operations vectorization by using AVX system. A typical situation is also the solution of problems in remote access via the Internet to a large general access supercomputer center based on the cloud computing.

It is important to note that now the world has a huge number of application and system products, both public and commercial, representing an invaluable intellectual potential that can and should be effectively used in new developments which should save many hundreds of man-years of programming. The creation of the ICE is of multi-purpose insignificance, and its architecture is defined from the following conceptual technological requirements: flexible expansion of a set of considered models and the applied algorithms, adaptation to evolution of supercomputer platforms, effective reusing of external software products, high modern mathematical characteristics and performance of computing methods and technologies, the coordinated participation of various groups of developers, intelligent output and output interfaces for the comfortable work of the end users with various professional backgrounds.

These characteristics of the development are designed for a long life cycle of ICE and its reasonable for demand various industries, which in general should ensure its competitiveness and economic success, which is an important factor, because of the global nature of the project.

The use of ICE is oriented in two directions. The first one is the automation of algorithms construction and their mapping on the architecture of a supercomputer, aimed at a significant increase in the programming productivity. More specifically, this part of the activity is the development of the functional content (models, methods and technologies) of BSM. The second direction and purpose of ICE is the automated design of the high-performance applied software packages or complexes with high-quality execution characteristics comparable to professional compilers or operating systems.

The creation of scientific ICE places high demands on the level of intelligence of the project, and the concept of its integration actually means the transition from artisan to industrial design and implementation of large applications, whose continued existence implies an active support, development and operation, which also involves a variety of user-friendly interfaces. Note that a number of intellectual issues of the methodology of mathematical modeling and creation of a new type of software for it are considered in [10,11].

This paper is constructed as follows. Paragraph 2 describes the tasks and approaches to solving the problem of intellectualization of the main technological stages of modeling. The third section contains the analysis of various methodologies and tools for of implementation of cognitive and ontological principles in the framework of ICE, and in conclusion discusses the current open issues of cognitive processes in basic research and optimal control in applied fields.

2 Features of Intellectualization of Technological Stages of Modeling

With all the variety of problems of mathematical knowledge mining, they are structurally decomposed in steps of fairly well-established methodologies.

2.1 Tasks of Geometric and Functional Modeling

The first stage of the computational experiment, of course, is statement of the task, preprocessing and analysis of the initial data. For a direct initial-boundary value problem, the description includes the equation to be solved, the computational domain, boundary and initial conditions:

$$L\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \bar{\Omega} = \Omega \bigcup \Gamma, \quad 0 < t \leq T < \infty, \tag{1}$$

$$l\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{x},\ t), \quad \boldsymbol{x} \in \Gamma, \quad \boldsymbol{u}(\boldsymbol{x},\ 0) = \boldsymbol{u}^0(\boldsymbol{x}). \tag{2}$$

Here, the unknown solution $\boldsymbol{u}$ is a vector function, L is a differential and/or an integral operator (possibly of the matrix type) defined in the computational domain Ω with the boundary Γ, $\boldsymbol{x}$ and t – spatial and temporal independent variables, l is the operator of boundary conditions, and given functions $f(\boldsymbol{x}, t)$ and $g(\boldsymbol{x}, t)$ can also depend on the original solution in a nonlinear way. The computational domain (bounded or unbounded) can consist of various subdomains Ω_j with the corresponding inner or outer boundary segments Γ_j^i, Γ_j^e:

$$\bar{\Omega} = \bigcup \bar{\Omega}_j,\ \Gamma = \Gamma^e \bigcup \Gamma^i,\ \Gamma^i = \bigcup \Gamma^i_{j,\ k} = \bigcup (\bar{\Omega}_j \bigcap \bar{\Omega}_k). \tag{3}$$

It is important to note that different types of equations can be solved in different subdomains, and different boundary conditions can be set on different surface segments.

Formal descriptions (1)–(3) cover a huge class of mathematical modeling problems, including systems of Maxwell's equations for electromagnetism, the Navier-Stokes equations – for hydro-gasdynamics, the Lame equations – for elasticity, Darcy equations – for the filtration problems, etc. In addition, initial statements can be given in classical or generalized (variational) formulations.

On the other hand, similar mathematical concepts may relate to completely different application areas or industry sectors: energy, engineering, biology, geophysics, chemistry, ecology, medicine, agriculture, etc.

In real cases the ultimate goal of the research consists in solving not direct but 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

$$\Phi_0(\boldsymbol{u}(\boldsymbol{x},\ t,\ \boldsymbol{p}_{opt}) = \min_{\boldsymbol{p}} \Phi_0(\boldsymbol{u}(\boldsymbol{x},\ t,\ \boldsymbol{p})), \tag{4}$$

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

$$p_k^{min} \le p_k \le p_k^{max}, \quad k = 1, \ldots, m_1, \tag{5}$$

and/or under the functional inequalities

$$\Phi_l \boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{p}) \le \delta_l, \quad l = 1, \ldots, m_2, \quad \boldsymbol{p} = \{p_k\} \in \mathcal{R}^m, \quad m = m_1 + m_2. \tag{6}$$

There are two main kinds of the 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 $\boldsymbol{p}^0 = (p_1^0, \ldots, p_m^0)$. The second problem presents the global minimization, i. e. the search for all extremal points of $\Phi_0(\boldsymbol{p})$, which requires the solution of many direct problems.

In the recent decades, such a scientific direction has been developed dramatically fast in many applications, based on the so-called surrogate optimization (see [12] and the literature therein). The last mentioned approach corresponds to the situation, when the run time for computing one value of the objective function is too expensive and requires several hours or more. In such a case, the design, or planning of numerical experiments is very important, as well as using special methods for approximation of the investigated functionals.

As can be seen even at first glance at formulas (1)–(6), the mathematical formulation of high-tech problem is a complex logical structure. The initial information is divided into the two main groups - functional and geometric. The first one includes the types of equations to be solved, representations of their coefficients, boundary and initial conditions, descriptions of minimized functionals and constraints for inverse problems, etc. The geometric data must uniquely describe the computational domain, including its subdomains, boundary fragments, edges (surface intersections), and nodes that are the intersection points of the edges.

The initial information should first of all be tested for its completeness and consistency, for example, to meet the conditions of existence, uniqueness and correctness of the solutions, as well as for the presence of any features or singularities that must be taken into account when using numerical algorithms.

It is obvious that the in-depth study of specific processes and phenomena is an important experimental study with multiple calculations and variation of the initial data of the problem. This gives rise to various problems of geometric modeling of functional objects. In particular, the relevant areas of analysis arise in the inverse problems related to the optimization of geometric and logical configurations, metric transformation, and discrete exterior differential forms, isogeometric analysis, etc., see [13,14]. Note that the end result of this stage should be interconnected geometric and functional data structures (GDS and FDS), completely and unambiguously determining the mathematical problem and specific source data for its solution.

2.2 The Intellectialization of the Grid Generation

Based on geometric and functional data structures we can realize the grid generation, which presents an important and resources-consuming stage of the numerical solution of the multi-dimensional problems. In the world software market, there are many available (free of charge) and expensive commercial codes, but the problem of constructing optimal or even "good" 3-D grids in the complicated computational domain is far enough from its final solution. Moreover, it is not easy to define the concept of an optimal or a good mesh, and there are many different quantitative characteristics of the grid quality.

Formally, the mesh data structure is similar to the GDS and includes the following objects: the grid computational domain $\bar{\Omega}^h = \Omega^h \bigcup \Gamma^h$ with the boundary Γ^h, the grid subdomains $\bar{\Omega}_k^h = \Omega_k^h \bigcup \Gamma_k^h$ with the corresponding faces Γ_k^h, the grid edges E_p^h and the nodes P_q^h. The conventional approach to the discretization of the computational domain consists in constructing an adaptive grid. This means that the vertices P_q, the edges E_p and the surfaces Γ_k of the computational domain Ω should coincide with the corresponding grid nodes P_q^h, the grid edges E_p^h and the faces Γ_k^h.

There are many kinds of the grids with different types of finite elements, with various distributions of the meshsteps h, local refinement and multi-grid approaches included. Also, there are a lot of algorithms for the mesh generation, which are based on the frontal principles, on conformal or quasi-conformal transformations on the differential geometry and various metrics, see review in [15]. In general, the grid computational domain consists of the grid subdomains which can have different types of the finite volumes, and grids in different subdomains can be constructed by different algorithms.

In a sense, the grids considered present a two-level hyper-graph at the macro- and micro-, or mesh, levels. In the technological sense, the final result of the grid generator should be the mesh data structure (MDS) with full mapping of the input geometric and functional data onto the micro (mesh) level. In particular, all inter-connections between grid objects should be strictly defined. Also, the affiliation of each finite volume T_j^h, grid face Γ_k^h into the corresponding subdomains Ω_k and the boundary surface segment Γ_p must be given.

The principles of constructing the library DELAUNAY are described in [16]. In fact, it presents the integrated instrumental media for supporting a considered class of problems, based on original algorithms, as well as on re-using the external codes (there are popular free available mesh generators NETGEN, GMESH, TETGEN, for example). In the world "grid developer community", there are several popular grid formats, and the subsystem DELAUNAY should include the corresponding data converters with the MDS. At this stage one of important operations includes the decomposition of the grid computational domain into grid subdomains, when the number of mesh points is too large.

There are many numerical approaches to construct the qualitative or optimal grids, but, in general, these mathematical questions are open yet. Also, we do not consider here resource-consuming problems of generating the dynamic meshes which are changed during the computational process.

Formally, the grid can be presented as an indirect graph, and many useful software tools for graph transformation (METIS or its parallel version parMETIS, for example) can be used efficiently for mesh generation and its improvement.

As can be seen from the above algorithmic aspects, the generator of "good" grids must solve many highly intelligent problems. The quality criteria for the discretization of the computational domain have many quantitative characteristics, on which the accuracy, reliability, resource consuming and efficiency of modeling largely depend. An important property is the adaptability of the grid to the geometric and material features of the mathematical formulation. In this case, the points-vertices of the boundary must be nodes in the mesh, and macro-edges and macro-facets of the subdomain should be well approximated by the relevant grid objects. Of great importance is the refinement of the grid in the vicinity of the singularity of the solution associated with the features of the boundary of the domains or coefficients of the original equation, whose nature can be determined based on a priori or a posteriori analysis.

2.3 Automation of Approximation Procedures

It can be said that this stage carries the greatest theoretical load, since it is here where the orders of accuracy of the approximate structures and spectral properties of the resulting systems of algebraic equations are laid, which basically determine the efficiency and the robustness of a constructed computational model. The original data for the approximation of the initial boundary value problem is the interconnected grid, geometric and functional data structures (MSD, GSD, FSD) formed at the previous stages.

The most popular approaches are finite difference, finite volume, finite element, and discontinuity Galerkin methods (FDM, FVM, FEM, and DGM, see [9] and reference therein for example). The advanced theoretical and applied mathematical results have profound foundations and technologies for constructing and justification of high order accuracy numerical schemes for complicated IBVPs with real data. The implementation of such algorithms on the non-structured grids is not simple, and the tools for automatic construction of the scheme are very useful for such problems (so called computer algebra tools are implemented in the program package FEniCS [3] and the language PYTHON, for example). A very powerful approach here is based on the element technology with computing the local matrices and assembling the global matrix, which provide the "natural" parallelization and easy programming of the algorithms.

The concept of the integrated operating environment for the methods of approximation of the multi-dimensional IBVP is presented in the library CHEBYSHEV [17] based on original algorithms and re-using the external software. The end result of this subsystem consists in the algebraic data structure (ADS) which presents the original problem to be solved at a discrete level. To provide the necessary accuracy the obtained systems of linear algebraic equations (SLAEs) should have very large dimensions (10^8 and more) and sparse

matrices. To save such systems in the memory, the conventional compressed formats are used, Compressed Sparse Row (CSR), for example. For the big degrees of freedom, the distributed versions of the CSR are used, i.e. a matrix is divided into block rows, and each one is placed in the corresponding MPI-process.

The concept of the library "approximators" as a universal tool environment for different types of original equations for different types of grids and methods of construction of discrete approximations, unlike other computational stages, has no analogues and allows the use of modern intelligent tools for automation of analytical tools with complex formula transformations to build algorithms.

2.4 Automatic Construction of the Algebraic Algorithms

The most resource-consuming stage of mathematical modeling is a numerical solution of large sparse SLAEs, because the volume of arithmetic operations grows nonlinearly, when the number of unknowns increases. Fortunately, the computational algebra is one of the most progressive parts of numerical mathematics, both in algorithmic and in technological senses (see [18–20] and the literature, cited therein). In particular, there are many applied software packages and libraries with algebraic solvers which are free accessible (with parallel algorithms in PETSc and PARMS included). The main approaches to solve large sparse SLAEs are based on the preconditioned iterative methods in the Krylov subspaces. The scalable parallelism is provided in the framework of the two-level iterative domain decomposition methods (DDM) by means of hybrid programming with using MPI tools for the distributed memory of the heterogeneous cluster MPS, multi-thread computing on the shared memory of the multi-core CPUs, vectorization of operations by means of the AVX system, as well as fast computation on the graphic accelerators (GPGPU or Intel Phi).

The grid computational domain is decomposed into grid subdomains with parametrized overlapping and different interface conditions which realize some Poincare - Steklov operators on the interior boundaries. Algebraically, the external iterative process presents the multi-preconditioned generalized minimal residual (GMRES) or a semi-conjugate residual (SCR) algorithm, based on the parallel block Schwarz - Jacobi method, coarse grid correction, deflation and/or augmentation procedures, restarted least squares approaches, and an advanced low-rank approximation of the original matrix. At each external iteration, solving the auxiliary SLAEs in subdomains is implemented synchronously by means of direct or iterative algorithms, with various internal preconditioning matrices (for example, defined by incomplete LU - factorization, Alternating Direction Implicit Schemes, Cimmino method, etc.). Here, it is very important to provide the balancing domain decomposition and minimization of the communication time, by means of buffering data to be transfered between subdomains.

The described parallel methods are realized in the framework of the library KRYLOV [19] which presents the integrated algebraic environment, based on the original algorithms and efficient re-using the external products. The robust matrix-vector operations and other algebraic tools from the library MKL Intel are applied in KRYLOV in a productive manner.

If the original continuous problem is nonlinear, then after its discretization we will have a system of nonlinear algebraic equations (SNLAEs). In these cases, the quasi-linearization process is applied, based on the Newtonian type of iterations, and at each of such steps the linear equations are solved.

Of course, the high performance and scalable parallelism are the most important for very resource consuming tasks which are presented by interdisciplinary inverse problems which require the multiply solutions of large direct problems. Some technological approaches for this issues are presented in [21].

As can be seen from the presented review of the modeling stages, the computational process is a complex scheme, since in general, the calculations include multiple nested cycles with modification of the initial data, with reconstruction of grids by recalculation of approximations, with numerical integration by time steps for non-stationary problems, with the solution of systems of nonlinear and linear algebraic equations. Therefore, the planning and control of a machine experiment is also an intellectual problem, which is based on the analysis of the preliminary results of the calculation.

As for parallel post-processing and visualization of the numerical results, this resource consuming stage can be realized by means of existing power tools (the cluster version of PARAVIEW [22], for example).

3 Methods and Tools of Artificial Intelligence for Mathematical Modeling

Recent decades have been marked by major scientific and practical achievements in the field of artificial intelligence. Well-known a success means in computer chess, the game of go and in other games, in robotics, in neural networks, in pattern recognition, in machine translation of texts, etc. Against this background, there are almost no breakthroughs in programming methods for modeling, despite the fact that computers were originally invented specifically for large calculations. Although the algorithmic languages of programming and compilers, operating systems, various instrumental tools, numerous applied developments are actively implemented, they are mainly focused on achieving the high code performance on the MPS. As a result, with a rapid growth of computer performance, there appears a global programming crisis, when the growth of developers' productivity is far behind the computer evolution rate. Obviously, this situation requires a change in the programming paradigm, primarily through the integration of development and intellectualization of technologies.

To date, there are quite many published works on semantic modeling [23] based on the apparatus of mathematical logic, and on the ontology of designing [24–27] based on cognitive principles, on the methodology of active knowledge [28] and on many aspects of deep machine learning and intelligent technologies that could become the foundation for creating a knowledge base on mathematical models and algorithms within an integrated computing environment, which will allow, according to the figurative expression by Kleppe [29], moving from "paleo-informatics" to "neo-informatics" in mathematical modeling.

Note that the methodology of creating the integrated instrumental environments for cloud technologies has been actively developed in the Irkutsk scientific school of programming for many years, see [30]. A large amount of basic mathematical knowledge, including symbolic calculations, is incorporated in the well-known Mathematica and Maple systems, as well as in the Python [31] language and in the multi-user Matlab [32] tools. The use of meta-programming components with C^{++} [33] templates, as well as "factories" of problem-oriented languages for scientific research studies (Domestic Specific Languages, [29]) can significantly increase productivity. As for the extremely relevant mathematical knowledge base for the mathematical models, computational methods and technologies, here a good information and methodological basis is the project ALGO-WIKI [34].

The intellectualization of the ICE seems to be non-alternative and the highest priority software for a new generation of mathematical modeling on supercomputers. We emphasize that the project of the type MegaScience does not cancel the individual programming and provides a comfortable environment with numerous tools, which greatly increases the professional specialization and avoids duplication of works. No less important is another focus of the intelligent ICE, namely, on the formation of friendly interfaces for the end users, which can be divided into the following categories: research scientists and production engineers. The first can be attributed to both theorists and experimenters. The supercomputer modeling becomes an indispensable tool for obtaining the new fundamental or applied knowledge from any scientific field. As for practitioners, they ideally get virtual, or digital twins for their objects of activities, with which they can carry out or optimization of design, or planning, or operating of their processes. In fact, in such a symbiosis, a person receives an indispensable and reliable assistant for decision making, but not a competitor or rival as some futurologists write about it. And what is very important in the social sense, we will obtain a wide class of the new professions with high supercomputer skills.

To make such a picture of the world a reality, the computing community will have to perform a huge amount of work that should unite specialists of the widest profile on the foundation of mathematization and artificial intelligence.

From the professional point of view, the intelligent software components of ICE can be divided into the three parts. The first one is oriented to mathematicians and application programmers who are responsible for the performance and automatic construction of algorithms on the supercomputer architectures. The second part would provide the flexible internal and external interfaces to interact between subsystems of ICE and other products (CAD, CAE, CAM systems included), as well as arrangement for the users with various skills. The last but the least intelligent tool concern the supports the own life cycle which requires of different system components to maintenance and further development of the integrated computational environment for the above considered stages of mathematical modeling.

4 Conclusion

In our paper, we consider the problems of artificial intelligence and mathematical modeling, which are separately studied quite actively, but have not yet become objects of the joint targeted research. Together with the trends in the use of the ICE for mathematical modeling, this leads to a project of a national scale that requires serious organizational decisions, infrastructure and appropriate investments, see [35]. But this is the case when the goal justifies the means. We must say that not every country can form such a comprehensive mega-science project, but the Russian surviving scientific schools can make such a breakthrough. It is important to note that the predicted revolutionary technological progress in the world will lead to a widening the gap between the advanced and developing countries, and therefore our historical mission is to take a place in the business class of the new Noah's ark departing for the future.

We just discuss the problem statement and the main technological ideas on the creating the intelligent instrumental media for the modern supercomputer simulation. Here many architecture principles, the structure solutions, integration problems, and the implementation issues still present the open questions and the best topics for future research.

References

1. IESP. www.exascale.org/iesp
2. ANSYS. www.ansys.com
3. FEniCS. http://fenicsproject.org
4. Netlib. http://netlib.org
5. Intel *R* Mathematical Kernel Library. http://software.intel.com/en-us/intel-mkl
6. DUNE. http://www.dune-project.org
7. INMOST: A toolkit for distributed mathematical modeling. www.inmost.org
8. Il'in, V.P.: The conception, requirements and structure of the integrated computational environment. In: Voevodin, V., Sobolev, S. (eds.) RuSCDays 2018. CCIS, vol. 965, pp. 653–665. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05807-4_56
9. Il'in, V.P.: Mathematical Modelling, Part I: Continuous and Discrete Models. SBRAS Publ., Novosibirsk (2017). (in Russian)
10. Il'in, V.P.: Fundamental issues of mathematical modeling. Her. Russ. Acad. Sci. **86**(2), 118–126 (2016)
11. Il'in, V.P.: Mathematical modeling and the philosophy of science. Her. Russ. Acad. Sci. **88**(1), 81–88 (2018)
12. Forrester, A., Sobester, A., Keane, A.: Engineering Design via Surrogate Modeling: A Practical Guide. Wiley, New York (2008)
13. Delfour, M., Zolesio, J.-P.: Shape and Geometries: Metrics, Analysis, Differential Calculus, and Optimization. SIAM Publ., Philadelphia (2011)
14. Cottrell, J., Hughes, T., Bazilevs, Y.: Isogeometric Analysis: Towards Integration of CAD and FEA. Wiley, Singapore (2009)
15. Liseikin, V.D.: Grid Generation Methods. Springer, Berlin (2010). https://doi.org/10.1007/978-90-481-2912-6

16. Il'in, V.P.: DELAUNAY: technological environment for grid generation. Sib. J. Ind. Math. **16**, 83–97 (2013). (in Russian)
17. Butyugin, D.S., Il'in, V.P.: CHEBYSHEV: the principles of automatical constructions of algorithms for grid approximations of initial-boundary value problems. In: Proceeding of International Conference on PCT-2014, pp. 42–50. SUSU Publ., Chelyabinsk (2014). (in Russian)
18. Dongarra, J.: List of freely available software for linear algebra on the web (2006). http://netlib.org/utk/people/JackDongarra/la-sw.html
19. Butyugin, D.S., Gurieva, Y.L., Il'in, V.P., Perevozkin, D.V., Petukhov, A.V.: Functionality and algebraic solvers technologies in Krylov library. Vestnik YuUrGU. Ser. Comput. Math. Inform. **2**(3), 92–105 (2013). (in Russian)
20. Il'in, V.P.: Problems of parallel solution of large systems of linear algebraic equations. J. Math. Sci. **216**(6), 795–804 (2016)
21. Il'in, V.P.: On the numerical solution of the direct and inverse electromagnetic problems in geoprospecting. Sib. J. Num. Math. **6**(4), 381–394 (2003). (in Russian)
22. PARAVIEW. https://www.paraview.org
23. Goncharov, S.S., Sviridenko, D.I.: Logical language of description of polynomial computing. Dokl. Math. **99**(2), 1–4 (2019). ISSN 1064–5624
24. Valkman, Y.R., Tarasov, V.B.: From ontologies of cognitive semiotics. Ontol. Des. **f8**(1(27)), 8–34 (2018). (in Russian)
25. Mikony, S.V.: Formalization of the cognitive process using the basis of models. Ontol. Des. **8**(1(27)), 35–48 (2018). (in Russian)
26. Borgest, N.M.: Key tems the ontology of designing: review, analysis, generalization. Ontol. Des. **3**(3(9)), 9–341 (2013). (in Russian)
27. Zagorulko, Y.A., Borovikova, O.I.: An approach for realization of the content patterns in implementation of the scientific domains. Syst. Inform. **12**, 27–39 (2018). (in Russian)
28. Malyshkin, V.E.: Literacy for oncoming centuries. In: Proceedings of the 13th International Conference on Intelligent Software Methodologies, Tools, and Techniques (SoMeT). Frontiers in Artificial Intelligence and Applications, vol. 246, pp. 899–905. IOS Press (2014)
29. Kleppe, A.: Software Language Enginneering: Creating Domain-Specific Language Using Metamodels. Addison-Wesley, New York (2008)
30. Feoktistov, A., Kostromin, R., Sidorov, I.A., Gorsky, S.A.: Development of distributed subject-oriented applications for cloud computing through the integration of conceptual and modular programming. In: Proceedings of the 41st International Conference on Information and Communication Technology, Electronics and Microelectronics, pp. 251–256. IEEE (2018)
31. Python Language. http://www.python.org
32. MATLAB. https://www.mathworks.com/products/matlab.html
33. Krasnov, M.M.: $C++$ template metaprogramming in the mathematical physics problems. Keldysh IAM RAS, Moscow, preprint (2017)
34. ALGOWIKI. https://algowiki-project.org
35. Il'in, V.P.: How to reorganize computer science and technologies? Vestnik RAS **89**(3), 232–242 (2019). Moscow. (in Russian)