INMOST Parallel Platform for Mathematical Modeling and Applications

Kirill Terekhov[1]✉ and Yuri Vassilevski[1,2,3]

[1] Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow 119333
[2] Moscow Institute of Physics and Technology, Dolgoprudny 141701
[3] Sechenov University, Moscow 199991

kirill.terehov@gmail.com

Abstract. In the present work we present INMOST, the programming platform for mathematical modelling and its application to a couple of practical problems. INMOST consists of a number of tools: mesh and mesh data manipulation, automatic differentiation, linear solvers, support for multiphysics modelling. The application of INMOST to black-oil reservoir simulation and blood coagulation problem is considered.

Keywords: open-source library, linear solvers, automatic differentiation, reservoir simulation, blood coagulation

1 Introduction

Nowadays the necessity in parallel modelling of complex phenomena with multiple stiff physical processes is very acute. This puts a significant burden on the programmer who has not only to cope with various computational methods but also to manage the unstructured grid, data exchanges with MPI, assembly of large distributed linear systems, solve the resulting linear and nonlinear systems and finally postprocess the result. The INMOST [1] is an open-source library that alleviates the most of the complexity from the programmer and provides a unified set of tools to address each of the aforementioned issues. We have used the INMOST platform to implement the fully implicit black-oil reservoir model and fully coupled and implicit blood coagulation problem. The black-oil reservoir model involves simultaneous solution of three Darcy laws that describe the mixture of water, oil and gas. The blood coagulation model couples the Navier-Stokes equations with a Darcy term and nine additional advection-diffusion-reaction equations that participate in reactive cascade during coagulation of the blood. The numerical results for both models are presented.

Among notable alternatives for multiphysics modelling, commercial are COMSOL [2], ANSYS Fluent [3], Star-CD [4] and open-source are Dumux [5], OPM [6], Elmer [7], OOFEM [8], OpenFOAM [9], SU2 [10], CoolFluid [11] and many others. A comparison of some of these packages is available in [12]. Perspectives for multiphysics software are discussed in [13]. Should be noted, that all

of these packages provide modelling environment with integrated set of computational methods. The methods are implemented with certain assumptions and has certain limitations on physics, time stepping and couplings. An attempt to apply OpenFOAM package to blood coagulation model didn't allow to simulate the problem in reasonable time due to impossibility to construct fully-coupled approach from provided methods.

At present INMOST does not contain integrated computational methods but provides a programming platform to implement them. Among programming platforms the notable alternatives are Dune [17], Trilinos [15] and PETSc [14]. Dune is famous for distributed mesh management. Trilinos and PETSc are famous for a collection of built-in parallel linear solvers and seamless integration of third-party linear solvers; both also provide excellent nonlinear solvers, but rely on third party libraries for mesh management. Trilinos provides Sacado package for automatic differentiation. The framework for coupling of multiple physics modules is under active development in Dune (within Dumux project[5]) and in Trilinos (wihtin Amanzi project[41]) Functionality of all of these packages could be successfully used in large extent to build a simulator. The first version of the black-oil simulator reported in this work was based on linear and nonlinear solvers from PETSc and our own mesh management tool.

There is a number of C/C++ libraries that solve a particular task. For mesh management there are MSTK, MOAB, libMesh, FMDB and many other libraries. For linear system assembly and solution there are Trilinos, PETSc, SuperLU, MUMPS, Hypre and many others. For automatic differentiation there are Sacado (Trilinos), ADOL-C, FAD, ADEL, Adept and so on. For nonlinear solvers there are Trilinos, PETSc, SUNDIALS, Ipopt, Snopt and so on. For linear algebra integration there is Eigen library. The advantage of using separate tools could be in a greater level of maturity of popular libraries, the disadvantage is the absence of tight integration that is inevitably needed for construction of multiphysics framework.

2 Mesh and data

The algorithms and functionality that form the basis of mesh manipulation module in INMOST were previously reported in [19,20,22]. Parallel mesh refinement instruments were reported in [21]. We develop further the mesh modification functionality to support general mesh modification and synchronization in parallel and to preserve layers of ghost cells minimizing computational work.

The primary functionality required in this work are:

- load a mesh and associated data;
- compute partitioning of the mesh;
- redistribute the mesh;
- build multiple layers of ghost cells;
- access mesh elements, status of elements, mesh data;
- save the results in parallel VTK format.

Among novel functionalities of the mesh module we mention two novelties. First, the support of the Eclipse simulator mesh format *grdecl* is used in the INMOST black-oil reservoir model. Second, the ability to store on a mesh data containing partial derivatives is used in the implicit computation of a Jacobian matrix by the module of Automatic differentiation.

3 Automatic differentiation

The algorithms used in the automatic differentiation module were previously reported in [21]. The basis for the module are:

- a C++ class '*variable*' to represent a value of function with its first order partial derivatives;
- expression templates that form a templated tree of classes corresponding to operations on variables;
- a C++ class '*variable_ expression*' that can store expression template and evaluate it on demand;
- a dense linked-list structure '*Sparse::RowMerger*' is used for fast addition of sparse vectors, corresponding to the partial derivatives;

A novel functionality of INMOST is a templated class '*Matrix*' from the dense linear algebra module that allows for operations on dense matrices of values with specified type. Many '*blas*' and '*lapack*' operations are implemented in this class: sub-matrix access, addition, subtraction, multiplication, system solution, matrix inversion, singular value decomposition, pseudo-solve and pseudo-inverse and so on. The whole functionality is supported for matrices consisting first partial derivatives values. A set of simple rules prescribes the outcome when two matrices with different types of elements are involved in operations, *e.g.* multiplication of a matrix of doubles with a matrix of '*variable*'s results in a matrix of '*variable*'s). This functionality is heavily used in the blood coagulation model, see section 7.

To connect the mesh data and the automatic differentiation, we introduce an '*Automatizator*' class. An '*AbstractEntry*' sub-class allows to group the mesh data into blocks of unknowns and register them with the object of the '*Automatizator*' class. An index data is created and associated to each entry of a block of unknowns and enumeration for unknowns is performed on every mesh element containing the data related to the unknowns. Each block is enumerated consequently on each element of the mesh. This leads to the block-structured organization of the Jacobian matrix. Based on this, one can split the Jacobian matrix into blocks of physical processes and apply multigrid linear solvers [25,**?**,**?**,**?**] to blocks. In the future the information of the block structure of the matrix will be transfered to linear solvers.

The main functionality of the '*AbstractEntry*' class covers:

- *Value* provides on mesh element a matrix of values of unknowns;
- *Unknown* provides on mesh element a matrix of partial derivatives values;

- *Index* provides on mesh element a matrix of indices of unknowns, these indices are positions of unknowns in the Jacobian matrix;
- *MatrixSize* determines the size of the returned matrix.

Each value, unknown and index can be accessed individually if it is undesirable to get the whole matrix, the assembly of the matrix is being avoided in this case.

Various scenarios of unknowns organization is covered by sub-classes with extended functionality:

- '*SingleEntry*' is unknown as a single entry of datum on mesh elements;
- '*VectorEntry*' is unknown as multiple entries of datum on mesh elements (possibly with variable length);
- '*BlockEntry*' is unknown as fixed size data on mesh elements;
- '*StatusBlockEntry*' allows to change the status of unknowns in the block;
- '*MultiEntry*' is a union of entries of different types, *e.g.,* an object of '*SingleEntry*' can be used together with an object of '*VectorEntry*'.

Each of these sub-classes is inherited from the '*AbstractEntry*' class and retains the original functionality.

To facilitate the assembly of the residual vector and the Jacobian matrix, we introduced a '*Residual*' class. This class contains a sparse matrix and a residual vector (classes '*Sparse::Matrix*' and '*Sparse::Vector*', respectively). During the assembly stage a sparse matrix is stored in INMOST as a set of sparse vectors corresponding to rows of the matrix (class '*Sparse::Row*'). Each sparse vector is expandable and allows fast modifications. It is the same vector that is used to store partial derivatives values in the automatic differentiation module.

When an object of the class '*Residual*' is accessed by its index (as an array via square brackets $a[i]$), an object of the class '*variable_reference*' with references to corresponding entry in the residual vector and the row in the sparse matrix is returned. The object of this class can enter expression templates from automatic differentiation module and thus allows for all the same operations as an object of the '*variable*' class. Assignment to an object of this class results in modification of the sparse matrix and the residual vector stored in the object of the '*Residual*' class.

On top of that, an object of the class '*Residual*' can be accessed by a matrix of indices (*i.e.* $a[\mathbb{I}]$ where $\mathbb{I}$ is the matrix), then it returns a matrix with the type '*variable_reference*'. On assignment the underlaying sparse matrix and residual vector are modified in the block-structured fashion. In the future we shall introduce block-structured sparse matrices and use them in the '*Residual*' class to take advantage of the fast block-structured assembly.

All of this allows for seamless integration between the automatic differentiation module and the sparse linear algebra module. The next user's step is to solve the arising distributed linear system.

4 Linear solver

INMOST linear solver module provides integration of multiple open-source solver libraries: Trilinos [15], PETSc [14], SuperLU [16]. It also contains a number of

integrated solvers. Certain comparison of linear solvers was provided in [21]. One of the most widely used solvers is described below.

We solve the problem using preconditioned iterative method BiCGStab(L) [27]. This method makes L BiCG steps and fits a polynomial function to accelerate convergence of preconditioned residual to zero value.

The Additive Schwarz method with a prescribed overlap is used for the parallelization of the preconditioner.

For each local matrix extended by the overlap, we use the Crout-ILU method [28] with the dual threshold τ, τ_2 [29]. During elimination we estimate the condition number of the inverse factors $|L^{-1}|$ and $|U^{-1}|$ and adjust both thresholds accordingly to [31].

Before the elimination step, the matrix is preprocessed by reordering and rescaling in three steps. First, we maximize the product on the diagonal using the Dijkstra algorithm [33,34]. Second, we use the reverse Cuthill-Mckee algorithm to reduce the fill-in. At last, we use a rescaling to balance the second norms of all rows and all columns to unity [30].

In the future we plan to add local 2×2 pivoting [35], block-structured elimination and OpenMP parallelization [32].

5 Multiphysics

At present, INMOST contains trial implementation of the multi-physics module. The idea of the module is to provide basic functionality that allows one to split the problem into physical processes. Each physical process is responsible for a subset of unknowns and assembly of equations corresponding to these unknowns. Such a physical process is represented by an '*AbstractSubModel*' class. The programmer has to inherit from this class and implement the following functions:

- '*PrepareEntries*' introduces unknowns of a process to the model;
- '*FillResidual*' computes the residual for the process;
- '*UpdateMultiplier*' performs backtracking of an update to meet constraints;
- '*UpdateSolution*' updates unknowns during nonlinear iterations;
- '*UpdateTimeStep*' proceeds to the next time step;
- '*AdjustTimeStep*' computes an optimal time step for the process;
- '*RestoreTimeStep*' returns back in case of nonlinear solver failure.

Coupling between two physical processes introduces coupling terms into equations involving unknowns of both processes. This requires access to unknowns and equations of both processes.

Everything is managed by an object from a '*Model*' class. It incorporates an object of the '*Automatizator*' class and named arrays of meshes, entries of unknowns, sub-models and couplings. The coupling is represented by '*AbstractCoupling*' that has functions '*PrepareEntries*' and '*FillResidual*'.

This module uses heavily all the previously introduced modules and a nonlinear solver module which should guide the convergence of the multiphysics model to the solution. The latter module is not implemented in public repository yet.

6 Black-oil model

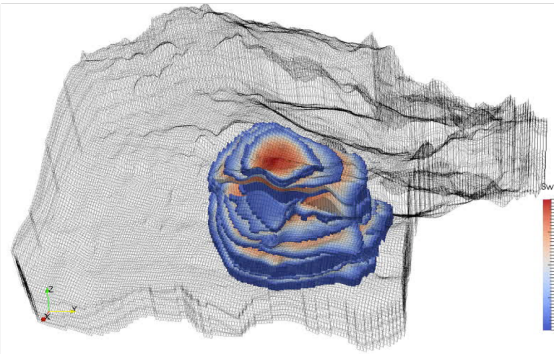

Fig. 1. An example of water injection into reservoir with real structure featuring layered heterogeneous media.

For the black-oil model we assume that the mixture of water, oil and gas fills all the voids, *i.e.* $S_o+S_w+S_g=1$, the mixture in the heterogeneous porous media is guided by the Darcy law with capillary pressure. There are three unknowns in the model: the water pressure p, the oil saturation S_o and either the gas saturation S_g or the bubble point pressure p_b depending on whether the gas phase is present or is fully dissolved in the oil phase, respectively. The oil and gas pressures are connected to water pressure p through the capillary pressures Pc_o and Pc_g. The system of equations takes the form:

$$\begin{aligned}
\frac{\partial \rho_w \theta S_w}{\partial t} - \nabla \cdot \lambda_w \mathbb{K} \left(\nabla p - \rho_w g \nabla z\right) &= q_w, \\
\frac{\partial \rho_o \theta S_o}{\partial t} - \nabla \cdot \lambda_o \mathbb{K} \left(\nabla p - \nabla Pc_o - \rho_o g \nabla z\right) &= q_o, \\
\frac{\partial \rho_{go} \theta S_o + \rho_g \theta S_g}{\partial t} - \nabla \cdot \lambda_g \mathbb{K} \left(\nabla p + \nabla Pc_g - \rho_g g \nabla z\right) & \\
- \nabla \cdot \lambda_{go} \mathbb{K} \left(\nabla p + \nabla Pc_g - \rho_{go} g \nabla z\right) &= q_g.
\end{aligned} \tag{1}$$

The equations are nonlinear due to the dependence of all the terms on unknowns of the problem. Here $\theta(p)$ is the porosity of the media, $\rho_\alpha(p)$ is the density of phase α, $\rho_{go}(p_b) = \rho_g(p_b) Rs(p_b)$ is the density of gas dissolved in oil, $Rs(p_b)$ describes the solubility of gas in the oil at a given bubble point pressure p_b, $\lambda_\alpha(p, S_\alpha) = \frac{\rho_\alpha(p) kr_\alpha(S_\alpha)}{\mu_\alpha(p)}$ is the mobility of phase α that accounts change in density, relative permeability and viscosity, $\lambda_{go}(p_b) = \frac{\rho_{go}(p_b) kr_g(S_g)}{\mu_g(p_b)}$ is the mobility for gas dissolved in oil phase, $Pc_o(S_o)$ and $Pc_g(S_g)$ are capillary pressures for oil and gas phase, respectively, q_α is the source or sink for phase α (conventionally the Peacman's well formula is used).

We use a nonlinear two-point flux approximation method [36,37] to calculate Darcy velocity $\mathbb{K}(\nabla \dots)$ terms. Once the velocity is obtained, a single point upstream discretization is used for the mobility. The whole system is approximated in the fully implicit manner by the backward Euler time-stepping method.

At each time step, the nonlinear system is solved by the Newton method. If on nonlinear iteration l the gas fully dissolves into oil $\widetilde{S}_g^l < 0$ or the gas is emitted $\widetilde{p}_b^l > p$, we solve a local nonlinear problem on the bubble point pressure p_b^l: $\rho_{go}\left(p_b^l\right) S_o^l = \rho_{go}\left(p^l\right) S_o^l + \rho_g(p^l)\widetilde{S}_g^l$ or on amount of the emitted gas S_g^l: $\rho_{go}\left(\widetilde{p}_b^l\right) S_o^l = \rho_{go}\left(p^l\right) S_o^l + \rho_g(p^l) S_g^l$, respectively. This step allows us to obtain physically reasonable quantities for the bubble point pressure and the released gas on the state switch. We also consider the same model without the gas phase, i.e. $S_g = 0$ and the last equation is not considered.

7 Blood coagulation

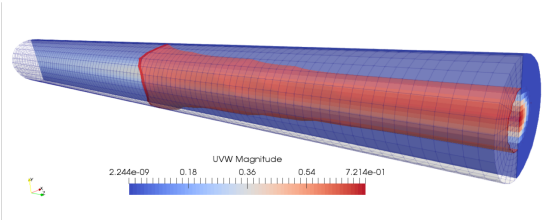

Fig. 2. An example of clot formation in the vessel.

The dynamics of the blood plasma in the present work is described by the incompressible Navier-Stokes equations with a Darcy permeability term:

$$\begin{cases} \dfrac{\partial \rho \mathbf{u}}{\partial t} + \mathrm{div}\left(\rho \mathbf{u}\mathbf{u}^T - \mu \nabla \mathbf{u} + p\mathbb{I}\right) = -\dfrac{1}{K_f}\mathbf{u}, \\ \mathrm{div}\mathbf{u} = 0. \end{cases} \tag{2}$$

The density ρ and viscosity μ are assumed to be constant. The equations (2) are coupled to the following concentrations of reactive components in the flow: prothrombin PT, thrombin T, anti-thrombin A, blood clotting factor FX Ba, fibrin F, fibrinogen F_g, fibrin polymer F_p, free platelets ϕ_f in the flow, platelets ϕ_c trapped in the clot. All the components except for the fibrin polymer and platelets satisfy the advection-diffusion equation: $\frac{\partial C}{\partial t} + \nabla \cdot (C\mathbf{u} - D\nabla C) = R_C$ for a component C. The platelets satisfy the nonlinear advection-diffusion equation $\frac{\partial C}{\partial t} + \nabla \cdot k(\phi_c)\,(C\mathbf{u} - D_c \nabla C) = R_C$ with $k(\phi_c) = \tanh\left(\pi(1 - \phi_c/\phi_{max})\right)$, $C = \phi_f, \phi_c$. Polymerised fibrin does not move with the flow. The set of kinetic reactions between components of the flow is taken from [40]. The parameter K_f

(2) describes the formation of porous media due to fibrin polymer and platelets similar to [40].

All the unknowns (velocity vector $\mathbf{u}$, pressure p and all the additional components) are collocated at the centres of the cells. For the advection and diffusion terms we use the second order finite-volume approximation, the inertia term is being approximated nonlinearly. Collocation of velocities and pressures at cell centers requires special stabilization for discretization of the flux related to the gradient of pressure and divergence of velocity. The finite volume discretization results in a system composed of 13 unknowns and equations per computational cell. The whole system is solved simultaneously in the implicit manner by the BDF2 time-stepping scheme.

8 Numerical results

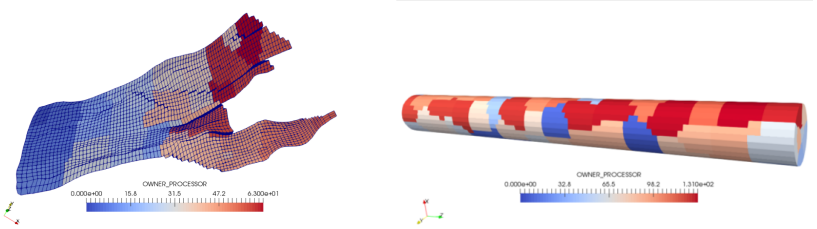

Fig. 3. Partitioning of the computational mesh among processors, the black-oil model on the Norne field (left), the blood coagulation model in a cylindric vessel corresponding to experiment [38].

For the black oil model we use the secondary recovery problem on the Norne field [39] with one injection well and two production wells. The problem runs for 100 modelling days with at maximum 1 day step. The partition of the mesh is demonstrated in Figure 3 (left). The parameters are taken from SPE9 test. For the blood coagulation model we simulate 15 seconds of the clot formation observed in the experiment [38]. The partition of the mesh is demonstrated in Figure 3 (right). The parameters are taken from [40].

In both problems the number of overlap layers in the mesh was set to 2 for assembly of fluxes and in additive Schwartz method was set to 1, the solver dropping parameters are $\tau = 10^{-2}$ and $\tau_2 = 10^{-3}$.

The performance of the models is demonstrated in Table 1. Results indicate that the clotting model scales very well with the growth of the number of processors. This happens due domination of reactions, the problem is hyperbolic due to low viscosity of blood plasma. As a result the solver behaves very good.

On contrary the black oil problem contains strong elliptic component and the performance of linear solver deteriorates faster. For good performance this

Table 1. Performance of the models on different numbers of processors.

Two-phase model								
Processors	Assembly		Solution		Total		Equations
1	1028	-	1544	-	2581	-	89830
8	172	6x	383	4x	558	5x	11229
16	89	12x	255	6x	345	8x	5615
32	53	20x	135	12x	189	14x	2808
64	30	35x	71	22x	101	26x	1404
128	19	55x	65	24x	84	31x	702
Three-phase model							
Processors	Assembly		Solution		Total		Equations
1	2783	-	5368	-	8171	-	134745
8	449	6x	3295	2x	3749	2x	16844
16	252	11x	1656	3x	1911	4x	8422
32	152	19x	472	11x	626	13x	4211
64	88	32x	325	17x	415	20x	2106
128	59	47x	154	35x	213	38x	1053
Blood clotting model							
Processors	Assembly		Solution		Total		Equations
1	2781	-	2156	-	4938	-	318500
8	461	6x	214	10x	680	7x	39812
16	253	11x	117	18x	373	13x	19906
32	148	19x	80	27x	234	21x	9954
64	90	31x	46	47x	141	35x	4977
128	46	60x	29	74x	77	64x	2489

problem require a specific solver [25] that can extract elliptic part of the problem and apply multi-grid on it. In this problem the assembly of the matrix does not ideally scale, since we have to perform certain operations on overlap which may become significant with large number of processors. Still it remains reasonable to increase number of processors to reduce total computational time.

The calculations were performed on cluster of INM RAS [18], we run the job on nodes with different types of processors.

9 Conclusion

We have presented briefly the open-source platform INMOST for parallel mathematical modelling. The platform allows to greatly facilitate programming of complex coupled models. Parts of the INMOST platform are still under active development. In the future we plan to advance the multiphysics tools for the solution of nonlinear systems of equations and abstractions for seamless modular integration of independent physical models.

Acknowledgement. This work was supported by the RFBR grants 17-01-00886, 17-51-53160.

References

1. INMOST – a toolkit for distributed mathematical modeling.
   URL: http://www.inmost.org (access date: 15.04.2018)
2. COMSOL Multiphysics Reference Manual, version 5.3, COMSOL, Inc,
   URL: www.comsol.com (access date: 30.05.2018)
3. ANSYS FLUENT ,
   URL: https://www.ansys.com/products/fluids/ansys-fluent (access date: 30.05.2018)
4. STAR-CD,
   URL: https://mdx.plm.automation.siemens.com/star-cd (access date: 30.05.2018)
5. *Flemisch B., Darcis M., Erbertseder K., Faigle B., Lauser A., Mosthaf K., Müthing S., Nuske P., Tatomir A., Wolff M., Helmig R.* DuMux: DUNE for multi-phase, component, scale, physics,... flow and transport in porous media //Advances in Water Resources. – 2011. – T. 34. – №. 9. – C. 1102-1112.
6. The Open Porous Media (OPM) initiative encourages open innovation and reproducible research for modeling and simulation of porous media processes.
   URL: https://opm-project.org (access date: 30.05.2018)
7. Elmer – Finite Element Solver for Multiphysical Problems,
   URL: https://www.csc.fi/web/elmer (access date: 30.05.2018)
8. *Patzák B.* OOFEM—an object-oriented simulation tool for advanced modeling of materials and structures //Acta Polytechnica. – 2012. – T. 52. – №. 6.
9. OpenFOAM is the free, open source CFD software.
   URL: www.openfoam.com (access date: 30.05.2018)
10. SU2 is an open-source collection of software tools written in C++ and Python for the analysis of partial differential equations (PDEs) and PDE-constrained optimization problems on unstructured meshes with state-of-the-art numerical methods.
    URL: https://su2code.github.io (access date: 30.05.2018)
11. COOLFluiD is a component based scientific computing environment that handles high-performance computing problems with focus on complex computational fluid dynamics (CFD) involving multiphysics phenomena.
    URL: https://github.com/andrealani/COOLFluiD/wiki (access date: 30.05.2018)
12. *Babur Ö., Smilauer V., Verhoeff T., van den Brand M.* A survey of open source multiphysics frameworks in engineering //Procedia Computer Science. – 2015. – T. 51. – C. 1088-1097.
13. *Keyes D. E. et al.* Multiphysics simulations: Challenges and opportunities //The International Journal of High Performance Computing Applications. – 2013. – T. 27. – №. 1. – C. 4-83.
14. PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations.
    URL: http://www.mcs.anl.gov/petsc (access date: 15.04.2018)
15. Trilinos – platform for the solution of large-scale, complex multi-physics engineering and scientific problems.
    URL: http://trilinos.org/ (access date: 15.04.2018)
16. SuperLU is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations.
    URL: http://crd-legacy.lbl.gov/ xiaoye/SuperLU/ (access date: 15.04.2018)
17. Distributed and Unified Numerics Environment.
    URL: https://dune-project.org/ (access date: 30.05.2018)

18. INM RAS cluster.
URL: http://cluster2.inm.ras.ru (access date: 15.04.2018)
19. *Terekhov K.M* Application of unstructured octree grid to the solution of filtration and hydrodynamics problems (in Russian) // PhD Thesis, INM RAS, 2013
20. INMOST - programming platform and graphical environment for development of parallel numerical models on general grids (in Russian) // Yu. V. Vassilevski, I. N. Konshin, G. V. Kopytov, K. M. Terekhov. — Moscow University Press, 2013. — 144 p.
21. *Bagaev, D. V., Burachkovskii, A. I., Danilov, A. A., Konshin, I. N., Terekhov, K. M.* Developement of INMOST programming platform: dynamic grids, linear solvers and automatic differentiation. // Russian Supercomputing Days 2016.
22. *Danilov, A. A., Terekhov, K. M., Konshin, I. N., Vassilevski, Yu. V.* . Parallel software platform INMOST: a framework for numerical modeling. // Supercomputing Frontiers and Innovations, 2(4), 55-66, 2015.
23. *Jonathan J. Hu, Prokopenko A., Siefert C.M., Tuminaro R.S. and Wiesner T.A.* MueLu multigrid framework.
URL: http://trilinos.org/packages/muelu (access date: 15.04.2018)
24. SAMG – Efficiently solving large Linear Systems of Equations.
URL: https://www.scai.fraunhofer.de/en/business-research-areas/fast-solvers/products/samg.html
25. *Lacroix, S., Vassilevski, Yu. V., Wheeler, M. F.*. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS). // Numerical linear algebra with applications, 8(8), 537-549, 2001.
26. *Castelletto, N., White, J. A., Tchelepi, H. A.* Accuracy and convergence properties of the fixed-stress iterative solution of two-way coupled poromechanics. // International Journal for Numerical and Analytical Methods in Geomechanics, 39(14), 1593-1618, 2015
27. *Sleijpen G. L. G., Fokkema D. R.* BiCGstab (l) for linear equations involving unsymmetric matrices with complex spectrum // Electronic Transactions on Numerical Analysis. – 1993. – Т. 1. – №. 11. – С. 2000.
28. *Li N., Saad Y., Chow E.* Crout versions of ILU for general sparse matrices //SIAM Journal on Scientific Computing. – 2003. – Т. 25. – №. 2. – С. 716-728.
29. *Kaporin I. E.* High quality preconditioning of a general symmetric positive definite matrix based on its UTU+ UTR+ RTU-decomposition //Numerical linear algebra with applications. – 1998. – Т. 5. – №. 6. – С. 483-509.
30. *Kaporin I. E.* Scaling, reordering, and diagonal pivoting in ILU preconditionings //Russian Journal of Numerical Analysis and Mathematical Modelling. – 2007. – Т. 22. – №. 4. – С. 341-375.
31. *Bollhöfer M.* A robust ILU with pivoting based on monitoring the growth of the inverse factors //Linear Algebra and its Applications. – 2001. – Т. 338. – №. 1-3. – С. 201-218.
32. *Aliaga, J. I., Bollhöfer, M., Marti, A. F., Quintana-Orti, E. S.* Exploiting thread-level parallelism in the iterative solution of sparse linear systems //Parallel Computing. – 2011. – Т. 37. – №. 3. – С. 183-202.
33. *Olschowka M., Neumaier A.* A new pivoting strategy for Gaussian elimination //Linear Algebra and its Applications. – 1996. – Т. 240. – С. 131-151.
34. *Duff I. S., Koster J.* The design and use of algorithms for permuting large entries to the diagonal of sparse matrices //SIAM Journal on Matrix Analysis and Applications. – 1999. – Т. 20. – №. 4. – С. 889-901.

35. *Bunch J. R., Kaufman L.* A computational method for the indefinite quadratic programming problem //Linear Algebra and its Applications. – 1980. – T. 34. – C. 341-370.
36. *Nikitin K., Terekhov K., Vassilevski Yu. A* monotone nonlinear finite volume method for diffusion equations and multiphase flows //Computational Geosciences. – 2014. – T. 18. – №. 3-4. – C. 311-324.
37. *Terekhov K. M., Mallison B. T., Tchelepi H. A.* Cell-centered nonlinear finite-volume methods for the heterogeneous anisotropic diffusion problem //Journal of Computational Physics. – 2017. – T. 330. – C. 245-267.
38. *Shen, F., Kastrup, C. J., Liu, Y., Ismagilov, R. F.* Threshold response of initiation of blood coagulation by tissue factor in patterned microfluidic capillaries is controlled by shear rate //Arteriosclerosis, thrombosis, and vascular biology. – 2008. – T. 28. – №. 11. – C. 2035-2041.
39. Norne: the full Norne benchmark case, a real field black-oil model for an oil field in the Norwegian Sea. URL: https://opm-project.org/?page_id=559 (access date: 15.04.2018)
40. *Bouchnita A.* Mathematical modelling of blood coagulation and thrombus formation under flow in normal and pathological conditions // PhD thesis, Université Lyon 1 - Claude Bernard; Ecole Mohammadia d'Ingénieurs - Université Mohammed V de Rabat - Maroc, 2017.
41. *Coon E. T., Moulton J. D., Painter S. L.* Managing complexity in simulations of land surface and near-surface processes //Environmental Modelling and Software. – 2016. – T. 78. – C. 134-149.