

Ani3D-extension of parallel platform INMOST and hydrodynamic applications

Vasily Kramarenko^{1,2}, Igor Konshin^{1,3}, and Yuri Vassilevski^{1,2}

¹ Institute of Numerical Mathematics of the Russian Academy of Sciences, Moscow
119333, Russia

² Moscow Institute of Physics and Technology, Dolgoprudny, Moscow Region,
141701, Russia

³ Dorodnicyn Computing Centre, FRC CSC RAS, Moscow 119333, Russia
kramarenko.vasily@gmail.com

Abstract. The paper is devoted to an extension of the parallel platform INMOST by finite element and meshing libraries of the Ani3D software package. The extension allows us to develop parallel finite element solvers of boundary value problems and, in particular, hydrodynamic problems. The Ani3D package allows one to build, refine, locally adapt and improve the quality of tetrahedral meshes, perform finite element discretizations of partial differential equations for various types of finite elements, solve the appearing algebraic systems, and visualize the discrete solutions. The INMOST software platform provides tools for creating and storing distributed general conformal grids with arbitrary polyhedral cells, parallel assembling and parallel solution of arising distributed linear systems. We present the integration of two libraries from Ani3D into INMOST platform and demonstrate the functionality of the joint software on the solution of two model hydrodynamic problems on multiprocessor systems.

Keywords: parallel computing, finite element method, parallel solvers, hydrodynamic problems

1 Introduction

We consider an extension of the parallel platform INMOST by finite element and meshing libraries of the Ani3D software package. The extension provides a tool for developing parallel finite element solvers of boundary value problems and, in particular, hydrodynamic problems. The Ani3D package [1] offers advanced finite element discretizations on tetrahedral meshes and various options of tetrahedral mesh generation, refinement, and adaptation. The parallel platform INMOST [2] provides tools for creating and storing distributed general conformal grids with arbitrary polyhedral cells, parallel assembling and parallel solution of arising distributed linear systems. Integration of two libraries from Ani3D into the INMOST platform offers a new technology of parallel solution of boundary value problems. Functionality of the joint software is demonstrated on the solution of two model hydrodynamic problems on multiprocessor systems.

2 V. Kramarenko, I. Konshin, and Yu. Vassilevski

The paper is organized as follows. Sections 2 and 3 contain brief descriptions of Ani3D and INMOST packages, respectively. Section 4 provides technical details of merging the packages. Section 5 demonstrates the parallel solution of two model hydrodynamic problems.

2 Ani3D package

The package Ani3D [1] is developed for generation of unstructured tetrahedral meshes, adaptation of these meshes isotropically or anisotropically, discretization of PDE systems, solution of linear and nonlinear systems, visualization of meshes and associated solutions. It consists of a set of independent libraries oriented to the solution of the specific tasks. All these libraries allow a user to operate with data in sequential mode only. We consider the Ani3D-extension of the INMOST platform by two Ani3D libraries, Ani3D-MBA and Ani3D-FEM.

The main purpose of the Ani3D-MBA library is generation of conformal tetrahedral meshes which are quasi-uniform in a given metric. Additionally, the library provides tools to read/write a tetrahedral mesh from/to the disk in a specific Ani3D format and to perform its uniform mesh refinement by splitting each tetrahedron into 8 sub-tetrahedra.

The Ani3D-FEM library provides a flexible interface to generate a local finite element discretization (local matrix and right-hand side vector) on a mesh tetrahedron and to assemble the local discretizations into a global system of grid equations. Importantly, the local discretization may involve different types of finite elements: for instance, the local matrix for the Stokes problem may exploit quadratic basis functions for velocity and linear basis functions for pressure unknowns. Our finite element extension of the INMOST platform uses a user-defined subroutine `FEM3Dext` where the local finite element matrix is generated. The library Ani3D-FEM is equipped with a great number of examples of this subroutine for various applications. The rules for creation of the subroutine `FEM3Dext` and respective examples can be found in Ani3D documentation [1]. In particular, the user should specify explicitly the order of cell elements collocating the user's finite element basis functions. For instance, quadratic basis functions have four degrees of freedom collocated at the vertices of the tetrahedron and six degrees of freedom collocated at the mid-edges of the tetrahedron. Also, the user may transfer user data to the subroutine `FEM3Dext` with the help of special working arrays. This feature can be used for acquiring the solution from the previous time step which is inevitable for unsteady time stepping implementations.

3 INMOST platform

The INMOST software platform [2] is instrumented for creating and storing distributed general conformal grids with arbitrary polyhedral cells, parallel assembling of systems of grid equations and their parallel solution. However, INMOST does not provide software for generation and assembling of local finite element discretizations. In order to add to INMOST the finite element environment from

Ani3D-FEM, we take advantage of Mesh class, Solver class, and Sparse::Matrix class of INMOST.

The Mesh class is designed for storing distributed grids. It contains a number of cells consisting of nodes, faces, edges. In parallel mode each processor has a sets of “owned” and “shared” elements and a set of “ghost” elements. Each “ghost” element in fact is the copy of an element owned by another processor and marked as “shared” there. These elements are used for the construction of overlapping communication layers between processors. We note that some cell can be “ghost” for Processor A, but its node/edge/face can be owned by Processor B.

The important data structure of INMOST is Tag which is used to connect any data with a mesh element, i.e. cell, face, edge, or node. The simplest case of tagged data is a real or integer array associated with every mesh element of particular type (e.g. every edge). The main function of Tag is to provide automatic exchanges of tagged data between neighboring processors.

The important feature of Sparse::Matrix class is that it stores the matrix by rows in parallel regime. Processor B cannot add entries to a row owned by Processor A. In order to assemble local matrices in parallel, one has to use special numbering of rows.

No special features of Solver class will be used for our purposes. By this reason, any of five inner linear solvers or five external linear solvers from PETSc, Trilinos and SuperLU can be exploited in the same interface.

The detailed description of INMOST software platform can be found in [3,4,5].

4 Parallelization technology

In this section we present technological details of integration of the Ani3D uniform mesh refinement and the Ani3D local finite element matrix generation into the INMOST platform.

The first part of the Ani3D-INMOST technology allows us to generate huge meshes on multi-processor systems. The major steps are reading the initial mesh, its partitioning, refinement, and merging the refined submeshes into a global distributed mesh. Reading the mesh is performed by a standard Ani3D-MBA library routine on the root processor. The result of the reading, the object INMOST::Mesh, is processed by one of INMOST partitioning algorithms (e.g., ParMETIS [6] partitioning) and redistributed among available processors. Each submesh is refined independently. Parallel multilevel mesh refinement requires mesh conformity control. The uniform refinement function in Ani3D guarantees this property provided that the initial (coarse) submeshes form a conformal global mesh and the number of refinement levels is the same on all processors. Merging the fine submeshes removes duplicate nodes, edges, and faces in the global fine mesh. Once the fine submeshes are generated and processed to constitute the global conformal mesh, we construct an additional layer of “ghost”

4 V. Kramarenko, I. Konshin, and Yu. Vassilevski

cells. This layer is needed for correct assembling of the global finite element matrix.

The second part of the integrated Ani3D-INMOST software initializes the INMOST tags and data which will be used in assembling of local finite element matrices generated by Ani3D-FEM library. First, all mesh elements involved in the discretization are numbered within the global mesh. INMOST function `AssignGlobaID` numbers the respective elements (cells and/or faces, edges, nodes) marked by appropriate mask. Second, for all globally numbered elements a special tag is created, the size of the tag in each cell being equal to the number of finite element degrees of freedom associated to the mesh element. Finally, the tags are synchronized between processors. Global numbering and synchronized tags provide easy recovering of the global matrix order as well as the order of matrices owned by processors. Importantly, flexibility for numbering degrees of freedom within INMOST allows the user to generate distributed matrices with desirable ordering. The proper ordering may improve the performance of INMOST linear solvers.

The third part of the Ani3D-INMOST technology assembles the local matrices generated on each tetrahedral cell by the user-defined Ani3D-FEM routine `FEM3Dext`. On each processor, INMOST-based assembling selects rows of each local matrix which correspond to owned (by the processor) mesh elements, and writes the respective entries to the global matrix and the right-hand side vector.

Once the global system is assembled, any INMOST parallel linear solver can be applied to the solution of the distributed global linear system.

5 Parallel solution of model hydrodynamic problems

5.1 Stokes problem

We consider the finite element solution of the Stokes problem in a rectangular 3D channel with a backward step. We impose the non-homogeneous Dirichlet boundary condition (Poiseuille's profile) at the inflow boundary, the homogeneous Neumann boundary condition at the outflow boundary, and the homogeneous Dirichlet boundary condition (no-slip, no-penetration) on the channel walls. A sequence of three quasi-uniform tetrahedral meshes is considered. The coarsest mesh S0 with 25113 cells (see Figure 1) is uniformly refined to get more finer meshes S1 and S2 with 200904 and 1607232 cells, respectively.

The minimal order Taylor–Hood finite elements are used for the discretization of the Stokes problem. The pressure p is approximated by continuous piecewise linear functions with nodal degrees of freedom, the velocity \mathbf{v} is approximated by continuous piecewise quadratic functions with degrees of freedom collocated at nodes and mid-edges of the mesh (see Figure 2). The discretization method results in a symmetric saddle-point matrix with zero diagonal pressure block:

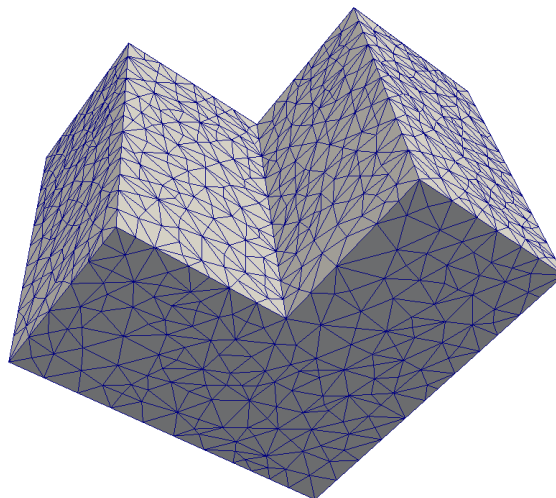


Fig. 1: The coarsest mesh S0

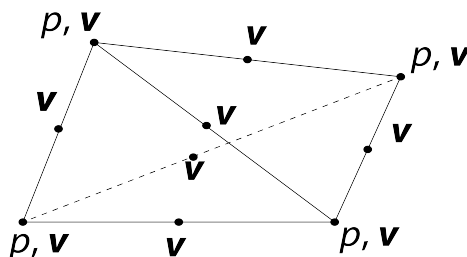


Fig. 2: Unknowns p and \mathbf{v} associated with the vertices and edges of the tetrahedral cell for P1-P2 finite element

\mathbf{v}_{edge}	*	\times	\times
\mathbf{v}_{node}	\times	*	\times
p_{node}	\times	\times	0

The numerical experiments were performed on the INM RAS cluster [7] in the x10core segment:

- Compute Node Arbyte Alkazar+ R2Q50;
- 20 cores (two 10-cores processors Intel Xeon E5-2670v2@2.50GHz);
- 64 GB RAM;
- SUSE Linux Enterprise Server 11 SP3 (x86_64).

Table 1 presents statistics for all three finite element problems.

6 V. Kramarenko, I. Konshin, and Yu. Vassilevski

Table 1: The Stokes problems parameters

Problem name	S0	S1	S2
Number of nodes	5187	36824	279903
Number of edges	31637	243079	1908542
Number of tetrahedra	25113	200904	1607232
Matrix size	115659	876533	6845238
Number of nonzeros	10751851	84374191	668849086

To solve the linear system with the saddle-point matrix, we used the BiCGstab iterative solver preconditioned by the first order BIILU method [8,9]. In order to avoid zero pivots during ILU factorization, on each processor pressure unknowns are enumerated last so that the zero pressure block be the last diagonal block [10]. Parameters of the BIILU method are as follows: the threshold parameter for the conventional incomplete factorization $ILU(\tau)$ $\tau = 0.001$ and the number of overlap levels $q = 2$. The use of the conventional first order incomplete factorization is more beneficial compared to the more robust second order incomplete factorization $ILU2(\tau_1, \tau_2)$ due to very large average number of nonzero elements per matrix row (about 100, see Table 1). The stopping criterion for the iterations is 10^{12} -fold reduction of the initial residual.

The results of numerical experiments for problems S0, S1, and S2 are presented in Tables 2, 3, and 4, respectively. In these tables p denotes the number of processors used, T_{ini} , T_{ass} , T_{prec} , and T_{iter} are the times for the preliminary data initialization, assembling of the linear system, preconditioner construction, and performing iterations by the BiCGstab method, respectively, N_{iter} stands for the number of BiCGstab iterations, Dens specifies the preconditioner density with respect to that of the original matrix of the system, PM is the number of pivot modifications, $T_{sol} = T_{prec} + T_{iter}$ is the total linear system solution time, while $S = T_{sol}(1)/T_{sol}(p)$ is the actual speedup relative to the solution time on one (Tables 2, 3) or four (Tables 4) processors. Table 4 does not contain data for runs on 1 and 2 processors due to memory restrictions.

Table 2: The solution of S0 problem on $p = 1, \dots, 32$ processors

p	T_{ini}	T_{ass}	T_{prec}	T_{iter}	N_{iter}	Dens	PM	T_{sol}	S
1	0.06	8.77	3.30	4.26	102	0.81	0	7.56	1.00
2	0.04	5.72	2.27	3.27	132	0.96	0	5.54	1.36
4	0.03	3.95	1.57	2.34	152	1.18	0	3.91	1.93
8	0.02	2.25	1.19	1.81	172	1.53	5	3.00	2.52
16	0.02	1.63	1.20	1.40	182	2.01	3	2.83	2.67
32	0.02	1.49	1.50	1.33	182	2.80	10	2.83	2.67

We do not observe a slowdown even for the solution of system with the smallest matrix S0 on 32 processors when approximately 5000 matrix rows are

Table 3: The solution of S1 problem on $p = 1, \dots, 32$ processors

p	T_{ini}	T_{ass}	T_{prec}	T_{iter}	N_{iter}	Dens	PM	T_{sol}	S
1	0.38	69.07	31.41	97.20	242	0.85	0	128.61	1.00
2	0.28	40.97	20.30	76.88	322	0.93	0	97.18	1.32
4	0.21	25.31	12.76	47.88	332	1.03	0	60.64	2.12
8	0.14	13.76	7.83	28.68	332	1.15	1	36.51	3.52
16	0.10	8.37	4.77	18.46	362	1.37	3	23.23	5.53
32	0.06	5.09	4.12	12.70	402	1.65	8	16.82	7.64

Table 4: The solution of S2 problem on $p = 4, \dots, 32$ processors

p	T_{ini}	T_{ass}	T_{prec}	T_{iter}	N_{iter}	Dens	PM	T_{sol}	S
4	1.48	181.05	142.29	1484.02	722	0.97	0	1626.31	1.00
8	0.90	94.03	76.01	847.62	802	1.03	0	923.63	1.76
16	0.58	52.03	72.70	481.50	802	1.10	2	554.20	2.93
32	0.37	29.23	26.88	288.73	802	1.21	2	315.61	5.15

associated with each processor. For the moderate size matrix S1 the maximal speedup is 7.64, while for the largest matrix S2 the speedup is 5.15 when the number of processors increases from 4 to 32.

5.2 Unsteady convection–diffusion problem

In the second test we consider the finite element solution of an unsteady convection–diffusion problem in a cubic domain. We impose homogeneous Dirichlet boundary conditions on all boundaries of the cube, except a patch centered at one cube face. In the patch the concentration is set to one. The diffusion coefficient is $D = 10^{-4}$, the convection field is the constant vector $\mathbf{v} = (1, 0, 0)$. The problem coefficients imply tongue-type propagation of the concentration in time and space (see Fig. 3). The initial quasi-uniform tetrahedral mesh L0 with 13952 cells is uniformly refined one and two times to produce meshes L1 and L2, respectively. The unknown concentration is approximated by continuous piecewise linear basis functions with nodal degrees of freedom. The finite element discretization of the convection operator is stabilized by the Streamline Upwind Petrov–Galerkin (SUPG) method. The second order implicit backward differentiation formula (BDF) scheme is used for time stepping. The problem is solved for the time period $[0; 0.5]$ with time step $\Delta t = 0.015$.

The numerical experiments were performed on the same computational system with the same computational method as in the previous test. Table 5 presents statistics for all three finite element problems on meshes L0, L1, and L2.

The time measurements for the solution of problems L0, L1, and L2 are presented in Tables 6, 7, and 8, respectively. In these tables p denotes the number of processors used, T_{ini} , T_{ass} , and T_{sol} , are the times for the preliminary data initialization, cumulative time of assembling the linear systems for all time steps,

8 V. Kramarenko, I. Konshin, and Yu. Vassilevski

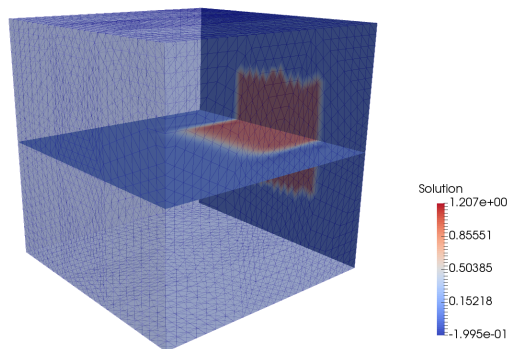


Fig. 3: The concentration at time $t = 0.5$

Table 5: The problems parameters

Problem name	L0	L1	L2
Number of nodes	20417	155905	1218561
Number of tetrahedra	111616	892928	7143424
Matrix size	20417	155905	1218561
Number of nonzeros	291393	2281217	18053121

and cumulative time of the solution of all linear systems, respectively. In addition, T_Σ is the total problem solution time for all time steps and

$$S = \frac{T_{\text{sol}}(1) + T_{\text{ass}}(1) + T_{\text{ini}}(1)}{T_{\text{sol}}(p) + T_{\text{ass}}(p) + T_{\text{ini}}(p)}$$

is the actual speedup relative to the solution time on one processor.

Table 6: The solution of problem on mesh L0 on $p = 1, \dots, 32$ processors

p	T_{ini}	T_{ass}	T_{sol}	T_Σ	S
1	2.87	62.68	3.27	68.82	1.00
2	1.93	40.45	2.05	44.43	1.54
4	1.29	26.94	1.24	29.47	2.33
8	0.82	16.76	0.76	18.34	3.75
16	0.58	11.91	0.53	13.02	5.28
32	0.45	8.89	0.55	9.89	6.95

Similarly to the solution of the Stokes problem, we do not observe a slowdown even for the smallest matrix L0 on 32 processors (when less than 700 matrix rows are associated to a processor). For the moderate size matrix L1 the maximal speedup is 11.77, while for the largest matrix L2 the speedup is 19.24. This test shows good scalability of the Ani3D-extension of the INMOST platform for unsteady problems.

Table 7: The solution of problem on mesh L1 on $p = 1, \dots, 32$ processors

p	T_{ini}	T_{ass}	T_{sol}	T_{Σ}	S
1	26.16	552.87	33.86	612.89	1.00
2	16.48	327.93	21.31	365.72	1.67
4	9.83	197.89	12.19	219.91	2.78
8	5.75	114.81	7.28	127.84	4.79
16	3.56	73.57	4.03	81.16	7.55
32	2.31	47.28	2.45	52.04	11.77

Table 8: The solution of problem on mesh L2 on $p = 1, \dots, 32$ processors

p	T_{ini}	T_{ass}	T_{sol}	T_{Σ}	S
1	436.82	5692.31	723.23	6852.36	1.00
2	169.17	2628.09	258.71	3055.97	2.24
4	94.33	1461.24	147.30	1702.87	4.02
8	53.62	874.21	91.24	1019.07	6.72
16	31.52	522.99	53.28	607.79	11.27
32	17.60	308.84	29.61	356.05	19.24

Conclusion

We presented the Ani3D-extension of the parallel platform INMOST. The extension widens the functionality of INMOST by the finite element and meshing libraries of the Ani3D software package. Two numerical examples demonstrated the efficiency of the presented approach for the parallel solution of two model hydrodynamic problems.

Acknowledgements. This work has been supported by RFBR grant 17-01-00886.

References

1. Lipnikov K., Vassilevski Yu., Danilov A. et al.: Advanced Numerical Instruments 3D. URL: <http://sourceforge.net/projects/ani3d> (accessed: 15.04.2017)
2. INMOST – a toolkit for distributed mathematical modeling. URL: <http://www.inmost.org> (accessed: 15.04.2017)
3. Vassilevski Yu., Konshin I., Kopytov G., Terekhov K.: INMOST – a software platform and graphical environment for development of parallel numerical models on general meshes. Lomonosov Moscow State Univ. Publ., Moscow, 144 p. (2013) (in Russian)
4. Danilov A. A., Terekhov K. M., Konshin I. N., Vassilevski Y. V.: Parallel software platform INMOST: a framework for numerical modeling. Supercomputing Frontiers and Innovations V.2, No.4, 55–66 (2015)
5. Konshin I., Kapyrin I., Nikitin K., Terekhov K.: Application of the parallel INMOST platform to subsurface flow and transport modelling. Lecture Notes in Computer Science, Vol. 9574, 277–286 (2016)

10 V. Kramarenko, I. Konshin, and Yu. Vassilevski

6. ParMETIS – Parallel graph partitioning and fill-reducing matrix ordering.
URL: <http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview> (accessed: 15.04.2017)
7. INM RAS cluster. URL: <http://cluster2.inm.ras.ru> (accessed: 15.04.2017) (in Russian)
8. Kaporin I. E., Konshin I. N.: Parallel solution of large sparse SPD linear systems based on overlapping domain decomposition. Lecture Notes in Computer Science, Vol. 1662, Springer, 436–445 (1999)
9. Kaporin I. E., Konshin I. N.: A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems. Numer. Linear Algebra Appl. V. 9, N. 2, 141–162 (2002)
10. Konshin I., Olshanskii M., Vassilevski Yu.: ILU preconditioners for nonsymmetric saddle-point matrices with application to the incompressible Navier-Stokes equations. SIAM J. Sci. Comp., V. 37, N. 5, A2171–A2197 (2015)