Research Article

Speeding up the ODETTA code for solving particle transport problems*

Anastasiya V. Shoshina[1], Viktor I. Belousov[2]

1 *National Research Nuclear University MEPhI, 31 Kashirskoye shosse, Moscow, 115409, Russia*
2 *NRC Kurchatov Institute, 1 Akademika Kurchatova Sq., Moscow, 123182, Russia*

Corresponding author: Viktor I. Belousov (Danilenko_L@bsu.edu.ru)

Academic editor: Georgy Tikhomirov ♦ **Received** 17 February 2020 ♦ **Accepted** 6 December 2020 ♦ **Published** 24 March 2021

Citation: Shoshina AV, Belousov VI (2021) Speeding up the ODETTA code for solving particle transport problems. Nuclear Energy and Technology 7(1): 15–20. https://doi.org/10.3897/nucet.7.64365

Abstract

Mathematical simulation of fast neutron reactors requires high-precision calculations of protection problems based on unstructured meshes. The paper considers and analyzes a parallel version of the ODETTA code (Belousov et al. 2019) with the use of the MPI (Message Passing Interface) library technology (Knyazeva et al. 2006). The code is designed for numerical simulation of neutronic processes in shielding compositions of fast neutron lead cooled reactor plants in normal operating modes, and can be used to calculate the radiation conditions of using structural components and equipment of nuclear power facilities which are assumed to be the sources of and/or exposed to ionizing radiation during their safety justification. The operation of the generated code is compared against the previous version. The MPI-based development of the ODETTA code's algorithmic part is described. Peculiarities and specific features of the code parallelization are presented, the code modification is given, and respective algorithms are considered. The structure of the ODETTA code based on the MPI is described in brief. The results of using the ODETTA code's serial and parallel versions in OS Linux (Kostromin 2012) for NRNU MEPhI's HPC cluster are provided (Savchenko et al. 2020). A comparative analysis is presented for two code implementation options in terms of speed and accuracy of results when using two different clusters and different numbers of nodes for these. Peculiarities of cluster-based calculations are noted.

Keywords

Parallel programming, MPI, ODETTA code, finite element method, radiation safety, HPC cluster

Introduction

The purpose of the study is to use parallel computations for solving problems of neutron and gamma quanta transport in a multi-group SnPm approximation by the finite element method based on unstructured tetrahedral meshes, including mesh data handling. The study is conducted as part of an investigation by the Nuclear Safety Institute of the Russian Academy of Sciences (IBRAE) at MEPhI's computation center. Extremely complex problems are involved in mathematical simulation of fast neutron reactors. It is exactly the solution of such problems that requires high-precision mass calculations of protection problems based on unstructured meshes. An analog is the ATTILA code (McGhee et al. 2007) which uses the discontinuous finite element method (DFEM) to solve transport equations based on spatial meshes consisting of tetrahedral and hexagonal finite elements with nonplanar faces.

* Russian text published: Izvestiya vuzov. Yadernaya Energetika (ISSN 0204-3327), 2020, n. 4, pp. 130–141.

Peculiarities of the ODETTA code

Mathematical simulation of fast neutron reactors requires highly precise calculations of protection problems using unstructured meshes. The ODETTA code in the neutronic calculation implementation module simulates the solution of the transport equation by the finite element method (FEM) based on unstructured tetrahedral meshes in a discrete ordinate method approximation (Sychugova and Seleznev 2014).

With the use of discrete ordinates, the transport equation is written as

$$\mu_m \frac{\partial \Psi_m}{\partial x} + \eta_m \frac{\partial \Psi_m}{\partial y} + \xi_m \frac{\partial \Psi_m}{\partial z} + \Sigma_T(x,y,z)\cdot\Psi_m(x,y,z) = Q_m(x,y,z),$$

where the group index number *g* is omitted, and the index *m* ($m = 1, 2, \ldots, M$) is matched by the discrete direction $\mathbf{\Omega}_m$ = (m$_m$, h$_m$, x$_m$) out of a quadrature set (with equal weights ES*n* or Chebyshev-Legendre's CL*n*), the unit sphere surface value being measured in 4π, i.e. $\Sigma\omega_m = 1$. FEM formulas are developed by approximating the transport equation according to Galerkin using the weighted residuals method. The equation includes the full macroscopic interaction cross-section Σ_T and the Q_m function, the right-hand part of the transport equation depending on the solution of Ψ_m. The right-hand part includes the source of intergroup and intragroup transitions, the source of fissions, and the given internal source. Zero values of the angular flow are given on the boundary G of the considered 3D region for directions inside of the region or the reflection condition. This results in a boundary problem for solving the equation of particle transport in a convex 3D region.

The anisotropic scattering is represented by a series expansion in associated Legendre functions up to the fifth order. The spatial rebalance method is used to accelerate the convergence of internal iterations. The code has been developed in Fortran (standard Fortran 90 and later) using OpenMP parallelizing. The ODETTA code operation steps are as follows:

- preparation of mesh data;
- preparation of group macroscopic cross-sections;
- formation of the radiation source;
- neutronic calculation;
- calculation of functionals.

The SALOME code is used to handle CAD models and unstructured meshes (SALOME 2020). The CONSYST code with the BNAB-RF library is used to prepare group constants (Manturov 2017). The SIEGFRIED preprocessor developed at IBRAE specifically for the ODETTA code, including such modules as Readconst (the interface between the NISN multigroup constant storage format and the ODETTA code internal format), Source (formation of the radiation source in the original geometry and interpolation in the unstructured mesh nodes), MKE3D (neutronic calculation module), and RECONST (post-processing, calculation of functionals), is used to ensure that the user can handle the ODETTA code in the most comfortable way.

The solution obtained after the ODETTA simulation can be visualized by a 3D graphic plotter, e.g., the VisIt program (About VisIt 2020)).

The spatial region under consideration is decomposed into a finite number of elements with the fixed number of endpoints referred to as nodes. The accuracy of the results depends greatly on the decomposition quality. Tetrahedrons are used as elements for the calculation. They have common nodal points and, taken together, approximate the region shape. Normally, decomposition is started from the region boundaries so that to obtain as accurate approximation of the region shape as possible (Fig. 1).

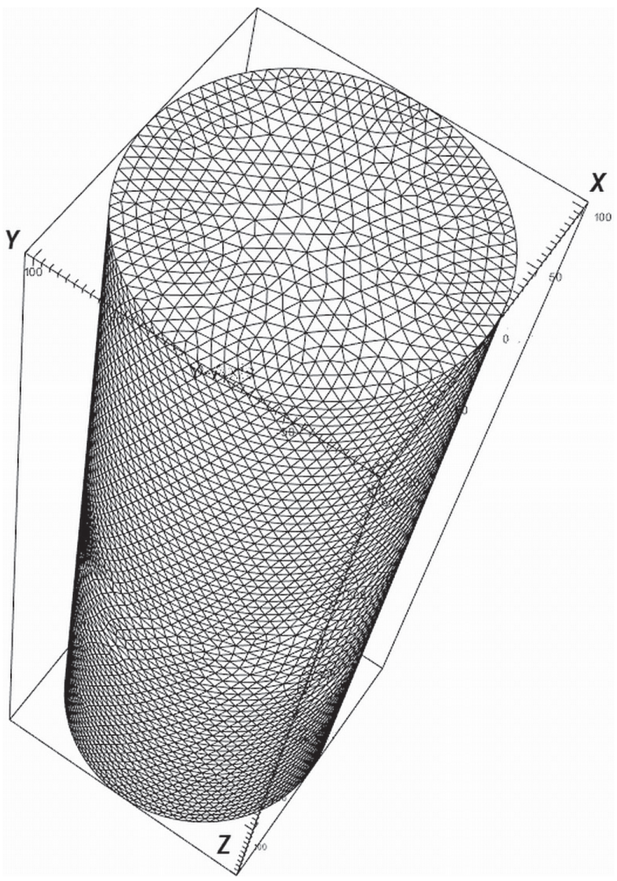

Figure 1. An example of the region decomposition into a finite number of tetrahedrons.

The considered region and the tetrahedrons can be decomposed by directions. The unit sphere of directions (an angular variable mesh) is decomposed into eight octants (Fig. 2a). Since calculation in each of these can be done independently, the use of parallel calculations will speed up noticeably the program execution. This forms the basis

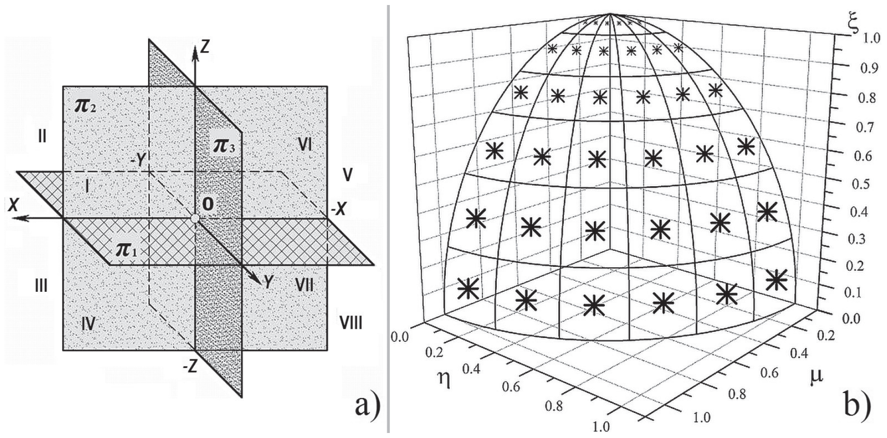

Figure 2. A unit sphere of directions decomposed into octants (**a**); a unit sphere decomposed by directions in the first octant (**b**).

for using the MPI in the neutronic calculation module of the ODETTA code.

Decomposition by directions (Fig. 2b) is used for the calculation in each octant – this is illustrated with an example of the first octant. Such space fragmentation is required to speed up the ODETTA neutronic calculation using OpenMP (OpenMP 2020). Decomposing a unit sphere of directions into octants and the independence of computations in each of these enable calculations in parallel and independently for each direction of the given space using the MPI. Thus, parallelizing through the MPI was used between octants, and that through OpenMP was used inside of octants.

Developing the algorithmic part of the MPI application for the ODETTA code

The growing interest in parallel programming these days is explained by the transition to mass production of multicore architectures (Nemnyugin 2007). The MPI technology is a set of utilities and library functions (for such languages as C/C++ (Pavlovskaya 2003) and Fortran (Ryzhikov 2000, Barteniev 2000, Chapman Stephen 2018, Shterenlikht 2018, Building Programs with GNU 2020)) which make it possible to create and start applications operating in parallel computation units of a different nature.

The ODETTA code uses a Fortran source code like most of the codes currently used in nuclear industry. Many of them cannot use more than one process, so they require conversion to parallel computations to speed up their operation. There are two ways to do this.

The first one is to use a compiler in which case the developer needs to indicate the code region where exactly parallelizing should be used inside of the program text. Such approach is used in the OpenMP system but is possible only in shared memory systems. This parallelizing option was introduced earlier in the ODETTA code.

The second approach suggests that the developer as such specifies the distribution of and communication among processes in the program code. The second approach was used in the new version of the ODETTA code. This section describes the alterations made to the ODETTA source code for its new implementation with the MPI technology.

Decomposing the space into eight octants, the calculation in which can be done independently, makes it possible to use several computation nodes to optimize the code runtime. The ODETTA calculation algorithm is executed in parallel, and, theoretically, the more nodes are used per cluster, the shorter is the problem simulation time. The program's source code used one computation node, and the octant calculation was done in series.

The resultant program was tested using two and four nodes of MEPhI's cluster, though the eight-node cluster was assumed to be the best option. The use of the MPI cuts the calculation time considerably since all computations are done in parallel. Different numbers of nodes and their respective distributions of processes are shown in Fig. 3.

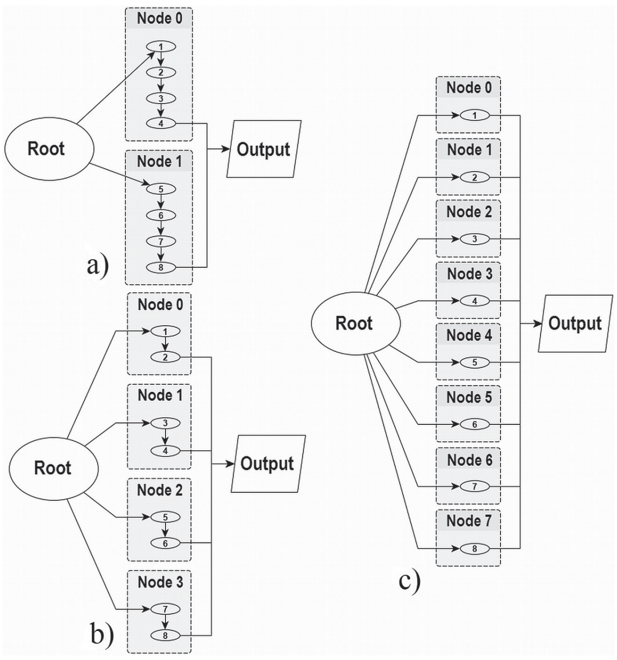

Figure 3. Distribution of octants by computation nodes: **a)** two nodes; **b)** four nodes; **c)** eight nodes.

Computations were supposed to be done using eight octants, so a maximum of eight processes, distributed by the nodes requested on the cluster, were used for the code implementation. The parallel computation capability is expected to provide a distinct advantage in terms of the calculation speed.

Directives controlling the compiler operation were engaged in the ODETTA code implementation using the MPI, this making it possible to extend the code capabilities.

Design and implementation

The final ODETTA algorithm using the MPI is represented by a block diagram with standard symbols commonly used in structured programming, including oval (the beginning or end of the considered program unit), rectangle (operations block), hexagon (the algorithm cycle containing the "body" of repetitive operations, one input, and one output (Fig. 4).

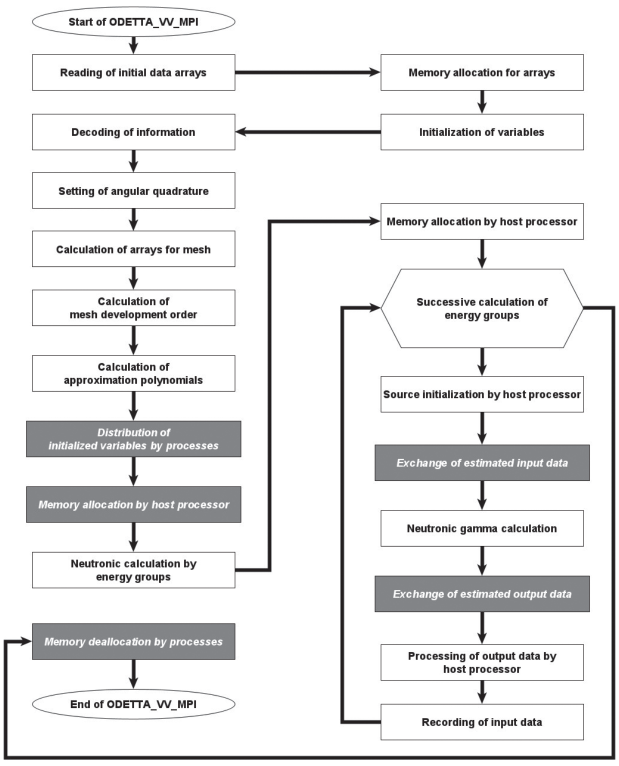
Start of ODETTA_VV_MPI
Reading of initial data arrays
Memory allocation for arrays
Decoding of information
Initialization of variables
Setting of angular quadrature
Memory allocation by host processor
Calculation of arrays for mesh
Successive calculation of energy groups
Calculation of mesh development order
Calculation of approximation polynomials
Source initialization by host processor
Distribution of initialized variables by processes
Exchange of estimated input data
Memory allocation by host processor
Neutronic gamma calculation
Neutronic calculation by energy groups
Exchange of estimated output data
Memory deallocation by processes
Processing of output data by host processor
End of ODETTA_VV_MPI
Recording of input data

Figure 4. MPI-based ODETTA diagram.

The MPI use blocks in the implemented algorithm are highlighted with a grey background and an White Italic text. Changes pertain largely to the use of data in the interaction between parallel processes.

Results

A radiation safety problem, which modeled a fuel assembly of the MOX-1000 fast neutron reactor benchmark model with mixed oxide uranium-plutonium fuel and sodium coolant (NEA/NSC/R(2015)9 2016) was used as the test problem for the ODETTA code. The assembly was withdrawn after staying in the operating reactor and put into a cylindrical steel container (shell) which was then filled with lead. The common calculation simulation elements are shown in Table 1.

Table 1. Common calculation pattern using the ODETTA code

Stage	Simulated item	Problem solved	Input data	Output data
Calculation of shielding	Fuel assembly in a steel container (shell) filled with lead	Non-homogeneous problem with given source	299-group source of neutrons, 127-group photon source, geometrical and physical parameters	Full neutron flux, full gamma quanta flux, calculation time
Comparison of results	ODETTA with MPI and without MPI			

It was assumed for simplifying the model calculation that the cylindrical container shell was made of HT-9 steel. SnPm approximation with the parameters n = 12 and m = 3 was used for the problem simulation which means that there is a total of 288 directions for the unit sphere or 36 directions per one octant (CL12 Chebyshev-Legendre quadrature). The number of energy groups for neutrons is 299 and of those for gamma quanta is 127. The number of tetrahedrons in the decomposition mesh is 132883.

The software for the ODETTA code was implemented at MEPhI's high-performance computation center the clusters of which are Linux-based. The remote computer is controlled through the command line and using the SSH protocol through the Putty program (Putty Documentation 2020). Basov and Cherenkov clusters (Savchenko et al. 2020) with different numbers of nodes were used to calculate the radiation safety test problem using the ODETTA code. The results are presented in Tables 2 and 3.

Table 2. Basov-cluster implementation of ODETTA code

Number of processes	1	2	3	4
Time, h	10.036	8.922	10.824	9.372
Speed up	–	1.125	0.927	1.071
Efficiency	–	0.562	0.309	0.268

Table 3. Cherenkov-cluster implementation of ODETTA code

Number of processes	1	2	3	4
Time, h	13.435	8.809	12.103	9.579
Speed up	–	1.525	1.110	1.403
Efficiency	–	0.763	0.370	0.351

The speed up $S_p = t_1/t_p$ and the efficiency $E_p = S_p/p$ for the parallel algorithms (t_1 is the algorithm execution time for one process, t_p is the algorithm execution time for a system of p processes) are determined depending on the number of processes.

The radiation safety test problem was calculated using computational nodes numbering one to four, the use of one node being equivalent to the problem simulation without the use of the MPI, and the number of OpenMP processes for each process was equal to 32, that is, the maximum one for one node. Due to the cluster overload, there was no eight-node computation, which had the maximum efficiency of problem solving in terms of time. Since the clusters use different systems of processors, it can be noted that the problem runtimes are different with the best one being in the event of the Basov cluster, as the program runtime with four nodes is longer than with two due to the specific data transmission between the cluster nodes. The ODETTA algorithm with the MPI requires exchange of large data arrays between the cluster nodes, so data transmission leads to an additional calculation delay. An additional speed up was however expected with eight cluster nodes thanks to processors which would minimize the costs of the node data exchange thanks to their speed.

A comparative analysis for the accuracy of computing full fluxes of particles is presented in Tables 4 and 5 where the calculation with one node, that is, without the use of the MPI, was assumed to be the reference result. The tables also present the maximum estimated relative error d$_{max}$ and the root-mean-square deviation s. The following relations were used to compute the errors

$$\delta_{max} = \max(\delta_1, \delta_2, ..., \delta_n),\ \delta_i = 1 - | x_i/x^*_i | = | \Delta x_i / x^*_i |,$$

where n is the number of the solution points; x_i is the calculated solution; x^*_i is the reference solution; and $\Delta x_i = x_i - x^*_i$. The root-mean-square deviation from the reference solution is

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} \Delta x_i^2}\ .$$

The relative error during the calculations with two and four nodes do not exceed 0.00155% when the Basov cluster is used and 0.00204% when the Cherenkov cluster is used, this indicating a minor deviation from the reference result.

The best option for obtaining the ODETTA cluster-based computation results is the use of one node with two MPI processors and 16 OpenMP processors installed for each process. This is possibly explained by the peculiarities of the cluster node structure which allow the program parallelizing algorithms to achieve the maximum speed up in terms of the calculation rate. The advantages of this computation option are shown in Tables 6 and 7.

Table 4. Comparison based on calculation results for full particle fluxes with the Basov cluster

	Number of processes	2	3	4
Neutrons	δ_{max}, %	$1.26 \cdot 10^{-3}$	$9.31 \cdot 10^{-3}$	$1.55 \cdot 10^{-3}$
	σ	$4.52 \cdot 10^{5}$	$1.48 \cdot 10^{5}$	8.21E$\cdot 10^{5}$
Gamma quanta	δ_{max}, %	$1.49 \cdot 10^{-3}$	$1.07 \cdot 10^{-3}$	$1.45 \cdot 10^{-3}$
	σ	$2.39 \cdot 10^{4}$	$7.57 \cdot 10^{3}$	$3.92 \cdot 10^{4}$

Table 5. Comparison based on calculation results for full particle fluxes with the Cherenkov cluster

	Number of processes	2	3	4
Neutrons	δ_{max}, %	$2.03 \cdot 10^{-3}$	$1.48 \cdot 10^{-3}$	$1.49 \cdot 10^{-3}$
	σ	$1.82 \cdot 10^{6}$	$1.56 \cdot 10^{6}$	$1.99 \cdot 10^{6}$
Gamma quanta	δ_{max}, %	$2.04 \cdot 10^{-3}$	$1.47 \cdot 10^{-3}$	$1.50 \cdot 10^{-3}$
	σ	$1.10 \cdot 10^{5}$	$7.84 \cdot 10^{4}$	$1.09 \cdot 10^{5}$

Table 6. ODETTA implementation using one node with two MPI processors

Cluster	Basov	Cherenkov
Time, h	6.951	7.029
Speed up	1.444	1.911
Efficiency	0.722	0.956

Table 7. ODETTA computation characteristics using one node with two MPI processors

	Cluster	Basov	Cherenkov
Neutrons	δ_{max}, %	$1.34 \cdot 10^{-3}$	$1.80 \cdot 10^{-3}$
	σ	$3.16 \cdot 10^{5}$	$1.75 \cdot 10^{6}$
Gamma quanta	δ_{max}, %	$1.25 \cdot 10^{-3}$	$1.96 \cdot 10^{-3}$
	σ	$1.69 \cdot 10^{4}$	$9.65 \cdot 10^{4}$

The relative error during the calculations with one node using two MPI processors does not exceed 0.00134% when the Basov cluster is used and 0.00196% when the Cherenkov cluster is used, this indicating high precision of the obtained results.

Conclusion

The MPI technology was used to speed up the ODETTA code for solving the neutron and gamma quanta transport problem in a multigroup SnPm approximation by the finite element method based on unstructured tetrahedron meshes. The results of the source code modification were tested using a radiation safety problem as an example.

The program modifications were tested using a radiation safety problem (Bereznev et al. 2017). To check the accuracy of the obtained results, these were compared with the results of the ODETTA serial version which were assumed to be standard. The deviation from the standard was about 0.002% which indicates that the MPI parallelizing algorithm has been implemented correctly.

When analyzing the starts of computations using different numbers of the MPI nodes and 32 maximally possible OpenMP processes, a conclusion can be made that the most efficient distribution was that by two Cherenkov cluster nodes where the speed up was about 1.52. The calculation was slower with four computation nodes used than with two, this being explained by the peculiarities of the algorithm and the cluster architecture, that is, using the Hyper-threading technology (Savchenko et al. 2020) for the Basov and the Cherenkov, which makes it possible to use two computation threads in one physical processor core. The increase in the efficiency varies among applications. The execution of some programs can slow down as was the case in this study. A possible reason is connected with the “system of repetitions” of the Xeon processors which occupies the required computational resources, leading to the idle state of other computation threads.

As the result, the use of one node with two MPI processors and 16 OpenMP processors was the best option for the ODETTA cluster operation. The maximum speed up achieved with the Cherenkov cluster was 1.911 (nearly double). The maximum time gain amounted so to about 48%. In turn, the minimum time gain was about 30% as compared even with the best one-processor calculation.

It should be additionally noted that the ODETTA algorithm with the use of the MPI requires large arrays of data to be exchanged between the cluster nodes, so the transmission of data led to a major calculation delay. The use of several processors has a negative impact on speed due to an extra interprocessor interaction, but the calculation time remains much smaller as compared with the serial version.

References

- About VisIt (2020) VisIt Home. https://wci.llnl.gov/simulation/computer-codes/visit [accessed Feb 10, 2020]
- Barteniev OV (2000) Modern Fortran. 3rd ed. Dialog MIFI Publ., Moscow, 449 pp. [in Russian]
- Belousov VI, Bereznev VP, Seleznev YeF (2019) ODETTA computational code for solving neutron and gamma quanta transport problems in a multigroup SnPm approximation by the finite element method based on unstructured tetrahedral meshes, including mesh data handling. Ver. 2.1. (ODETTA). Computer Program Validation Certificate, Reg. No. 497, Dec. 19. FBU NTTs YaRB Publ., Moscow, 6 pp. [in Russian]
- Bereznev VP, Belousov VI, Grushin NA, Seleznev EF, Sychugova EP (2017) New neutronic codes based on the discrete ordinates method using methods of finite differences and finite elements. In: Proceedings of the International Conference on Fast Reactors and Related Fuel Cycles: Next Generation Nuclear Systems for Sustainable Development (FR17). Paper CN-195. Yekaterinburg. ROSATOM Publ., 10 pp.
- Building Programs with GNU (2020) Building Programs with GNU Make. http://coderway.ru/cpp/make [accessed Feb 10, 2020] [in Russian]
- Chapman Stephen J (2018) Fortran for Scientists and Engineers. 4th edn. BAE Systems Australia, 1049 pp.
- Knyazeva MA, Molchanova LA, Tarasov GV (2006) Parallel Programming. Dalnevostoch-ny Universitet Publ., Vladivostok, 61 pp. [in Russian]
- Kostromin VA (2012) Linux Tutorial for User. BKhV-Peterburg Publ., St. Petersburg, 672 pp. [in Russian]
- Manturov GN (2017) Procedural Constant and Software for Neutronic Calculations of Fast Reactors and Estimation of Computational Prediction Errors. Dr. Tech. Sci. Diss. SSC RF-IPPE Publ., Obninsk, 202 pp. [in Russian]
- McGhee JM, Wareing TA, Barnett DA (2007) Attila User's Manual. Transpire Inc., Jan 15, 1077 pp.
- NEA/NSC/R(2015)9 (2016) Nuclear Science Committee. Benchmark for Neutronic Analysis of Sodium-cooled Fast Reactor Cores with Various Fuel Types and Core Sizes, 25-Feb. https://www.oecd-nea.org/science/docs/2015/nsc-r2015-9.pdf [accessed Feb 10, 2020]
- Nemnyugin SA (2007) Programming Tools for Multiprocessor Computing Systems. Intel Multicore Curriculum Initiative. St. Petersburg State University Publ., St. Petersburg, 88 pp. [in Russian]
- OpenMP (2020) OpenMP Application Program Interface. http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf [accessed Feb 10, 2020]
- Pavlovskaya TA (2003) C/C++. High-Level Language Programming. Piter Publ., St. Petersburg, 461 pp. [in Russian]
- Putty Documentation (2020) Putty Documentation. https://putty.org.ru/docs.html [accessed Feb 10, 2020] [in Russian]
- Ryzhikov YuI (2000) Fortran Powerstation Programming for Engineers. A Practical Guide. Korona-Print Publ., St. Petersburg, 161 pp. [in Russian]
- SALOME (2020) SALOME – The Open Source Integration Platform for Numerical Simulation. https://www.salome-platform.org [accessed Feb 10, 2020]
- Savchenko AV, Anikeyev AA, Okunev DYu (2020) High-Performance Computing Center of NRNU MEPhI. User Manual. NRNU MEPhI Publ., Moscow, 24 pp. https://ut.mephi.ru/pdf/projects/hpc/userguide.pdf [accessed Feb 10, 2020] [in Russian]
- Shterenlikht A (2018) Parallel Programming with Fortran 2008 and 2018 Coarrays. Mech. Eng. Dept., The University of Bristol, Bristol BS8 1TR, 27 pp.
- Sychugova YeP, Seleznev YeF (2014) The Finite Element Method for Solving the Transport Equation based on Unstructured Tetrahedral Meshes. Preprint No. IBRAE-2014-03. IBRAE RAN Publ., Moscow, 21 pp. [in Russian]