INFORMATICA, 2016, Vol. 27, No. 1, 161–178

DOI: http://dx.doi.org/10.15388/Informatica.2016.80

On Efficiency Analysis of the OpenFOAM-Based Parallel Solver for Simulation of Heat Transfer in and Around the Electrical Power Cables

Vadimas STARIKOVIČIUS[1]*, Raimondas ČIEGIS[2], Andrej BUGAJEV[2]

[1]*Laboratory of Parallel Computing, Vilnius Gediminas Technical University*
Saulėtekio 11, LT-10223 Vilnius, Lithuania

[2]*Department of Mathematical Modelling, Vilnius Gediminas Technical University*
Saulėtekio 11, LT-10223 Vilnius, Lithuania

e-mail: vadimas.starikovicius@vgtu.lt, raimondas.ciegis@vgtu.lt, andrej.bugajev@vgtu.lt

Received: July 2015; accepted: November 2015

Abstract. In this work, we study the efficiency of developed OpenFOAM-based parallel solver for the simulation of heat transfer in and around the electrical power cables. First benchmark problem considers three cables directly buried in the soil. We study and compare the efficiency of conjugate gradient solver with diagonal incomplete Cholesky (DIC) preconditioner, generalized geometric-algebraic multigrid GAMG solver from OpenFOAM and conjugate gradient solver with GAMG multigrid solver used as preconditioner. The convergence and parallel scalability of the solvers are presented and analyzed on quadrilateral and acute triangle meshes. Second benchmark problem considers a more complicated case, when cables are placed into plastic pipes, which are buried in the soil. Then a coupled multi-physics problem is solved, which describes the heat transfer in cables, air and soil. Non-standard parallelization approach is presented for multi-physics solver. We show the robustness of selected parallel preconditioners. Parallel numerical tests are performed on the cluster of multicore computers.

Key words: OpenFOAM, parallel algorithms, domain decomposition, preconditioner, multigrid, multi-physics problem.

1. Introduction

The control of heat conduction processes in high-voltage electrical cables is a very important task in operation of electricity transferring infrastructure. Thus a detailed knowledge of heat generation and distribution in and around the cables is necessary to optimize the design and exploitation of such infrastructure. Engineers are interested in maximum allowable current in different conditions, optimal cable parameters, cable life expectancy estimations and many other engineering factors.

Presently used IEC (International Electrotechnical Commission) standards for the design and installation of electrical power cables are based on the worst case analysis of

*Corresponding author.

simplified models (Neher and McGrath, 1957). Such models (mostly given in the form of analytical formulas) cannot accurately account for the various conditions under which the cables are actually installed and used. They estimate the cable's current-carrying capacity with significant margins to stay on the safe side (Makhkamova, 2011). More accurate mathematical models and simulation tools are needed to meet the latest technical and economical requirements and to define new, cost-effective design rules and standards.

When we need to deal with mathematical models for the heat transfer in various media (metals, insulators, soil, water, air) and non-trivial geometries, only parallel computing technologies can allow us to get results in an adequate time. To solve numerically selected models, we are developing our numerical solvers by using the OpenFOAM (Open source Field Operation And Manipulation) package (OpenFOAM, 2015). OpenFOAM is a free, open source CFD (Computational Fluid Dynamics) software package. It has an extensive set of standard solvers for popular CFD applications. At the same time the user has a possibility to implement modified nonstandard mathematical models, new numerical schemes and algorithms, utilizing the rich set of OpenFOAM capabilities (Weller *et al.*, 1998). However, application of OpenFOAM libraries for solving specific multi-physics problems still requires an appropriate theoretical and empirical analysis and a nontrivial selection of optimal numerical algorithms. Examples of such problems are described in Higuera *et al.* (2013) and Petit *et al.* (2011).

The important consequence of this software development approach is that obtained application solvers can automatically exploit the parallel computing capabilities already available in the OpenFOAM package. Parallelization of solvers in OpenFOAM is based on the domain decomposition method and MPI (Message Passing Interface) standard. However, the modular structure of this package allows the development of parallel applications for modern hybrid and heterogeneous HPC (High-Performance Computing) platforms.

Due to the growing popularity of OpenFOAM package, the number of various OpenFOAM-based applications in different fields is constantly increasing. The most important challenges for parallel solvers implemented with OpenFOAM are scalability and efficiency on new hybrid and heterogeneous parallel computing systems. There are different attempts to reduce the MPI communication overhead and increase the scalability of parallel solvers by making use of threads parallelism on multicore cluster systems (Liu, 2011). However, MPI+OpenMP (Open Multi-Processing) hybridization does not show any significant improvement so far (Culpo, 2012; Dagna and Hertzer, 2013).

For hybrid GPU (graphics processing unit) + CPU (central processing unit) parallel computing systems, modular design of OpenFOAM solvers allows to transfer solution of linear systems to GPUs (AlOnazi, 2013). Various open and closed source GPU linear algebra libraries provide coupling with OpenFOAM. See, for example, Cufflink (Cuda For FOAM Link) library (Cufflink, 2012) as interface to CUSP (2015) and Thrust (2015) libraries.

In Rivera *et al.* (2011) it is noted that the scalability of parallel OpenFOAM solvers is not very well understood for many applications when executed on massively parallel systems. We note that an extensive experimental scalability analysis of selected OpenFOAM applications is one of the tasks solved in PRACE (Partnership for Advanced Computing in Europe) project, see Culpo (2012), Dagna and Hertzer (2013), Dagna (2012).

In Dagna (2012) are presented results on IBM BlueGene/Q (Fermi) and Hewlett Packard C7000 (Lagrange) parallel supercomputers for a few CFD applications with different multi-physics models. The presented experimental results for selected OpenFOAM applications are showing a good scaling and efficiency with up to 2048–4096 cores. It is noted that such results are expected when balancing between computation, message passing and I/O work is good. Obviously, the next generation of ultrascale computing systems will cause additional challenges due to their complexity and heterogeneity. Hence the importance of proper workload balancing will only increase.

Another challenge for the efficiency of parallel solvers is causing the fact that most of the newly developed applications are dealing with multi-physics. For mathematical models describing coupled multi-physics problems, it is important to investigate two different approaches to design robust and efficient solvers for such problems (Muddle *et al.*, 2012). In coupled approach, monolithic solvers operate directly on one system of nonlinear algebraic equations, obtained after the discretization of system of all PDEs (Partial Differential Equations). In the partitioning approach, the discrete system is solved by using the single-physics solvers in decoupled fixed-point iterations. The latter approach is natively supported in OpenFOAM and used in multi-region solvers. A good review and comparison of some popular fixed-point methods is given in Kuettler and Wall (2008).

It is well known that for many real world applications, which are described by the systems of nonlinear PDEs, the biggest part of CPU time is used to solve very large systems of linear equations with sparse matrices. Thus the quality and scalability of parallel preconditioning algorithms has the major impact on the scalability and efficiency of whole parallel solver. It is clear that for engineering and scientific community it is important to know the properties of parallel preconditioning algorithms and their implementations, which are available in widely used parallel numerical libraries, such as PETSc (2015), Trilinos (2015) and OpenFOAM.

As already mentioned, scalability studies of some of the standard OpenFOAM application solvers can be found in the literature (Rivera *et al.*, 2011; Duran *et al.*, 2015). In this work, we present the parallel performance analysis of our own OpenFOAM-based solver for simulation of the heat transfer in and around the electrical power cables. We gradually develop and test our solver for real multi-physics problem: coupled heat transport in cables, soil and air. The parallel single-region solver for heat transfer was developed and studied in Čiegis *et al.* (2014). In our work, we adapt a non-standard for OpenFOAM coupling approach. Namely, one single temperature equation is solved in numerical solution scheme for the whole simulated domain, i.e. it is not split between the different regions like it is done in OpenFOAM multi-region solvers. The remaining processes (e.g. air flow) are simulated using proper solvers in according local regions.

The main contribution of this paper is a comprehensive study of performance of parallel preconditioners. For our study, we have selected three state of the art combinations of linear system solvers and preconditioners, commonly used in commercial and open source simulation software for the heat transfer problems. Namely, we are testing and comparing the conjugate gradient solver with diagonal incomplete Cholesky (DIC) preconditioner, which was already studied in Čiegis *et al.* (2014), the generalized geometric-algebraic

multigrid GAMG solver, which is available in OpenFOAM library, and the conjugate gradient solver with GAMG solver used as preconditioner. We study the convergence and parallel scalability of the linear solvers, the sensitivity of parallel preconditioners with respect to the mesh size and the number of parallel processes.

Two different types of 2D space meshes are investigated. Quadrilateral meshes are generated by using OpenFOAM native meshing tools. They lead to classical five-point stencils in approximation of diffusion operators. Acute triangle meshes are generated by Acute meshing library (Erten and Ungor, 2009) and converted to OpenFOAM mesh format. Diffusion operators are approximated on triangle meshes by non-classical four-point stencils. Our aim is to study the sensitivity of selected preconditioners to the changes in grid stencils used in numerical approximation.

Our parallel performance tests demonstrate the quality of domain decomposition methods available in OpenFOAM. We study the ability of OpenFOAM-based parallel solvers to efficiently deal with the heterogeneity of computer cluster, to provide a proper workload balancing in case of complicated geometries and multi-physics problem.

In most applications of mathematical modelling technique for simulation and analysis of various industrial and technological processes and devices, the last and most important part of the project consists in implementation of an optimization step. Next, we only outline the most important aspects and challenges of this step.

The goal of any global optimization algorithm is to find the best possible element in a search space according to a given objective function. In the case of industrial problem, this task is very nontrivial and requires to solve some specific challenges. An extensive review on the parallel optimization algorithms used for solving industrial problems is presented in Starikovičius *et al.* (2011). Here, we restrict to mentioning only the most important aspects.

First, in order to compute one single value of the objective function usually we need to solve large size discrete problems resulting from the systems of 2D/3D nonlinear PDEs, what requires large CPU time costs. Second, there is no possibility to get the standard information on the properties of the objective function, which is normally required to justify the convergence of optimization algorithms. Third, we are not concentrated on finding the exact global extremum. It is usually sufficient to get a solution which improves the known engineering approximation.

In Čiegis *et al.* (2008), a two level parallel Master–Slave (MS) template was used to implement optimization algorithms for simulation and optimization of electrical cables in automotive industry. One of the main tasks for engineers is to determine optimal conductor cross-sections in bundles of electric cables in order to minimize the total weight of cables. A simple heuristic algorithm based on a greedy type search method was used as the optimization technique.

In Starikovičius *et al.* (2011), it is shown how an already existing direct solver for the 3D simulation of flow through the oil filter is integrated into the developed MS template to obtain a parallel optimization solver. Some capabilities and performance of this solver are demonstrated by solving geometry optimization problem of a filter element.

It should be noted, that for solving optimization problems for real-world problems a very popular approach is to use parallel genetic and colony type algorithms, and various

modifications. Analysis of new parallel optimization algorithms is done in Lančinskas *et al.* (2015) and Filatovas *et al.* (2015). See also references therein.

In Section 2, we shortly describe the mathematical models selected for our heat transfer application and two benchmark problems used for parallel numerical tests. In Section 3, we describe our OpenFOAM-based solver and discuss the parallelization approach. In Section 4, we present and analyze the parallel performance results on convergence and scalability of the considered linear solvers and preconditioners for both benchmark problems. Finally, some conclusions are drawn in Section 5.

2. Mathematical Models and Benchmark Problems

As the first benchmark problem, we solve the heat conduction problem for electrical high power cables when the cables are directly buried in the soil. It is also assumed that the thermo-physical properties of the soil remain constant. The heat source is described by the Joule–Lenz law. Then the mathematical model of non-stationary heat transfer is described by the following mathematical model (Bergman and Incropera, 2011; Čiegis *et al.*, 2007):

$$\begin{cases} \rho c \dfrac{\partial T}{\partial t} = \nabla \cdot (\lambda \nabla T) + q, & t \in [0, t_{\max}],\ \vec{x} \in \Omega, \\ T(\vec{x}, 0) = T_b, & \vec{x} \in \Omega, \\ T(\vec{x}, t) = T_b, & \vec{x} \in \partial\Omega, \end{cases} \tag{1}$$

where $T(\vec{x}, t)$ is the temperature, $c(\vec{x}) > 0$ is the specific heat capacity, $\rho(\vec{x}) > 0$ is the mass density, $\lambda(\vec{x}) > 0$ is the heat conductivity coefficient, $q(\vec{x}, t, T)$ is the heat source function due to the power losses, T_b is the initial and boundary temperature. Coefficients $\lambda(\vec{x})$, $c(\vec{x})$, $\rho(\vec{x})$ are discontinuous functions. Their real values of these coefficients can vary between the metallic conductor, insulators and soil by the several orders of magnitude (Makhkamova, 2011).

Due to discontinuity of coefficients, the full formulation of the mathematical model also includes the following conjugation condition

$$T, \quad \lambda \nabla T \text{ are continuous } \vec{x} \in \Omega. \tag{2}$$

In this work, we have used 2D geometry for our benchmark problems. Three cables are buried in the soil in a flat formation as shown in Fig. 1.

The real geometry of the underground electric cable is very complex and contains the following components: the copper conductor core made of strands, the conductor screen (made of polyethylene type material), insulation (cross-linked polyethylene), the insulation screen (polyethylene type material), the copper wire screen, the binder tape (polyethylene type material) and oversheath (medium density polyethylene). To deal with computational meshes of reasonable sizes, the real geometries of the conductor and cooper wire screen are simplified to the inner cylinder and outer ring (annulus). They are shown in red in Figs. 1, 2 and 3. Such simplifications are very common in engineering simulations and

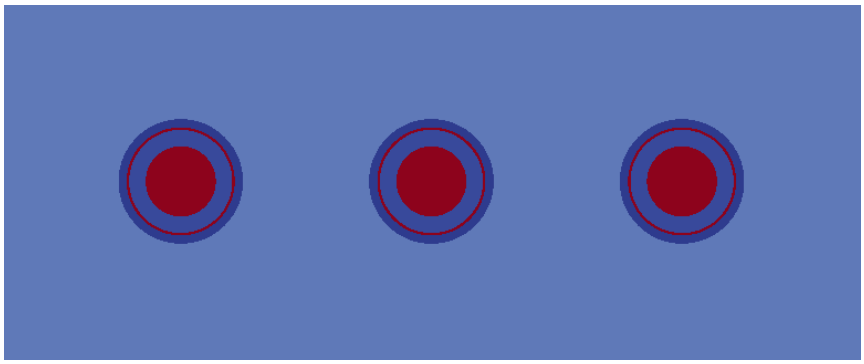
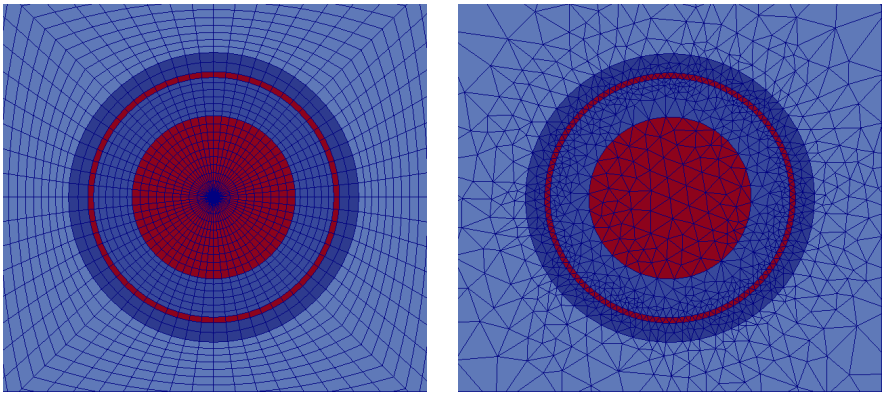


Fig. 1. First benchmark problem: three electric cables directly buried in the soil in a flat formation.



(a) quadrilateral mesh (b) acute triangle mesh

Fig. 2. A single cable geometry and computational mesh.

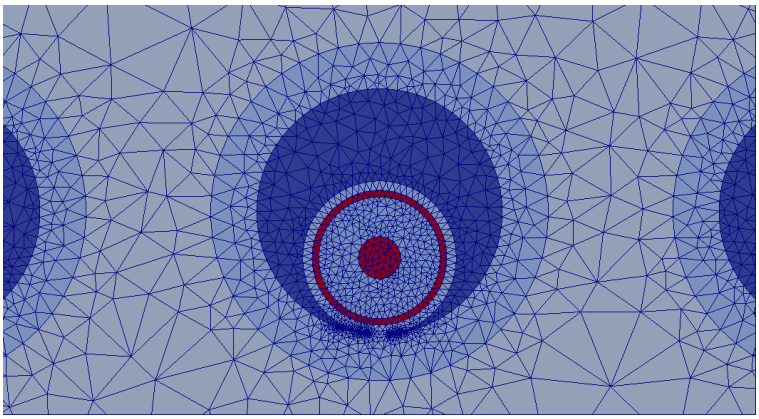


Fig. 3. Second benchmark problem: cables are placed into plastic pipes, which are buried in the soil.

they allow a significant reduction in the computational time of simulations with satisfactory accuracy.

In Fig. 2 we show the examples of 2D meshes that we will use for parallel performance evaluation tests in Section 4. Figure 2(a) shows a quadrilateral mesh, which was generated by using OpenFOAM native meshing tool – *blockMesh*. Figure 2(b) shows an acute triangle mesh, which was generated by Acute meshing library (Erten and Ungor, 2009) and converted to OpenFOAM mesh format.

For the first benchmark problem we have used geometry of the real 10 kV electrical power cable. The dimensions are as follows: the diameter of copper conductor is 9.8 mm; the thickness of insulation is 4.6 mm; the thickness of the copper screen is 0.7 mm and the thickness of oversheath is 2.3 mm. The distance between the centres of cables is 70 mm. The size of computational domain is 240 mm $\times$ 100 mm.

Second benchmark problem considers a more complicated case, when electrical power cables are placed into special containers – plastic pipes. Three such pipes with cables inside are buried in the soil in a flat formation as shown in Fig. 3.

In this case, inside the plastic pipes there is an air, which has a different main heat transfer mechanism compared to cooper, insulation and soil. The main heat transfer in the air is caused by free air circulation inside the plastic pipe, which is called natural (free) convection. In general case, buoyancy driven air flow is modeled considering air as compressible turbulent fluid. However, for relatively small velocities (less than 80–100 m/s) it is often assumed that flow is incompressible and laminar, viscous dissipation is negligible, air is Newtonian and constant-property Boussinesq fluid (so called Boussinesq approximation).

Then natural convection inside two-dimensional enclosure is governed by the Navier-Stokes and thermal energy conservation equations (Bergman and Incropera, 2011):

$$\nabla \cdot \vec{u} = 0, \tag{3}$$

$$\rho \frac{\partial \vec{u}}{\partial t} + \rho \vec{u} \nabla \cdot \vec{u} - \nabla \cdot (\eta \nabla \vec{u}) = -\nabla p - \rho \vec{g}\big(1 - \beta(T - T_0)\big), \tag{4}$$

$$\rho c \left(\frac{\partial T}{\partial t} + \nabla \cdot (\vec{u} T) \right) = \nabla \cdot (\lambda \nabla T), \tag{5}$$

where $\vec{u}$ is two-dimensional velocity of the air, p is air pressure, η is air viscosity, $\vec{g}$ is gravity acceleration vector, β is thermal expansion coefficients, T_0 is reference temperature.

To couple the air flow and heat transfer model (3)–(5) in air region with heat conduction model (1) in neighboring regions (cable's oversheath and container's pipe, see Fig. 3), we apply the no-flow and temperature continuity conditions (2) at the interfaces between these regions.

For the second benchmark problem we have used the cable of the following dimensions: the diameter of copper conductor is 9.25 mm; the thickness of insulation is 17 mm; the thickness of the copper screen is 2.9 mm and the thickness of oversheath is 4.05 mm. The inner radii o the container's pipe is 53.12 mm. The thickness of the pipe is 20 mm. The distance between the centres of cables is 200 mm. The size of computational domain is 3000 mm $\times$ 2000 mm.

3. Parallel OpenFOAM-Based Solver

OpenFOAM (2015) is a C++ toolbox (open source library) for the development of customized numerical solvers for partial differential equations. For numerical solution of PDEs OpenFOAM uses the Finite Volume Method (FVM) with co-located arrangement of unknowns (Weller *et al.*, 1998).

We obtain a numerical solver for our first benchmark problem (1) by starting from the the standard *laplacianFoam* solver and implementing the following modifications: adding variable problem coefficients $c(\vec{x})$, $\rho(\vec{x})$, $\lambda(\vec{x})$ and a nonlinear source term $q(\vec{x}, t, T)$ dependent on temperature T. A proper interpolation should be used for calculation of discontinuous coefficient $\lambda(\vec{x})$ on cooper-insulator and insulator-soil interface walls of according finite volumes (see Fig. 2). We employ the *harmonic* interpolation to ensure the continuity of heat flux $\lambda\nabla T$ according to condition (2).

To solve any problem with OpenFOAM, properly generated mesh of problem domain should be supplied to a solver. If we generate the mesh for domain Ω using OpenFOAM preprocessing tool *blockMesh*, we obtain quadrilateral finite volume mesh, see Fig. 2(a). FVM discretization of Eq. (1) on this mesh has the classical five-point stencil. If we generate mesh using Acute meshing library (Erten and Ungor, 2009), after conversion to OpenFOAM format we obtain acute triangle finite volume mesh, see Fig. 2(b). FVM discretization of Eq. (1) on triangle mesh has non-classical four-point stencil. In both cases OpenFOAM FVM discretization module produces systems of linear equations with symmetric matrices. These linear systems can be solved by preconditioned conjugate gradient method with various preconditioners and multigrid method.

Parallelization in OpenFOAM is robust and implemented at a low level using the MPI library. Solvers are built using high level objects and, in general, don't require any parallel-specific coding. They will run in parallel automatically. Thus there is no need for users to implement standard basic steps of any parallel code: decomposition of the problem into subproblems, distribution of these tasks among different processes, implementation of data communication methods. The most important drawback of this approach is that the user has very limited possibilities to modify the generated parallel algorithm if the efficiency of the OpenFOAM parallel code is not sufficient.

OpenFOAM employs the domain decomposition method for parallelization of numerical algorithms for solution of PDEs. The mesh and its associated fields are partitioned into sub-domains, which are allocated to different processes. Parallel computation of the implicit finite FVM algorithm requires two types of communication. First, the local communication is done between neighboring processes for the approximation of the Laplacian term on the given stencil and matrix-vector multiplication in iterative linear solvers. Second type is the global communication between all processes for computation of scalar products in iterative linear solvers.

OpenFOAM supports four methods of domain decomposition, which decompose the data into non-overlapping sub-domains: simple, hierarchical, scotch and manual (OpenFOAM, 2015). For our parallel tests the mesh is partitioned by using methods from the Scotch library (Chevalier and Pellegrini, 2008). Scotch is a library for graph and mesh

partitioning, similar to the well-known Metis library (Karypis and Kumar, 1999). No geometric input is required from the user and the decomposition method attempts to minimize the number of boundary edges between sub-domains. In addition, the user can specify the weights of the sub-domains, what can be useful on heterogeneous clusters of parallel computers with different performance of processors. We will use this feature to solve our benchmark problem on heterogeneous cluster.

The OpenFOAM-based multi-physics solver for our second benchmark problem is obtained in a non-standard way. In OpenFOAM, the standard approach is to generate separate meshes for different regions (air and solid = cable + soil) and to employ single-physics solvers coupled through interface boundary conditions using fixed-point iterations. We have developed a solver, which solves a single temperature equation in whole simulated domain using its mesh. On the mesh of air region we solve only Navier–Stokes equations (3)–(4) using the according numerical scheme from the OpenFOAM library.

However, this decision has its negative consequences. For solver to work properly in parallel mode, some additional modifications are needed. First, we require from *scotch* domain decomposition method to preserve all air regions of separate cables in single process domains, i.e. different processes cannot have "air" finite volumes from the same cable. Next, we require from all processes to solve whole problem on air region mesh, i.e. without decomposition. Finally, we create maps between finite volume indexes in air region mesh and whole domain mesh for all processes. Then we can copy the obtained velocity values from buoyancy driven air flow solver to heat conduction solver of global temperature equation in each multi-physics solver iteration without any additional MPI communications.

Obviously, such parallelization approach can be efficient only when CPU time spent in air flow solver is small compared to the time spent in temperature equation solver. Note that due to the huge influence of boundary conditions in heat conduction problems, usually we need to simulate much bigger problem area compared to the size of cable container. Our second benchmark problem has real dimensions for simulation in non-parallel mode to produce accurate temperatures. From the given dimensions, we see that air region makes up 0.27% of whole problem domain. Hence we can expect good performance results from the presented parallelization approach.

4. Parallel Performance Tests and Analysis

In this study we want to investigate and compare the convergence and parallel scalability of conjugate gradient linear solver with diagonal incomplete Cholesky preconditioner (DIC/CG), generalized geometric-algebraic multigrid solver (GAMG) and conjugate gradient solver with GAMG solver used as preconditioner (GAMG/CG). We want to study the scalability of those linear solvers and the influence of the mesh partitioning on parallel preconditioners. So, the tolerance of linear solvers is fixed and set to 10^{-6}. In each test, we do 10 time steps with different number of iterations, which are dependent on the convergence rate.

Parallel numerical tests were performed on the computer cluster "Vilkas" at the Laboratory of Parallel Computing of Vilnius Gediminas technical university. We have used eight nodes with Intel® Core™ i7-860 processors with 4 cores (2.80 GHz) and 4 GB of RAM per node. Computational nodes are interconnected via Gigabit Smart Switch.

It is well-known that the efficiency of parallel iterative solvers (and the quality of preconditioners) depend strongly on the properties of the adaptive mesh (geometrical issue) and on the distribution of coefficients of the different materials (material issue).

First, we have investigated the influence of the geometrical factor. We have used uniform problem coefficients in our first benchmark problem (1) to study the impact of geometry and different meshes on the convergence of linear solvers. Performance results are presented for all selected combinations of linear solvers and preconditioners with increasing finite volume mesh.

In Table 1, we present the total wall time T_p of 10 time steps, the average number of iterations N_p^{av}, the average time of one iteration $T_p^{av} = T_p/N_p^{av}/10$ obtained with p processes on quadrilateral mesh. Here $p = n_d \times n_c$ is the number of parallel processes using n_d nodes with n_c cores per node. The discrete time step was set to 0.1 seconds.

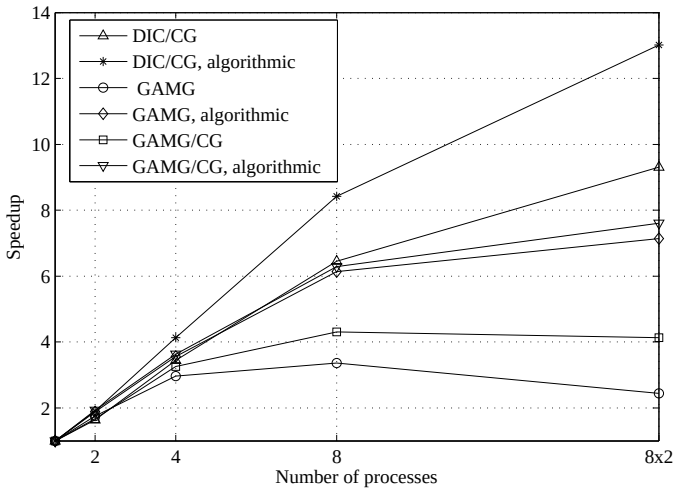
Note that the case $p = 1 \times 1$ provides data on convergence and elapsed wall time for the sequential algorithms. There are no data for the mesh size 8192000, because this case does not fit into memory available on the single node.

The following main conclusions can be done from presented experimental results.

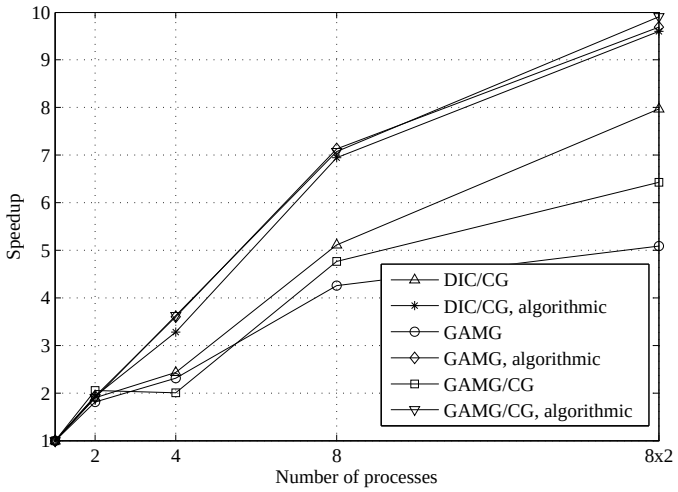
- The number of iterations of DIC/CG solver is increasing as $\sqrt{2}$ when the mesh size is doubled. This is in accordance with the theory. The number of iterations of GAMG solver is also growing with the mesh size, but not as fast as in the case of DIC/CG, compare $24.6/20.4 \approx 1.21$ to $1427/721.9 \approx 1.99$ (2 expected from theoretical analysis).
- Compared to the GAMG solver the GAMG/CG solver does almost two times smaller number of iterations. Our hypothesis is that the GAMG solver performs two iterations as a preconditioner per one conjugate gradient solver iteration in the GAMG/CG solver. Notably, this number (i.e. 2) is not accessible or modifiable in the standard user interface of OpenFOAM.
- In our tests, we see that the number of iterations of GAMG/CG solver is also slightly increasing with the increasing mesh size.
- Comparing the elapsed times T_p we see that the GAMG and GAMG/CG solvers are significantly faster than the DIC/CG solver for the test case with the constant (uniform) problem coefficients. The GAMG solver achieves the biggest advantage over the DIC/CG solver for sequential tests. For parallel tests, the advantage is constantly decreasing with increasing number of processes p. We note two reasons for this. First reason is different degradation in performance of the parallel preconditioner. For parallel CG method with DIC preconditioner the number of iterations is increasing quite gradually up to 40% going from 1 to 16 processes. At the same time the number of GAMG and GAMG/CG iterations exhibits significant jumps, altogether up to 2–3 times going from 1 to 16 processes.
- Another reason for the significant degradation in the parallel performance of GAMG and GAMG/CG solvers is the algorithmic scalability of the parallel GAMG algo-

Table 1
Performance results of first benchmark problem (1) on quadrilateral mesh.

	p	1×1	2×1	4×1	8×1	8×2
			Mesh size – 1018488			
DIC/C	T_p	265.1	161.7	77.0	41.1	28.5
	N_p^{av}	721.9	839.3	865.4	942.4	1009.7
	T_p^{av}	0.0367	0.0192	0.0089	0.0044	0.0028
GAMG	T_p	47.8	27.2	16.1	14.2	19.5
	N_p^{av}	20.2	21.6	24.2	36.9	59.0
	T_p^{av}	0.2364	0.1258	0.0665	0.0386	0.0331
GAMG/C	T_p	47.5	28.5	14.6	11.0	11.5
	N_p^{av}	8.9	10.3	9.9	13.0	16.4
	T_p^{av}	0.5333	0.2765	0.1471	0.0848	0.0701
			Mesh size – 2048000			
DIC/CG	T_p	799.3	437.1	237.0	133.4	87.2
	N_p^{av}	1015.6	1140.1	1142.6	1448.2	1401.6
	T_p^{av}	0.0787	0.0383	0.0207	0.0092	0.0062
GAMG	T_p	100.7	64.6	36.6	27.2	20.9
	N_p^{av}	21.3	26.2	28.0	39.1	38.8
	T_p^{av}	0.4726	0.2465	0.1307	0.0696	0.0538
GAMG/CG	T_p	94.7	71.7	39.4	24.6	17.8
	N_p^{av}	8.9	13.2	13.9	16.2	15.5
	T_p^{av}	1.0643	0.5435	0.2836	0.1518	0.1145
			Mesh size – 4102264			
DIC/CG	T_p	2119.4	1114.9	872.2	414.6	265.8
	N_p^{av}	1427.0	1452.0	1928.4	1939.4	1717.8
	T_p^{av}	0.1485	0.0768	0.0452	0.0214	0.0155
GAMG	T_p	243.9	134.9	105.6	57.3	47.9
	N_p^{av}	24.6	26.3	38.4	41.2	46.8
	T_p^{av}	0.9913	0.5130	0.2751	0.1390	0.1023
GAMG/CG	T_p	267.0	130.2	133.1	56.0	41.5
	N_p^{av}	12.4	11.5	22.4	18.4	19.1
	T_p^{av}	2.1530	1.1319	0.5942	0.3043	0.2173
			Mesh size – 8192000			
DIC/CG	T_p	–	3933.5	2165.9	1126.8	829.6
	N_p^{av}	–	2468.9	2463.9	2535.1	2635.1
	T_p^{av}	–	0.1593	0.0879	0.0445	0.0315
GAMG	T_p	–	390.9	185.2	106.2	77.4
	N_p^{av}	–	38.6	33.7	38.0	40.1
	T_p^{av}	–	1.0127	0.5495	0.2795	0.2436
GAMG/CG	T_p	–	431.0	213.6	127.4	80.9
	N_p^{av}	–	19.8	17.9	21.0	19.3
	T_p^{av}	–	2.1766	1.1935	0.6064	0.4191



(a) on quadrilateral mesh with 1018488 elements



(b) on quadrilateral mesh with 4102264 elements

Fig. 4. Parallel speedup $S_p = T_1/T_p$ and algorithmic speedup $S_p^{alg} = T_1^{av}/T_p^{av}$ of the linear solvers on quadrilateral mesh.

rithm. In Fig. 4(a) and (b), we show the parallel speedup $S_p = T_1/T_p$ and algorithmic speedup $S_p^{alg} = T_1^{av}/T_p^{av}$ of the linear solvers for two meshes with sizes of 1018488 and 4102264, respectively. We see that as the size of the sub-domain reduces the parallel algorithm of DIC/CG solver scales much better than the parallel algorithms of GAMG and GAMG/CG solvers.

Next, we have studied the ability of our OpenFOAM-based parallel solver to utilize the full power of heterogeneous cluster made of computational nodes of different speeds. For these numerical tests we have used eight additional nodes with Intel Quad Q6600 processors, 4 cores (2.4 GHz) per node.

Table 2
Performance results of first benchmark problem (1) on quadrilateral mesh using heterogeneous cluster.

	Mesh size – 8192000						
	p	16×1	16×1	16×2	16×2	16×4	16×4
	$i7(w)$	1.6	1.7	1.6	1.7	1.6	1.7
DIC/CG	T_p	647.2	728.6	574.6	536.6	540.7	524.7
	N_p^{av}	2340.7	2589.5	2790.7	2659.9	2630.2	2643.9
	T_p^{av}	0.0277	0.0281	0.0206	0.0202	0.0206	0.0199
GAMG	T_p	77.6	93.6	81.2	83.8	121.0	139.5
	N_p^{av}	42.0	50.9	53.7	55.7	55.3	60.4
	T_p^{av}	0.1848	0.1840	0.1512	0.1505	0.2189	0.2310
GAMG/CG	T_p	81.9	90.9	77.9	78.7	100.4	115.0
	N_p^{av}	20.2	22.3	24.2	24.5	22.0	24.2
	T_p^{av}	0.4055	0.4077	0.3221	0.3213	0.4561	0.4752

Computing nodes with Intel Core i7-860 processors are up to 1.6–1.7 times faster than the Intel Quad Q6600 processors for solving our benchmark problem. To achieve the load balancing between $i7$ and q nodes, we employ the weights in the mesh partitioning algorithm from the Scotch library (Chevalier and Pellegrini, 2008). The performance results of the tests on heterogeneous cluster are presented in Table 2. Here $i7(w)$ is the weighting factor used in mesh partitioning algorithm for faster $i7$ nodes.

- Solving our biggest case in this work with 8192 000 finite volumes, we obtain a real speedup only with DIC/CG linear solver. The performance of GAMG linear solver is further degrading. It is interesting to note that slight changes in the weighting factor w, can cause significant changes in the convergence of GAMG: from 43.2 to 51.7 iterations per time step with 16×1 processes.
- If we investigate the algorithmic speed-up, i.e. the averaged time for one iteration, the load balancing strategy gives speed-up for both GAMG and GAMG/CG solvers up to 16×2 processes.

In Table 3, we present the same performance data obtained for the first benchmark problem (1) on acute triangle meshes with similar number of elements: 1 million and 4 millions finite volumes. We have applied matrix renumbering algorithm (Cuthill and McKee, 1969) to reduce the matrix bandwidth. The following main conclusions can be done.

- The DIC/CG solver on acute triangle meshes has shown slightly bigger number of iterations than on quadrilateral meshes. However, due to the slightly smaller averaged time for one iteration T_p^{av}, the total running times T_p are quite similar. Smaller times T_p^{av} are obtained due to the better caching.
- However, the major difference is in the performance of GAMG and GAMG/CG solvers. Both solvers do not show degradation in the performance of parallel preconditioners with increasing number of processes. For both solvers the number of iterations N_p^{av} stays almost constant on acute triangle meshes. This makes a big difference in real speedup and efficiency of those linear solvers.

Table 3
Performance results of first benchmark problem (1) on acute triangle mesh.

	p	1×1	2×1	4×1	8×1	8×2
			Mesh size – 1023775			
DIC/CG	T_p	291.1	160.5	81.9	39.5	23.6
	N_p^{av}	933.2	980.2	1112.1	1109.0	1035.7
	T_p^{av}	0.0312	0.0164	0.0074	0.0036	0.0023
	S_p	–	1.81	3.55	7.37	12.32
	S_p^{alg}	–	1.91	4.23	8.75	13.68
GAMG	T_p	31.9	17.0	8.7	5.0	3.6
	N_p^{av}	13.4	13.4	13.0	12.8	12.6
	T_p^{av}	0.2382	0.1267	0.0666	0.0389	0.0283
	S_p	–	1.88	3.69	6.41	8.95
	S_p^{alg}	–	1.88	3.58	6.13	8.41
GAMG/CG	T_p	35.9	20.7	9.6	5.6	3.9
	N_p^{av}	6.9	7.6	6.8	6.9	6.7
	T_p^{av}	0.5195	0.2720	0.1406	0.0814	0.0579
	S_p	–	1.73	3.75	6.38	9.24
	S_p^{alg}	–	1.91	3.70	6.38	8.97
			Mesh size – 4102264			
DIC/CG	T_p	2094.1	1483.1	801.3	371.8	268.3
	N_p^{av}	1655.5	2129.4	2120.0	1974.4	1990.6
	T_p^{av}	0.1265	0.0697	0.0378	0.0188	0.0135
	S_p	–	1.41	2.61	5.63	7.80
	S_p^{alg}	–	1.82	3.35	6.72	9.38
GAMG	T_p	161.3	82.2	42.6	21.8	15.12
	N_p^{av}	16.4	16.0	15.6	15.0	15.0
	T_p^{av}	0.9834	0.5140	0.2732	0.1456	0.1008
	S_p	–	1.96	3.78	7.39	10.67
	S_p^{alg}	–	1.91	3.60	6.76	9.76
GAMG/CG	T_p	175.0	91.5	45.1	23.7	15.8
	N_p^{av}	8.2	8.3	7.7	7.6	7.4
	T_p^{av}	2.1343	1.1029	0.5863	0.3114	0.2128
	S_p	–	1.91	3.88	7.40	11.12
	S_p^{alg}	–	1.94	3.64	6.86	10.03

Finally, we test our OpenFOAM-based parallel multi-physics solver on second benchmark problem. Due to observed clear advantages of acute triangle meshes in parallel performance of GAMG and GAMG/CG solvers, we have chosen this type of finite volume mesh for our multi-physics benchmark problem. Another advantage of acute triangle meshes is that we can generate such meshes for complicated geometries with our own meshing tool, which is making calls to Acute meshing library (Erten and Ungor, 2009).

We have generated acute triangle mesh with 1021873 cells. We remind that air region makes up 0.27% of whole simulated domain area. However, due to employed in our meshing tool adaptation heuristics, see Fig. 3, the number of cells in the air region is 116062, i.e. 11%.

Table 4
Performance results of multi-physics solver for second benchmark problem on acute triangle mesh.

	p	1×1	2×1	4×1	8×1	8×2
			Mesh size – 1021873			
DILU/BiCG	T_p	1618.0	946.2	378.6	204.8	135.3
	$T_p(Te)$	1610.7	938.6	373.7	200.1	128.6
	N_p^{av}	2604.1	2848.7	2320.8	2634.9	2584.1
	T_p^{av}	0.0619	0.0330	0.0161	0.0076	0.0050
	S_p	–	1.71	4.27	7.90	11.96
	S_p^{alg}	–	1.88	3.84	8.15	12.23
GAMG	T_p	222.8	125.9	18.2	14.5	13.7
	$T_p(Te)$	215.6	118.4	13.2	9.9	7.1
	N_p^{av}	108.1	109.5	20.6	19.7	19.1
	T_p^{av}	0.1995	0.1081	0.0641	0.0503	0.0372
	S_p	–	1.77	12.24	15.37	16.26
	S_p^{alg}	–	1.85	3.11	3.97	5.37

Note that due to the fact that global temperature equation in "air" cells has convective term (see equation (5)) and approximation of it is not symmetric (upwind), we obtain monotone but non-symmetric matrices in our linear systems. To solve this systems we use preconditioned bi-conjugate gradient solver (BiCG) with diagonal incomplete LU (DILU) preconditioner – DILU/BiCG and multigrid GAMG solver with DILU smoother. Combination of Bi-conjugate gradient solver (BiCG) with GAMG preconditioner is currently not implemented in OpenFOAM.

The obtained parallel performance results of our multi-physics solver are shown in Table 4. Here we show the total wall time T_p of 10 time steps with p processes, the time $T_p(Te)$ of 10 time steps of global temperature equation solver with p processes, the average number of iterations N_p^{av} in global temperature equation solver with p processes, the average time of one iteration of global temperature equation solver – $T_p^{av} = T_p(Te)/N_p^{av}/10$, parallel speedup of multi-physics solver - $S_p = T_1/T_p$ and algorithmic speedup of global temperature equation solver – $S_p^{alg} = T_1^{av}/T_p^{av}$ with p processes for DILU/BiCG and GAMG linear solvers.

Performance results in Table 4 show that developed OpenFOAM-based multi-physics solver is very efficient, especially, with parallel multigrid GAMG solver. Obviously, the parallel scalability of the developed parallel solver remains to be tested on a bigger number of parallel processors.

5. Conclusions

We have tested the parallel performance of the conjugate gradient solver with diagonal incomplete Cholesky preconditioner (DIC/CG), generalized geometric-algebraic multigrid (GAMG) solver and the conjugate gradient solver with GAMG preconditioner (GAMG/CG) in the OpenFOAM-based application for heat transfer in and around elec-

trical power cables. The best running times in our tests were obtained with GAMG/CG and GAMG solvers. However, DIC/CG linear solver has shown to be less sensitive to the parallel preconditioning degradation. It has also shown a better algorithmic scalability.

The weighting factors in mesh partitioning algorithm allow efficient utilization of heterogeneous computing nodes for our parallel application. More research is needed on the influence of different graph partitioning algorithms.

Acknowledgments. The work of authors was supported by Eureka project E!6799 POWEROPT "Mathematical modelling and optimization of electrical power cables for an improvement of their design rules" and by EU under the COST programme Action IC1305, "Network for Sustainable Ultrascale Computing (NESUS)".

References

AlOnazi, A. (2013). *Design and optimization of OpenFOAM-based CFD applications for modern hybrid and heterogeneous HPC platforms.* Master's thesis, University College Dublin, Dublin, Ireland.

Bergman, T.L., Incropera, F.P. (2011). *Fundamentals of Heat and Mass Transfer*. Wiley, New York.

Chevalier, C., Pellegrini, F. (2008). PT-Scotch: a tool for efficient parallel graph ordering. *Parallel Computing*, 34(6–8), 318–331.

Cufflink (2012). *Library for linking numerical methods based on Nvidia's CUDA C programming language and OpenFOAM*. https://code.google.com/p/cufflink-library/.

Culpo, M. (2012). *Current bottlenecks in the scalability of OpenFOAM on massively parallel clusters*. Prace white papers, CINECA, Bologna.

CUSP (2015). *Library for sparse linear algebra and graph computations based on Thrust library*. http:// cusplibrary.github.io/.

Cuthill, E., McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In: *Proceedings 24th National Conference of the ACM*, 157–172.

Čiegis, R., Ilgevičius, A., Liess, H., Meilūnas, M., Suboč, O. (2007). Numerical simulation of the heat conduction in electrical cables. *Mathematical Modelling and Analysis*, 12(4), 425–439.

Čiegis, R., Čiegis, R., Meilūnas, M., Jankevičiūtė, G., Starikovičius, V. (2008). Parallel mumerical algorithm for optimization of electrical cables. *Mathematical Modelling and Analysis*, 13(4), 471–482.

Čiegis, R., Starikovičius, V., Bugajev, A. (2014). On parallelization of the OpenFOAM-based solver for the heat transfer in electrical power cables. In: *Euro-Par 2014: Parallel Processing Workshops*, *Lecture Notes in Computer Science*, Vol. 8805, pp. 1–11.

Dagna, P. (2012). *OpenFOAM on BG/Q porting and performance*. Prace report, CINECA, Bologna.

Dagna, P., Hertzer, J. (2013). *Evaluation of multi-threaded OpenFOAM hybridization for massively parallel architectures*. Prace white papers, wp98, CINECA, Bologna.

Duran, A., Celebi, M.S., Piskin, S., Tuncel, M. (2015). Scalability of OpenFOAM for bio-medical flow simulations. *The Journal of Supercomputing*, 71(3), 938–951.

Erten, H., Ungor, A. (2009). Quality triangulations with locally optimal Steiner points. *SIAM Journal of Scientfic Computing*, 31(3), 2103–2130.

Filatovas, E., Kurasova, O., Sindhya, K. (2015). Synchronous R-NSGA-II: an extended preference-based evolutionary algorithm for multi-objective optimization. *Informatica*, 26(1), 33–50.

Higuera, P., Lara, J., Losada, I. (2013). *Realistic wave generation and active wave absorption for Navier–Stokes models: application to OpenFOAM*. *Coastal Engineering*, 71(1), 102–118.

Karypis, G., Kumar, V. (1999). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1), 359–392.

Kuettler, U., Wall, W.A. (2008). Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43, 61–72.

Lančinskas, A., Ortigosa, P.M., Žilinskas, J. (2015). Parallel optimization algorithm for competitive facility location. *Mathematical Modelling and Analysis*, 20(5), 619–640.

Liu, Y. (2011). *Hybrid parallel computation of OpenFOAM solver on multi-core cluster systems*. Master's thesis, KTH Royal Institute of Technology, Stockholm.

Makhkamova, I. (2011). *Numerical investigations of the thermal state of overhead lines and underground cables in distribution networks*. PhD thesis, Durham University, Durham.

Muddle, R., Milhajlovic, M., Heil, M. (2012). An efficient preconditioner for monolithically-coupled large-displacement fluid-structure interaction problems with pseudo-solid mesh updates. *Journal of Computational Physics*, 231, 7315–7334.

Neher, J., McGrath, M. (1957). The calculation of the temperature rise and load capability of cable systems. *AIEE Transactions*, 76, 752–772.

OpenFOAM (2015). *Open source field operation and manipulation, CFD toolbox*. http://www.openfoam.org.

Petit, O., Bosioc, A., Nilsson, H., Muniean, S., Susan-Resigo, R. (2011). Unsteady simulations of the flow in a swirl generator using OpenFOAM. *International Journal of Fluid Machinery and Systems*, 4(1), 199–208.

PETSc (2015). *Portable, Extensible Toolkit for Scientific Computation PETSc: a suite of data structures and routines for the scalable parallel solution of scientific applications modeled by partial differential equations*. http://www.mcs.anl.gov/petsc/petsc-as/.

Rivera, O., Fürlinger, K., Kranzlmüller, D. (2011). Investigating the scalability of OpenFOAM for the solution of transport equations and large eddy simulations. In: *Algorithms and Architectures for Parallel Processing*, *Lecture Notes in Computer Science*, Vol. 7017, pp. 121–130.

Starikovičius, V., Čiegis, R., Iliev, O. (2011). A parallel solver for design of oil filters. *Mathematical Modelling and Analysis*, 16(2), 326–342.

Thrust (2015). *C++ template library for parallel platforms based on the Standard Template Library (STL)*. Http://Thrust.Github.Io/.

Trilinos (2015). *Project to develop scalable (parallel) solver algorithms and libraries within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific applications*. http://trilinos.org/.

Weller, H.G., Tabor, G., Jasak, H., Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Journal of Computational Physics*, 12(6), 620–631.

V. Starikovičius has graduated from Vilnius University Faculty of Mathematics in 1998. He received the PhD in mathematics from the Vilnius University in 2002. He is currently the head of Laboratory of Parallel Computing and associate professor at Department of Mathematical Modelling of Vilnius Gediminas Technical University. His main research interests include parallel and distributed computing, numerical methods for solving partial differential equations, mathematical modelling, porous media flows, hemodynamics.

R. Čiegis has graduated from Vilnius University Faculty of Mathematics in 1982. He received the PhD degree from the Institute of Mathematics of Byelorussian Academy of Science in 1985 and the degree of Habil. Doctor of Mathematics from the Institute of Mathematics and Informatics, Vilnius in 1993. He is a professor and the head of Mathematical Modelling department of Vilnius Gediminas Technical University. His research interests include numerical methods for solving nonlinear PDE, parallel numerical methods, mathematical modelling in nonlinear optics, porous media flows, technology, image processing, biotechnology.

A. Bugajev has graduated from Vilnius Gediminas Technical University, where achieved bachelor (2009) and master (2011) degree in mathematics. He is currently PhD student at the Department of Mathematical Modelling, Vilnius Gediminas technical university. He works at the Department of Mathematical Modelling of Vilnius Gediminas Technical University since 2011. His main research interests include theory of algorithms, numerical methods, parallel computing, computing using GPU, mathematical modelling.

Elektros kabelių modeliavimui skirtų lygiagrečiųjų sprendiklių, sukurtų OpenFoam pagrindu, efektyvumo analizė

Vadimas STARIKOVIČIUS, Raimondas ČIEGIS, Andrej BUGAJEV

Šiame darbe sprendžiamas šilumos pasiskirstymo elektros kabeliuose uždavinys. Lygiagretusis sprendiklis yra sukurtas panaudojant OpenFOAM įrankį. Ištirtas šio sprendiklio algoritmų efektyvumas. Pirmajame uždavinyje modeliuojami trys elektros kabeliai, kurie tiesiogiai pakloti žemėje. Palyginti trys tiesinių lygčių sistemų sprendikliai – jungtinių gradientų metodas su dviem matricos sąlygotumo modifikatoriais, įstrižaininis nepilno Chalesky skaidymo algoritmas ir algebrinis daugiatinklis sprendiklis. Taip pat naudotas daugiatinklis metodas kaip savarankiškas sprendiklis. Lygiagrečiųjų sprendiklių konvergavimo ir išplečiamumo savybės ištirtos aproksimuojant diferencialinį uždavinį keturkampiuose ir Acute įrankiu generuotuose trikampiuose tinkluose. Antrasis uždavinys yra sudėtingesnis, elektros kabeliai yra patalpinti į plastikines movas, kurios paklotos žemėje. Nagrinėjamas šilumos sklidimas elektros kabelyje, ore ir žemėje. Šiam multi-fizikos uždaviniui pateikiamas nestandartinis lygiagretusis sprendiklis. Ištirtas pasirinktų modifikatorių efektyvumas ir išplečiamumas. Skaitiniai eksperimentai atlikti klasteryje, sudarytame iš daugiabranduolinių skaičiavimo mazgų.