INFORMATION TECHNOLOGY
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Review article

UDC 004.382.2, 004.7

https://doi.org/10.23947/2587-8999-2024-8-1-12-28

Development of Supercomputer Technologies at the Institute of Mathematical Modelling and Keldysh Institute of Applied Mathematics of Russian Academy of Sciences

Mikhail V. Yakobovskiy ✉, **Marina A. Kornilina**

Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, Moscow, Russian Federation

✉lira@imamod.ru

Abstract

A review scientific work in the field of supercomputer technologies at the Institute of Mathematical Modelling and Keldysh Institute of Applied Mathematics is presented. Progress in supercomputer technologies, programming tools and technique (such as hyperbolization, load balancing, fault tolerance, adaptive mesh refinement, rational mesh decomposition) and several supercomputer applications are presented.

Keywords: supercomputers, supercomputing resource center, hyperbolic equations, hyperbolization, load balancing, fault tolerance, dynamically adaptive computational grids, rational mesh decomposition

Acknowledgements. We would like to express our sincere gratitude to S.V. Polyakov the head of Supercomputing Resource Center at the Keldysh Institute for the valuable discussions and materials provided, as well as to the team of the Center for the opportunity to use the facilities of the Center for carrying out numerous computations mentioned in this publication.

For citation. Yakobovskiy M.V., Kornilina M.A. Development of supercomputer technologies at the Institute of Mathematical Modelling and Keldysh Institute of Applied Mathematics of Russian Academy of Sciences. *Computational Mathematics and Information Technologies*. 2024;8(1):12–28. https://doi.org/10.23947/2587-8999-2024-8-1-12-28

Обзорная статья

Развитие суперкомпьютерных технологий в ИММ РАН и ИПМ им. М.В. Келдыша РАН

М.В. Якобовский ✉, **М.А. Корнилина**

Институт прикладной математики им. М.В. Келдыша Российской академии наук, г. Москва, Российская Федерация

✉ lira@imamod.ru

Аннотация

Представлен обзор работ в области суперкомпьютерных технологий, проводившихся в ИММ РАН и ИПМ им. М.В. Келдыша РАН. Описаны этапы развития вычислительной техники и алгоритмы, разработанные для суперкомпьютерных систем, такие как гиперболизация уравнений, балансировка загрузки, отказоустойчивость, построение динамически адаптивных расчетных сеток, рациональная декомпозиция сеток, а также некоторые прикладные и научные задачи, успешно решаемые с использованием суперкомпьютеров.

Ключевые слова: суперкомпьютеры, центр коллективного пользования, гиперболические уравнения, гиперболизация, балансировка загрузки, отказоустойчивость, динамически адаптивные расчетные сетки, рациональная декомпозиция сеток

Благодарности. Выражаем искреннюю благодарность за ценные обсуждения и предоставленные материалы руководителю ЦКП ИПМ им. М.В. Келдыша РАН С.В. Полякову, а также сотрудникам ЦКП, с использованием оборудования которого выполнены многочисленные расчеты, упомянутые в данной публикации.

Для цитирования. Якобовский М.В., Корнилина М.А. Развитие суперкомпьютерных технологий в ИММ РАН и ИПМ им. М.В. Келдыша РАН. *Computational Mathematics and Information Technologies*. 2024;8(1):12–28. https://doi.org/10.23947/2587-8999-2024-8-1-12-28

The article is prepared based on the presentation of the corresponding member of the Russian Academy of Sciences, M.V. Yakobovsky, at the scientific conference held at the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences, dedicated to the 80th anniversary of the scientific director of Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences, Academician B.N. Chetverushkin.

The Current State of Computing Technology in Russia and Worldwide. At present, scientific progress is inconceivable without the use of supercomputing technologies. Massive parallel computers are applied in all fields of activity, and the power of the most advanced ones has surpassed the coveted milestone of 1 Exaflop. The performance of the top 500 most powerful open supercomputing systems in the world, depicted in Fig. 1 on a logarithmic scale, resembles an inverse function. Only a few Russian systems made it to the Top-500 list [1], and as we can see, in terms of performance, they lag behind the most powerful computers by several orders of magnitude. However, the right branch of this graph is of particular interest. At the level of about 2 Petaflops (2×1015 operations per second), there is an almost horizontal line consisting of hundreds of systems with very similar performance. These are practically identical "workhorses" used in business and manufacturing (telecommunication systems, cloud providers, etc.) to solve routine technical and engineering tasks. Russian systems are absent there. Similarly, they are absent on the left vertical branch (in the "Research Academic" zone), where 20–30 of the most powerful computers in the world are located, which pose fundamentally new challenges for developers and provide researchers with fundamentally new capabilities. The current ranking list includes only 7 Russian supercomputers, among them three ("Lyapunov", "Chervonenkis", and "Galushkin") belong to "Yandex", two (Christofari and Christofari Neo) to Sberbank, and one each to Moscow State University ("Lomonosov") and MTS (GROM). At the time of its appearance in 2021, the "Chervonenkis" supercomputer of "Yandex" ranked 19th in the rating.

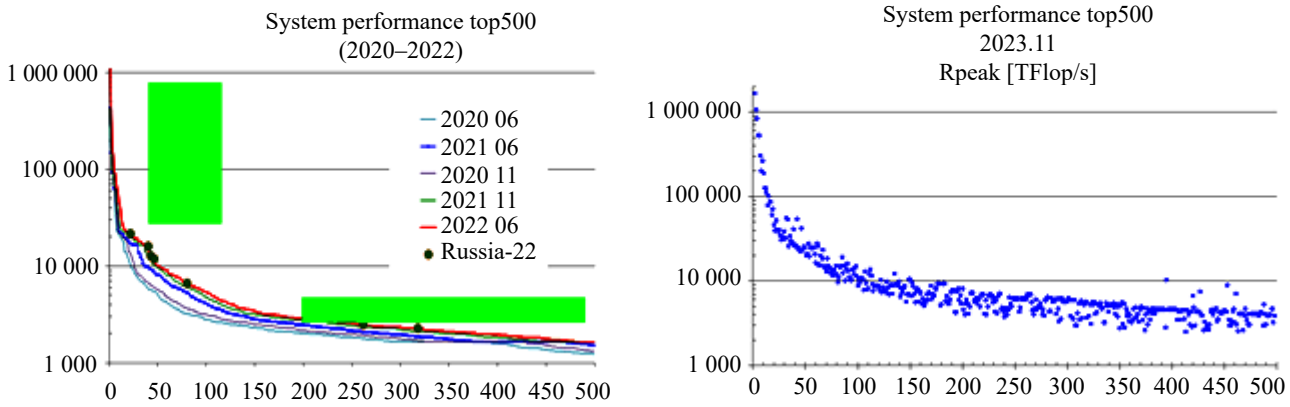


Fig. 1. Performance of supercomputing systems from the top-500 list:
a — in 2020–2022; *b* — in 2023

The computing machines at the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences are not included in the Top-500, yet they persistently and faithfully serve their purposes: they are utilized for algorithm development and practical problem-solving. The modern era of computing technology development at the IPM, marked by the transition to mass utilization of supercomputing technologies, began after the era of the BESM-6 and ES series of computers almost half a century ago. The stages of this development, undergone at the Institute of Mathematical Modelling of the Russian Academy of Sciences (IMM RAS) and the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences, are reflected in Table 1. It all began in 1988 with the pivotal decision to use transputer systems for scientific tasks [2]. Each transputer T800 [3, 4], in terms of its computing power (about 106 floating-point operations per second), was equivalent to the BESM-6 machine and was three times more powerful than the available PC-386 personal computers at that time. Importantly, it also included 4 duplex data transmission channels at 1 Mbyte/s in each direction. There were no local networks of this level in the institute at that time. It was easy to assemble a desktop supercomputer from several transputers.

They could be assembled in considerable quantities. The transputers passed military acceptance tests, were intended for space applications, and were highly reliable. There were no burned-out transputers in our practice. If they were connected now to some suitable system, they would work perfectly well. Based on the same technology, a 32-processor station APS-48 with a performance of 64 Megaflops was used.

Table 1

Historical Milestones of Computer Technology Development at the Keldysh Institute of Applied Mathematics

Dates of commencement	Computers	Research Directions, Some Solved Problems
Until the 1990s. The first Transputer boards at the Institute of Mathematical Modelling of the Russian Academy of Sciences		
October 1988	T-800	Computations of two-dimensional problems of viscous gas dynamics based on kinetically consistent schemes
The 1990s. IMM RAS Cluster based on Parsytec		
August 1991	32-processor APS-48 workstation, 64 MFLOP	Modelling of compressible viscous gas flows; Modelling of organic fuel combustion processes; Optimization of oil production processes; Modelling of processes in electron-hole plasma of semiconductors; Solution of problems of laser thermonuclear fusion (Richtmyer-Meshkov instability); Parallel algorithms for solving linear algebra problems; Processor load balancing algorithms; Visualization of results of multiprocessor calculations; Task flow management in distributed systems
1994–95	12-processor Parsytec PowerXplorer system, 1 GFlops	
1996	12-processor Parsytec CC system, 3.1 GFLOP	
May 1998	32-processor Parsytec CC system, 6.5 GFLOP	
October 1998 October 1998	Dual-processor Sun Enterprise 250 server (UltraSPARC 300 MHz) Clustering (12 + 32 Parsytec CC nodes)	
The 2000s. Computing Center of the Institute of Mathematical Modelling of the Russian Academy of Sciences		
April 24, 2001	24-processor cluster (12 dual-processor nodes) with a performance of 14.4 Gflops and Fast Ethernet (100 Mbps)	Modelling three-dimensional unsteady supersonic flows of viscous gas around bodies of complex shapes; Solving problems of aeroacoustics concerning the generation and propagation of noise in viscous gas flows; Modelling methane combustion processes in the atmosphere; Solving nonlinear electron transport problems in quantum switches; Development of a distributed visualization system for three-dimensional scalar and vector fields computed on high-performance computing systems; Training students and graduate students in the fundamentals of parallel programming
2010 and beyond. Shared-Use Center at the Keldysh Institute of Applied Mathematics of the Russian Academy of Sciences		
2010 2012 2017 2017 The Shared-Use Center has been included since 2019	K100 (created jointly with the Research Institute “Quantum”) K10 K60, section without GPU and section with GPU	Modelling applied problems in aerodynamics and aeroacoustics; Calculations of plasma physics problems; Solution of applied problems in continuum mechanics; Prediction of pollution spread in water bodies using the example of the Azov Sea; Investigation of problems in space flight mechanics and motion control; Modelling two-phase mass transfer in porous media; Development of computer models and methods for high-temperature hydrodynamics; Development of decision support information systems

Resource	Computational Nodes	CPU Cores per Node / Total	GPU Cores per Node / Total	Peak Performance
K100	64	2×6/768	3×448/86016	107.9 TFLOPS
K10	16	2×8/256	3×512/24576	31.9 TFLOPS
K60section without GPU	86	2×14/2408	–	81.9 TFLOPS
K60, section with GPU	10	2×16/320	4×5120/204800	300.0 TFLOPS

Subsequently, hybrid manufacturing systems from Germany, such as the Parsytec PowerXplorer (1 GFLOPS), began to be utilized. These systems integrated both transputers and more powerful processors simultaneously. Additionally, Parsytec CC systems were employed, no longer based on transputers and their links, but rather on IBM processors and high-speed interconnect channels (HS links).

The first Parsytec CC-12 system was acquired within the framework of a European Union (EU) project under the overall direction of Nobel laureate I.R. Prigogine for solving problems based on quasi-hydrodynamic systems of equations. In 1998, an EU-approved delivery of a 96-processor Parsytec CC system with a performance of 19.5 GFLOPS was planned. However, the United States imposed an embargo on this delivery, concerned that the Institute would receive too much computing power. Consequently, the infrastructure was installed for all modules, but only one-third of them were filled. The Institute received only a 32-processor Parsytec CC-32 system with a performance of 6.5 GFLOPS. Architecturally, it was fully equivalent to the Parsytec CC-12 system, which allowed, after their connection via an external network, running programs on the resulting extended system.

These were the first massive multiprocessor systems that practically enabled the development of scalable parallel algorithms and programs.

An additional advantage of Parsytec CC systems was that these supercomputers operated under the control of the AIX operating system, configured for real-time task solving. In Germany, they were used in production (in metallurgical, paper industries), on transport routes to monitor road conditions, etc. The virtually complete absence of runtime overheads not directly related to the executed program significantly facilitated debugging and tuning of algorithms. Modern systems based on Linux and Windows lack this particular positive feature.

Fig. 2 shows a historical photograph taken in front of the building of the Institute of Mathematical Modelling of the Russian Academy of Sciences (Miusskaya Square, 4A) in 1998, on the day of receiving, unloading, and beginning the installation of modules of the Parsytec CC-32 computing system. The porch of Building A was partially dismantled to facilitate the entry of equipment. Unloading from the truck platforms was carried out using lifting cranes, but transportation into and within the building was carefully done manually by the efforts of scientific and engineering staff, most of whom are captured in the photograph.

Fig. 2. Institute of Mathematical Modelling of the Russian Academy of Sciences staff after unloading Parsytec CC-32 modules. May 1998

Since only two-thirds of the computing modules were delivered, leaving plenty of unused “infrastructure” such as racks, empty rack units equipped with power supplies, and other components. Separate purchases were made for motherboards (dual-processor, using Intel processors), processors, disk subsystems, and network cards. From these, a computing cluster was manually assembled at the Institute of Mathematical Modelling, schematically depicted in Fig. 3. It operated for a considerable period, enabling research and solving various scientific problems across a wide spectrum. The described computing complex was decommissioned upon the arrival of more powerful computing systems, remaining fully operational until then.

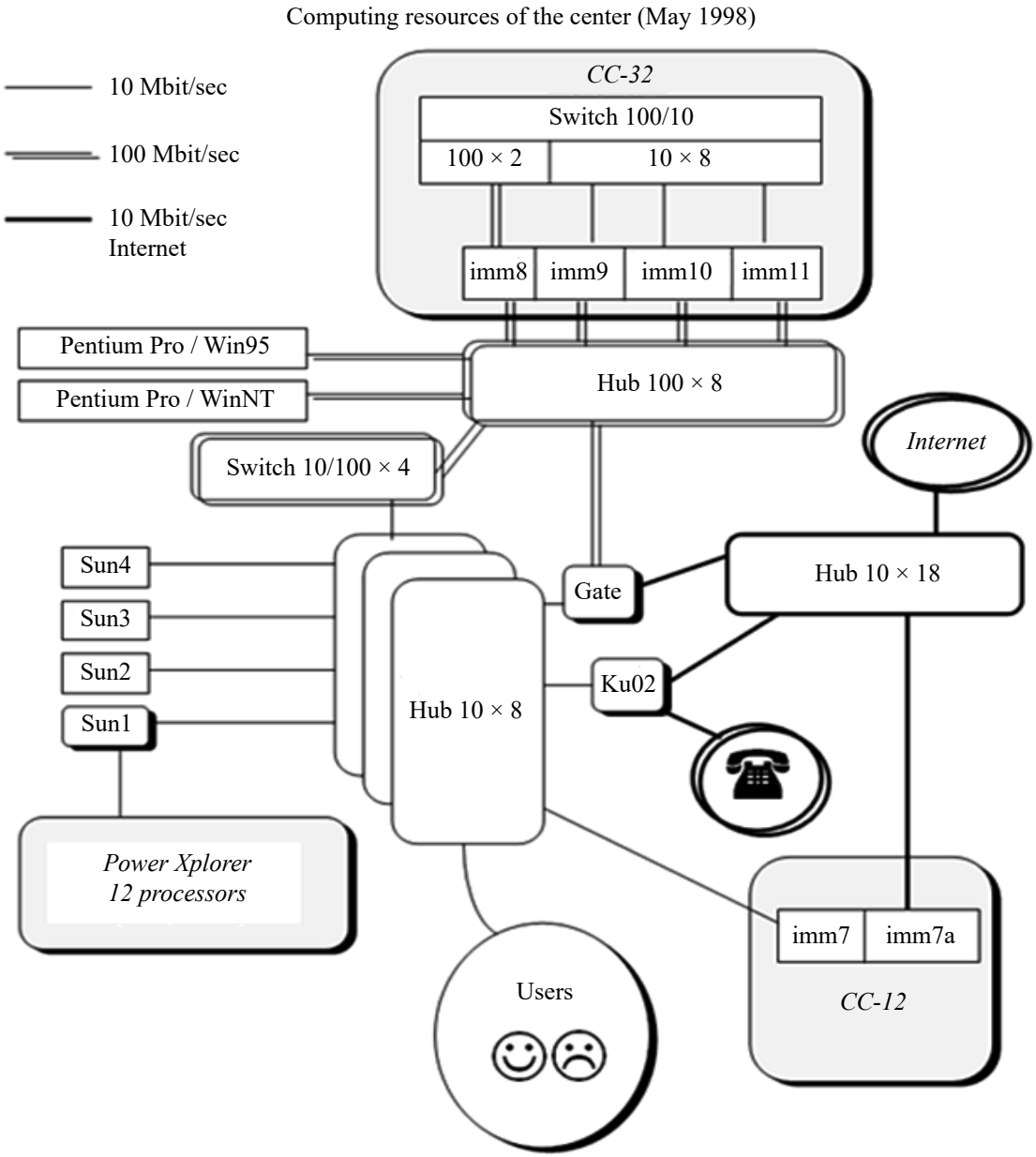


Fig. 3. Computing Cluster of the Institute of Mathematical Modelling, Russian Academy of Sciences (IMM RAS) Parsytec PowerXplorer — Parsytec CC-12 — Parsytec CC-32, 1998

At present, the institute, established to address important national economic tasks using advanced computing technology (primarily the creation of our country's missile-nuclear shield), has a number of high-performance systems at its disposal. While they cannot compete with the Top-500 list, they are fully utilized for problem-solving, algorithm development, and programming. These include the K-100 system with a peak performance of 107 teraflops, the K60 system, currently performing at around 400 teraflops, and others. The K-100 system [5, 6] was the first major computing system in Russia to use graphics cards as accelerators. Each computing node of the K-100 system consists of two six-core Intel Xeon X5670 processors and three nVidia Fermi C2050 accelerators, each containing 448 CUDA cores. The computing nodes are interconnected by high-speed data transfer networks MVC-Express and Infiniband. The K-100 system was installed in 2010 by direct order of Vladimir Vladimirovich Putin, then Prime Minister of Russia, who allocated targeted funds for this purpose. Within a year, the system was developed, created, put into operation, and has been operational ever since.

The Computing Center of the Institute is distributed, located in two territories at opposite ends of Moscow. This is the Shared-Use Center (SUC) [7], which provides remote access capabilities. On one hand, its resources are used by institute staff, while on the other hand, external users access them under agreements on either paid or unpaid basis, through commercial contracts and grants from the Russian Science Foundation. The Russian Science Foundation supports the use of shared-use centers and the implementation of projects, which includes financing (up to 20 % of the grant amount) for the use of SUC resources.

Supercomputer algorithms and solving complex current problems. The creation of the supercomputer Center for Collective Use at the Institute of Mathematical Modelling named after M.V. Keldysh of the Russian Academy of Sciences was greatly influenced by B.N. Chetverushkin. With his direct involvement, the computing cluster technology of the IMM RAS (Parsytec CC-12, CC-32 PowerXplorer) was obtained. However, equally important is his role in developing algorithms for supercomputer technology and supporting research in this direction. In the absence of adequate algorithms, the equipment remains inert, and developing algorithms for supercomputing, multiprocessor computing technology is by no means simple. It is necessary to efficiently solve a wide range of problems. An algorithm must be created that allows for task splitting — decomposition into a large number of "independent" fragments so that the computational processes performing the fragments are weakly connected (interact infrequently). Then, this algorithm needs to be turned into a program that will operate efficiently. However, computing technology is very diverse: multiprocessor, multicore, with various types of accelerators, distributed memory, workstations, and cloud systems. Therefore, there is a requirement for the adequacy and correspondence of supercomputer algorithms to computational architecture.

One of the earliest works dedicated to the development of supercomputer algorithms is the study by [8], which focuses on adapting the iterative (α–β) algorithm for solving multidimensional parabolic and elliptic equations and its parallelization method for systems with distributed memory.

Another significant achievement was the development of kinetically consistent schemes [9, 10] and the quasi-gasdynamic (QGD) system of equations. Unlike the Navier-Stokes equations, the QGD system is hyperbolic [11]. This allows for the use of explicit schemes in constructing numerical models, providing significant advantages when executing on high-performance systems with massive parallelism.

B.N. Chetverushkin's thesis is well-known, emphasizing that supercomputer algorithms should possess two properties: they must be logically simple, allowing for the utilization of complex hierarchical computing systems (including accelerators), and yet effective. This combination is not always achievable, as these two requirements often contradict each other. Explicit finite difference schemes are known to adapt well to supercomputing calculations because they exhibit an important property of locality. When computing grid variables in any cell of the computational grid, only values from neighboring cells are used. This property of locality allows for the decomposition of the problem into a large number of independent processes that only occasionally exchange data between the grid fragments processed by the processes. The well-known domain decomposition method (geometric parallelism), used in solving problems using explicit finite difference schemes, is based on this property of locality. However, explicit schemes are problematic because when applied to solving elliptic or parabolic equations, they impose very strict conditions on the time step. If the spatial step is halved, the time step must be reduced by a factor of four. This is dictated by the Courant criterion and the fundamental properties of the corresponding finite difference schemes and algorithms. This poses a problem because halving the spatial grid doubles the number of computational nodes. Additionally, reducing the time step by four times increases the computational error associated with accumulating errors due to finite machine precision. The idea of hyperbolizing parabolic systems of equations allows transitioning from parabolic schemes to schemes resembling explicit ones, i. e., hyperbolic schemes, with softer stability conditions. Thus, by halving the spatial step, it is permissible to reduce the time step not by 4 times, but by two times, or possibly slightly more than 2 times.

A vivid example of applying this idea can be found in the computation of a cosmological problem — a three-dimensional calculation of the gravitational potential [12, 13]. The problem (1) is complex, involving three-dimensional equations, gravitational potential, and magnetohydrodynamics.

$$\frac{\delta^2\varphi}{\delta x_i^2} = 4\pi G\rho(r), \tag{1}$$

$$\rho(r) = \begin{cases} \rho & r \le R \\ \delta & r > R, \end{cases}$$

$$\frac{\varphi_i^{j+1} - \varphi_i^j}{\Delta t} = \left(\varphi_{\bar{x}}\right)_x^j + F,$$

$$\frac{\varphi_i^{j+1} - \varphi_i^{j-1}}{2\Delta t} + \tau^* \frac{\varphi_i^{j+1} - 2\varphi_i^{j-1}}{\Delta t^2} = \left(\varphi_{\bar{x}}\right)_x^j + F.$$

where φ is the gravitational potential, F is the free term, and G is the gravitational constant.

In the equation, a term is highlighted that transforms the parabolic system into a hyperbolic one, demonstrating the extent to which gains are achieved in the time step size. By employing the hyperbolized system of equations instead of the original parabolic one, when refining the spatial step, we gain more than 40 times in the time step. Computational speed will increase by 40 times, and consequently, the accumulated error will be reduced as well (Fig. 4).

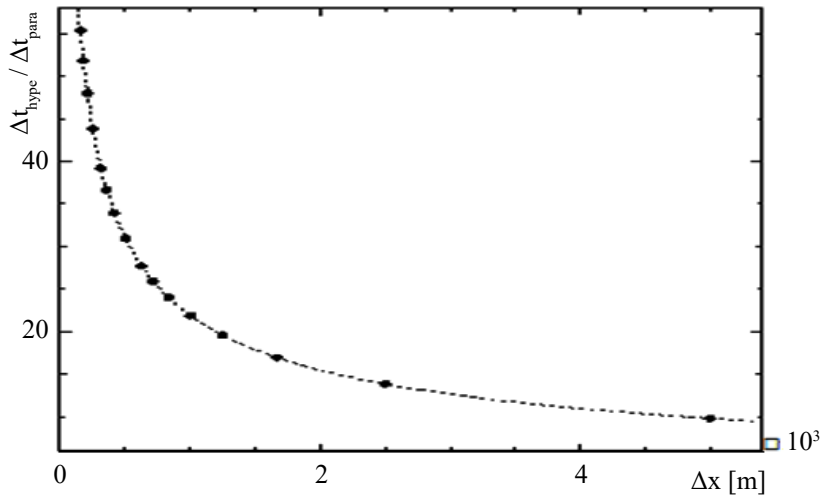


Fig. 4. The ratio of time discretization steps for the parabolic and hyperbolic methods as a function of spatial discretization in solving the Poisson equation

It should be noted that the proposed hyperbolization does not lead to a change in the quality of approximation. The quality of approximation remains within the same framework as for the Navier-Stokes equations. Fig. 5 illustrates the interaction between interstellar gas and a massive object. Two calculation options are presented in Figure 5: (*a*) shows the results of a calculation with low resolution, while (*b*) shows the results of a calculation on a high-resolution grid. High resolution implies the use of approximately 6 billion nodes for the three-dimensional problem. On the low-resolution grid (less than a billion nodes), we do not actually see a meaningful picture. However, on the high-resolution grid, we can observe the process of jet formation — a relativistic jet formed as a result of the interaction of gravitational and magnetic fields. The jets obtained in the calculations do exist. Astronomers observe the relativistic jet from the center of the M87 galaxy in various wavelength ranges using optical and radio telescopes. The image of the jet (J.A. Biretta et al., 2000) obtained from the Hubble Space Telescope [14] is widely known. To solve such problems, it is impossible to use just a fragment of a supercomputer — substantial computational power is required, essentially the entire supercomputer.

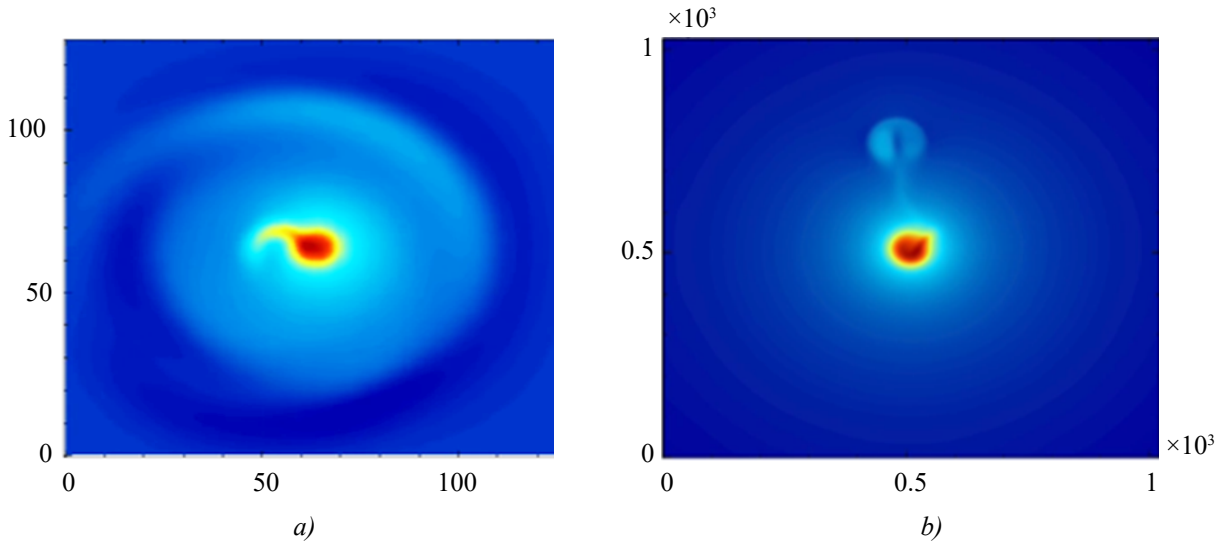


Fig. 5. Gravitational potential:
a — results of calculation with low resolution (blurry picture);
b — results of calculation with high resolution (relativistic jet).

It is worth noting that powerful supercomputers are predominantly used in a shared multitasking mode to solve streams of relatively small tasks by a large number of scientific groups. Each group applies only a small fraction of the supercomputer's capacity for a specific calculation — one-tenth, one-hundredth of its part. However, tasks of this class can only be solved using the full computational power of the entire supercomputer.

With sufficient computational power, it is possible to model these processes and study their digital model. If such computational power is not available, we only see a blurred picture, as shown in Fig. 5 *a*.

What does a system with high computational power look like? Such a system can contain hundreds and thousands of cores. The use of such large-scale systems requires the creation of special software. This is a significant challenge that generates many tasks aimed at developing common software for conducting large-scale computational experiments on high-performance supercomputer systems. Along with the traditional tasks that need to be solved for this purpose (calculation and grid decomposition, visualization of results, static and dynamic load balancing of processors), there is another problem that becomes more acute as computer systems become more complex. This is the problem associated with fault tolerance.

During long-running computations (several hours, days, or more) utilizing the entire supercomputer system, containing ~106 or more computational nodes and processing units, the probability of any part of it failing during the calculation approaches unity. For systems with a performance of ~1–10 Petaflops, the uptime is significant, and the problem is not relevant. However, for exaflop-scale systems, estimates indicate that the uptime is approximately 0.5–1 hour [15]. There is no reason to believe that hardware technology development will rectify this situation in the coming decades, as it is associated with fundamental technical constraints. Moreover, with the increase in the number of supercomputer nodes, the problem exacerbates. This means that without using intermediate data checkpointing, a failure almost inevitably prevents the completion of a lengthy calculation. Thus, the responsibility for data integrity and the ability to resume calculations in case of failure falls on the shoulders of program developers. The standard approach involving the use of "global checkpoints" is suitable for small-scale computing systems but becomes inadequate with a significant increase in the number of processor nodes [16]. In exaflop-scale systems, the time required to save a global checkpoint is of the same order as the uptime. Therefore, there is a significant likelihood that a new failure may occur during its writing or reading.

The second significant drawback of the standard approach is the need to repeat all calculation steps executed after saving the global checkpoint throughout the modeling domain, despite the fact that the failure of a single computing device leads only to the local loss of data in a small calculation area.

Approaches allowing for prolonged computations using local checkpoints were proposed in works [16, 17]. In this case, there is no need to pause the calculation for their recording, nor is there a need to restart the entire program after a failure. Data from local checkpoints are saved directly in the memory of the computing nodes involved in the calculation and those "adjacent" to each of them, ensuring the local nature of operation execution. Algorithms exhibiting the property of local processing of grid elements fall within the efficiently executable scope, including algorithms based on explicit difference schemes, including schemes obtained using the aforementioned hyperbolization procedure. The data recovery scheme after a failure also relies on the idea of hyperbolizing equations. If a computing node is excluded from the calculation process due to hardware or software failure, two or more of the pre-reserved backup nodes are activated in its place. Lost data due to failure are recovered from the operational or disk memory of "adjacent" nodes that have remained operational. An important property of hyperbolic equations is the finiteness of the disturbance propagation time and, consequently, the finiteness of the zone influencing the solution at the next time step at each local point (Fig. 6).

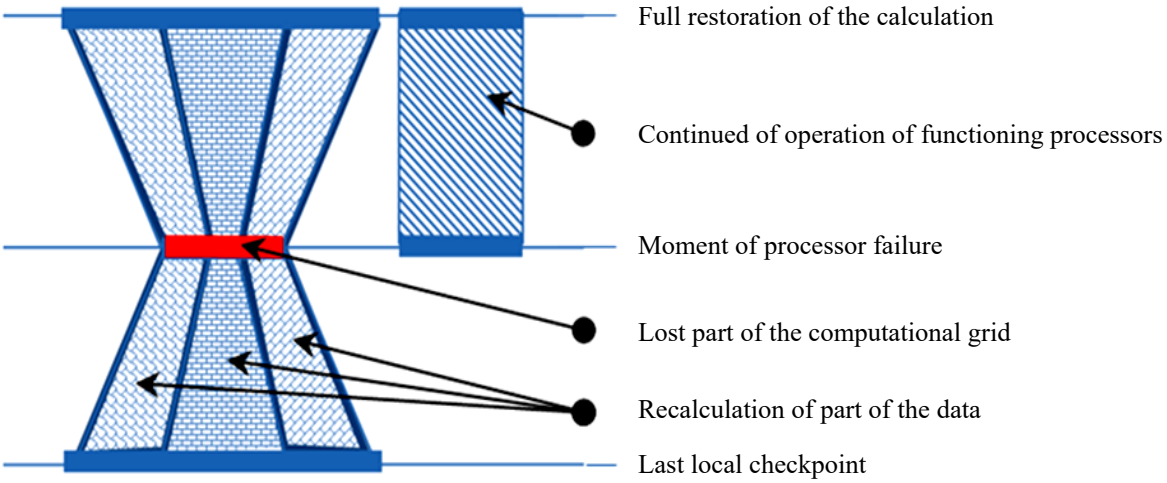


Fig. 6. Scheme of computation organization during calculation recovery

Thus, to recover from data loss, it will only be necessary to slightly expand the calculation zone, "rewind" back in time, and, taking the data from the previous local checkpoint, recalculate the required portion of the computational

domain quickly using multiple processors. The overall calculation does not stop during this time — all other nodes continue computing subsequent steps of model time.

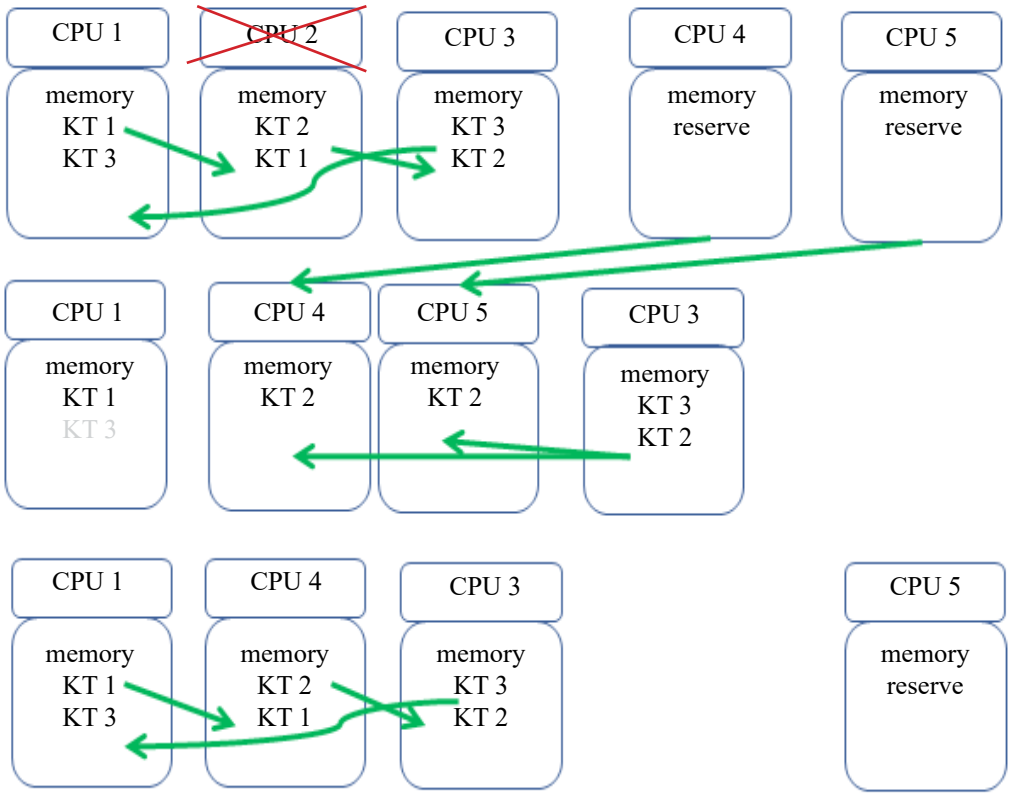


Fig. 7. General strategy for fault tolerance: distributed writing of checkpoint fragments and scheme for connecting backup nodes to recover the calculation

Therefore, after restoring the lost data, it is necessary to continue the accelerated execution of the calculation, allowing to "catch up" with the calculation performed by the remaining processors.

The scheme of distributed checkpoint writing and connection of backup processors is presented in Fig. 7. With such a scheme of data storage and local recalculation of the checkpoint, it is possible to restore the data and continue the calculation not only in the case of a single failure but also in the event of several simultaneous failures, independent in the metric of the computational grid.

Even in the three-dimensional case, only a few backup processors are required to correct a single failure. Moreover, all but one of them are returned to the reserve after the completion of the described procedure. The proposed algorithms have been successfully tested using the example of modeling the wave equation (5) with a source $F(x, t)$:

$$\frac{\partial^2 f}{\partial x^2} - \frac{1}{C^2}\frac{\partial^2 f}{\partial t^2} = F(x,t). \tag{2}$$

Let's present the estimate of the number of backup processors p_d obtained in the study [18] (3), required for complete recovery after a single failure at level 0 of the ongoing calculation to level 2 (the current level at which the ongoing calculation is "caught up") using checkpoint data from level 1 (Fig. 8):

$$p_d > \frac{1}{d+1}\frac{1}{k_2}\sum_{j=1}^{2} k_j \sum_{i=0}^{d} \alpha_j^i. \tag{3}$$

Here, the parameter $\alpha_j = 1 + 2\gamma\frac{k_j}{n_0}$; $\gamma = \frac{C\Delta t}{h}$ is the Courant number; c is the slope of the characteristics (5); t_0 is the time of failure; t_1 is the moment of model time at which the checkpoint was written; t2 is the time of complete recovery of the calculation; k_1 is the number of model time steps elapsed since the last checkpoint was written; k_2 is the number of steps to complete the recovery of the calculation, where $k_2 \le k_1$; n_0^d is the total number of grid points processed by the processor; d is the spatial dimension.

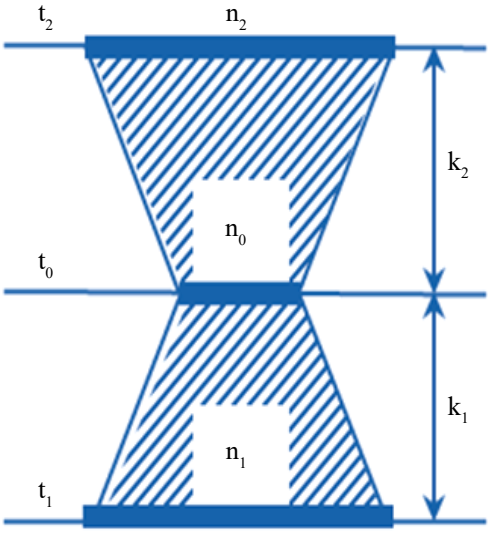


Fig. 8. Diagram of the spatial-temporal domain for estimating the number of backup processors

It is important to note that parallel algorithms, unlike sequential ones, are fundamentally nondeterministic. Nondeterminism leads to the impossibility of obtaining a complete set of tests for parallel programs, unlike sequential ones. Therefore, the use of basic methods becomes particularly important, such as data decomposition, pipeline parallelism, and collective decision-making, which have been extensively tested by numerous scientific communities. They work well but are not universal. Each of them is oriented towards its own class of tasks. The data decomposition method has the weakest constraint on potential acceleration when used, but it is poorly suited for implicit numerical schemes. The pipeline method has a fairly wide range of applications but imposes more significant limitations on acceleration. Collective decision-making has its own limitations on acceleration. These limitations are dictated by the ratio of the characteristic time for performing each task and the time for communication associated with the task.

Significantly, all basic methods are oriented towards a homogeneous computational process. In practice, when solving complex problems, such as on dynamically adaptive grids, it is substantially nonhomogeneous. The computational load may vary greatly even at one time step but at different stages of calculations (such as gas dynamics calculations, chemical kinetics calculations, decision-making optimization). Moreover, at each time step, the distribution of computational load among processors will differ for each of these computational stages. Additionally, different data distributions across processors may be required at different stages of one time step.

Fig. 9 illustrates the load balancing strategies used depending on the characteristics of the computational process. In the simplest case, when the computational load is homogeneous, static load balancing can be used for task distribution.

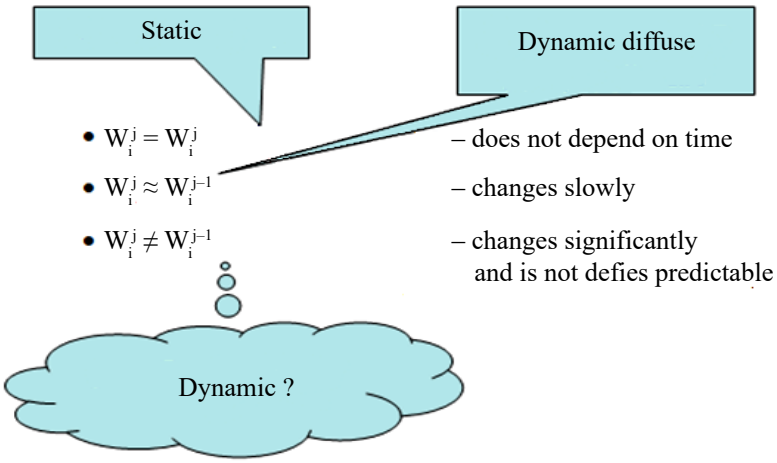


Fig. 9. Load balancing strategies depending on the ratio of computational load W_i^j for Calculating the *i*-th grid fragment on the j and $(j - 1)$ step

If the load changes smoothly, diffusion load balancing methods are used. However, there are tasks in which the load changes significantly and unpredictably, for example, combustion problems [19, 20]. For such tasks, standard load balancing methods prove to be ineffective.

The combustion problem (4) is described by the following equations:

$$\frac{\partial \mathbf{U}}{\partial t} + A\mathbf{U} = f, \quad \mathbf{U} = \left(p, \rho y^{(i)}, \rho u, \rho v, E\right)^T, \quad f = \left(0, \omega_i, 0, 0, 0\right)^T. \tag{4}$$

Here, ρ represents density; $y^{(i)}$ denotes the mass fractions of the i-th components; u, v denote velocities; p stands for pressure; E represents total energy; ω_i represents the component formation rates.

The problem is solved by partitioning into computational blocks (5–6) according to the Samarsky summation approximation method [21]:

I. Gas Dynamics Block:

$$\frac{\partial \mathbf{U}}{\partial t} + A\mathbf{U} = 0, \tag{5}$$

$$\frac{\mathbf{U}^{j+1} - \mathbf{U}^j}{\Delta t} + \frac{1}{2}\left(A\mathbf{U}^{j+1} + A\mathbf{U}^j\right) = 0, \tag{6}$$

II. Chemical Kinetics Block:

$$\frac{d\mathbf{U}}{dt} = f, \quad f = \left(0, \omega_i, 0, 0, 0\right)^T. \tag{7}$$

Practically all computations are concentrated in the zone of intense combustion, where methane mixes with oxygen. It is precisely there that intensive chemical reactions with light and energy release occur, and, accordingly, it is precisely in this zone that it is necessary to solve a multitude of independent stiff systems of ordinary differential equations (ODEs) (7), which requires significant computational power.

If the entire computational domain is uniformly distributed among processors according to the method of geometric parallelism, based on the needs of stages (5–6), then it turns out that at stage (7), only a few processors will be performing the main volume of computations, while all others will be practically idle most of the time (see Fig. 10).

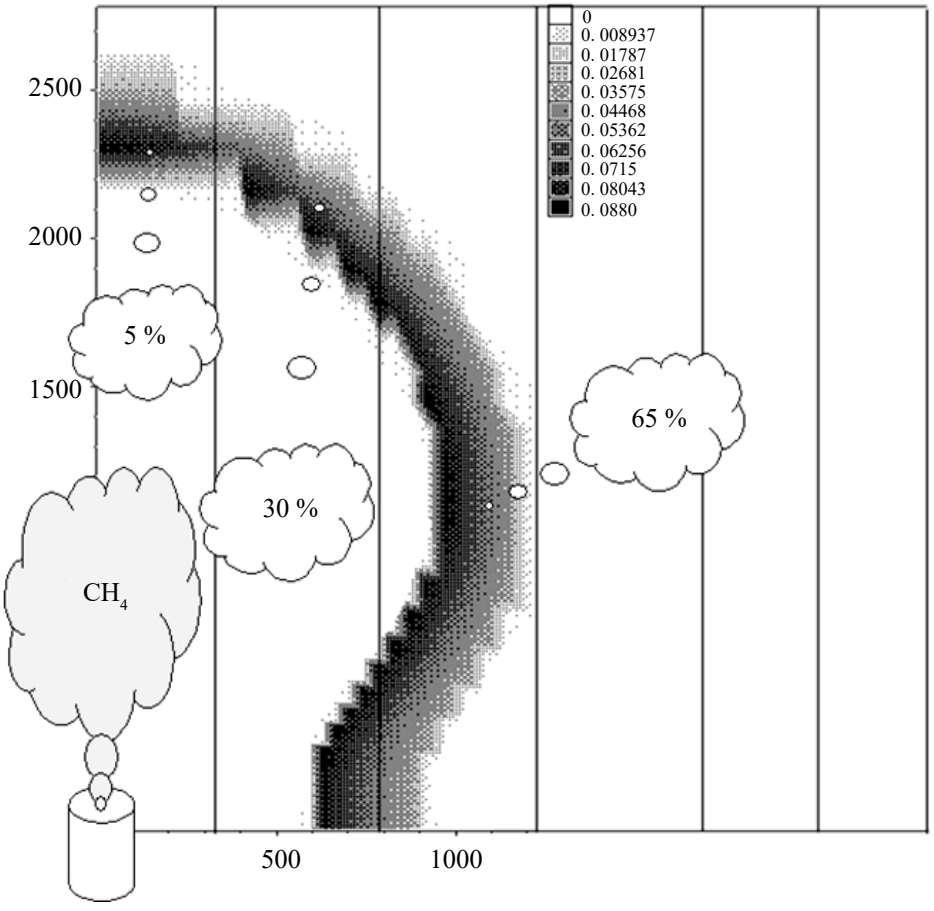


Fig. 10. Methane flare

This is a typical example of a problem that requires different computation distribution strategies at two stages within each time step of the model. Moreover, while static load balancing might suffice for the gas dynamics" stage, it is categorically inadequate for solving a multitude of ordinary differential equation (ODE) systems. Dynamic load balancing is required, for which standard methods are unsuitable. Special variants of collective decision-making methods should be employed, where multiple control processes are utilized, leading to the emergence of a complex task of ensuring their proper interaction. Each processor implements computational and control processes, greatly complicating the logic involved.

The corresponding method of dynamic load balancing was developed for problems similar to combustion and tested on loosely coupled clusters [19, 20]. An example of a loosely coupled cluster is the IMM RAS cluster (Fig. 3), consisting of Parcytec CC-12 and CC-32 systems connected by a thin communication channel. Inside each subsystem, there were 10-megabyte channels, while between them, there was a 1-megabyte channel.

The developed algorithms allowed for efficient solution of the combustion problem. The acceleration growth with an increase in the number of processors was nearly linear. Moreover, within this cluster, the processors were different, yet dynamic load balancing effectively equalized the load.

The same problem was successfully solved on a 1-teraflop computer MSC RAS MVS 1000M, which ranked 74th in the TOP-500 list in November 2002. The problem scales well with an efficiency of over 70 % when using up to 500 processors (Fig. 11). Further, the efficiency decreased to 62 %, which is associated with the use of a relatively small grid containing about 10 million nodes. For such grids, the calculation of the gas dynamics part leads to a decrease in efficiency. It is challenging to use large grids (~1 billion nodes) for these problems due to high computational power requirements necessary to conduct chemical kinetics calculations.

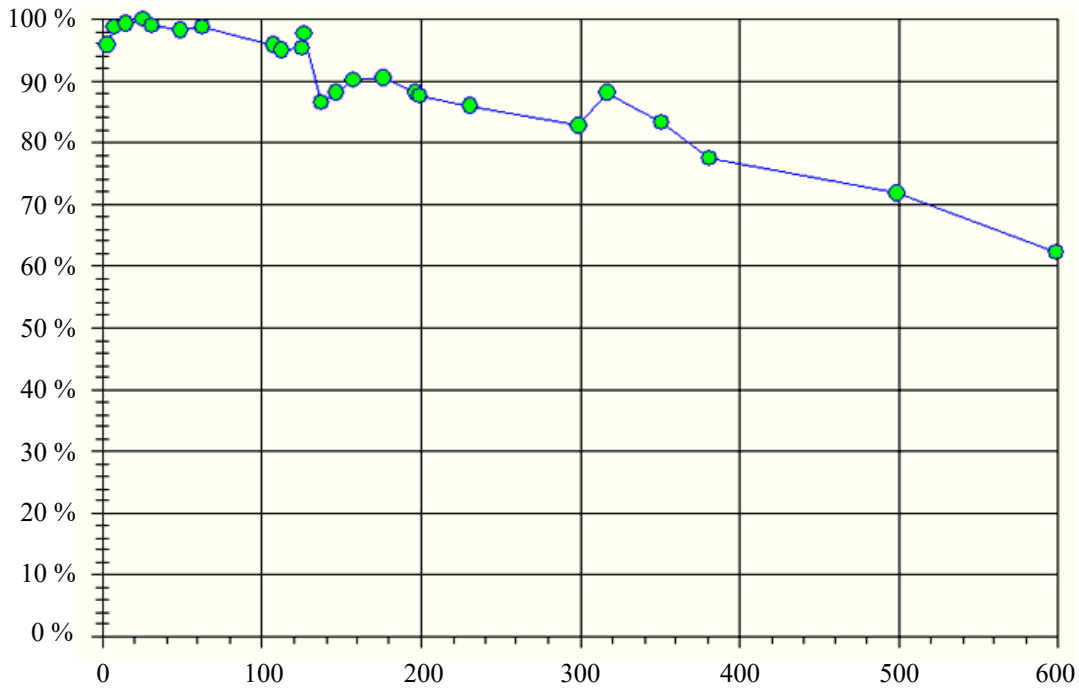


Fig. 11. Efficiency on the MSC RAS cluster

A significant fundamental problem associated with the use of supercomputers is the task of rational decomposition of computational grids. Research in the development of methods and software for working with large grids [22, 23] has been conducted at the institute since the 1990s.

To perform calulations, it is necessary to distribute grid variables and elements of the computational grid among computational nodes, ensuring uniform processor load while maintaining a low level of overhead associated with inter-processor communication. The complexity of the task increases non-linearly with the increasing size of the computational grids, especially unstructured ones. Figure 12 illustrates how, by using preliminary coarsening of the computational grid, one can transition from computational grids containing tens of billions of nodes to small macroscale grids, which are then divided into domains based on the number of processors.

To process macroraphs, one can utilize standard packages such as ParMETIS, Chaco, and others [24–27]. Grids containing several million nodes can be efficiently and swiftly processed using these tools. Such a hierarchical approach has proven effective for distributing grids among processors during computations, as well as for visualization, data storage, and the utilization of hybrid computational systems.

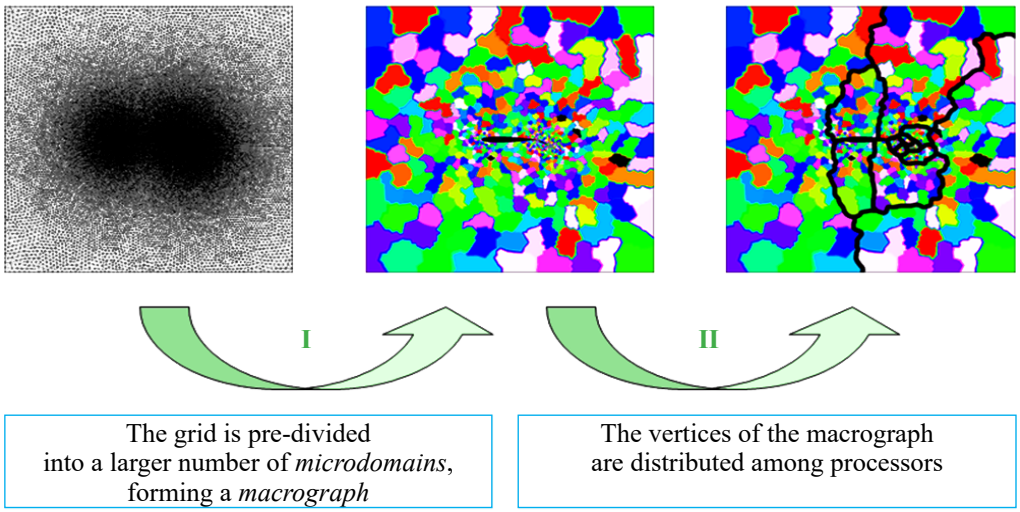


Fig. 12. Two-level partitioning of large grids

The problem of rational decomposition, already challenging for static grids, becomes significantly more complex when dealing with dynamically adaptive grids.

Let's illustrate the use of adaptive grids for solving the convection transport problem in two dimensions. In its two-dimensional form, the problem is described by the following equation:

$$\frac{\partial c}{\partial t} + v_x \frac{\partial c}{\partial x} + v_y \frac{\partial c}{\partial y} = 0, \tag{8}$$

where $c = c(x, y, t)$ is the concentration of the substance; t is time; (v_x, v_y) = const is the velocity vector of substance transport. The problem of droplet transport, which differs in its properties from those of the medium, was simulated [28]. A computational grid dynamically adjusting to the solution was employed. The result obtained on the adaptive grid was significantly more accurate than the one obtained on a uniform grid. Similar results were obtained in modelling processes of impurity transport in the aquatic environment, as exemplified by the simulation of an ecological disaster in the Azov Sea [29].

When using dynamically adaptive grids, it is necessary to dynamically perform grid partitioning and redistribute it among processors during computation. These functions are not supported by standard decomposition packages. Decomposition can be performed based on different principles: hierarchical decomposition algorithms (standard packages use them because they are the most universal); using space-filling curves, fractal curves (e. g., the Hilbert curve).

Dynamic balancing during computation is necessary for using computational systems in which the properties and topology of the interconnect are not a priori known, or for tasks for which the computational complexity of processing fragments of the computational grid is not a priori known.

Fig. 13 depicts domains formed as a result of hierarchical decomposition algorithm (Fig. 13 *a*, *b*) using the Metis package [26] and decomposition along the Hilbert curve (Fig. 13 *c*, *d*) for some two consecutive stages of grid adaptation. The second algorithm will be referred to as fractal. Fig. 13*a* and 13*c* correspond to one moment of grid adaptation, while Fig. 13 *b* and 13 *d* correspond to another. At the beginning of the calculation, the grid contained 64 base cells, each of which was divided into 16 elementary cells. The total number of elementary cells, according to the methodology [28], was approximately maintained throughout the calculation at 1024.

The quality of decomposition is acceptable in both cases, but attention should be paid to the volumes of data redistributed between processors in the case of hierarchical and fractal algorithms. When transitioning from decomposition (Fig. 13 *a*) to decomposition (Fig. 13 *b*), the volumes of redistributed data are larger than in the analogous transition when using decomposition along the fractal curve from (Fig. 13 *c*) to (Fig. 13 *d*). In the second case, the domain boundaries change more smoothly. Data movement affects only pairs of processors with adjacent numbers, whereas in the first case, it affects arbitrary pairs of processors.

Smaller computational costs for computing domains using fractal curves allow for more frequent invocation of the rebalancing procedure, further reducing the volumes of data transmitted with each rebalancing. It is worth noting that besides the Hilbert curve, there are many other space-filling curves (Fig. 14). For example, the Morton curve is popular due to its simplicity in calculating topology but, unlike the Hilbert curve, it does not preserve the proximity in physical

space of points with neighboring numbers on the curve. Nevertheless, methods based on fractal curves are not without drawbacks, as it is difficult to compute the neighborhood relationship of fragments located in different parts of the curve when using them.

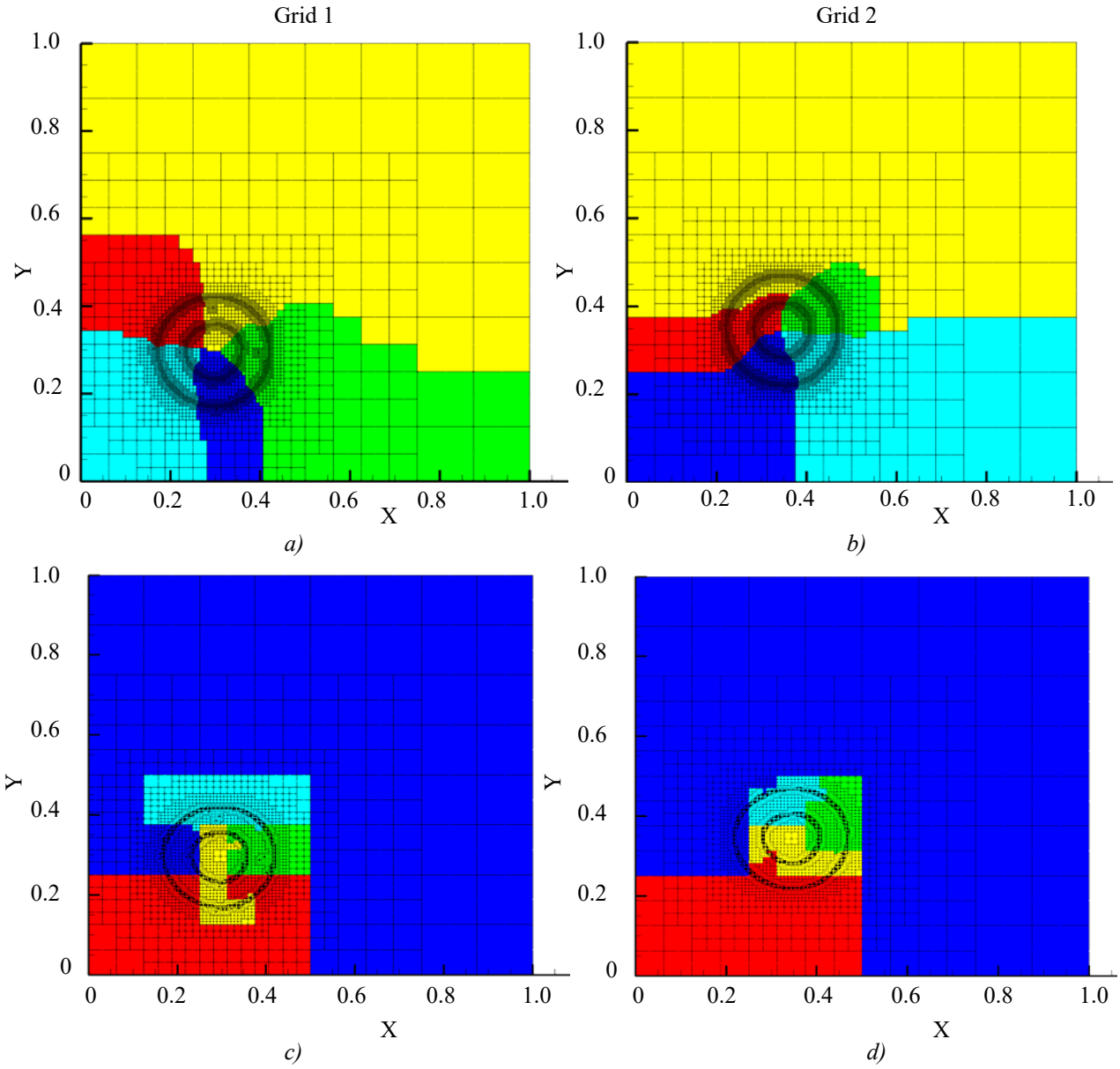


Fig. 13. Domains formed using the hierarchical method (*a*, *b*) and along the Hilbert curve (*c*, *d*)

Further investigation [30, 31] has shown that adhering to a certain grid discipline allows for the use of raster curves, rather than fractal ones, which prove to be more efficient. They enable the formation of grid domains by dividing the grid first into layers, then each layer into strips, and finally each strip into domains (Fig. 15).

The concept of raster blocks allows for smooth dynamics in domain boundary adjustments when using locally refined computational grids, thus reducing the volume of data transmission during grid reconstruction. An additional advantage of this approach is the ability to create simple and efficient parallel algorithms for decomposition, requiring minimal time expenditure for its execution.

This publication describes many, but not all, scientific studies conducted under the guidance and direct participation of Academician B.N. Chetverushkin, in which (except for the problem of relativistic jets), the authors of this article were also involved.

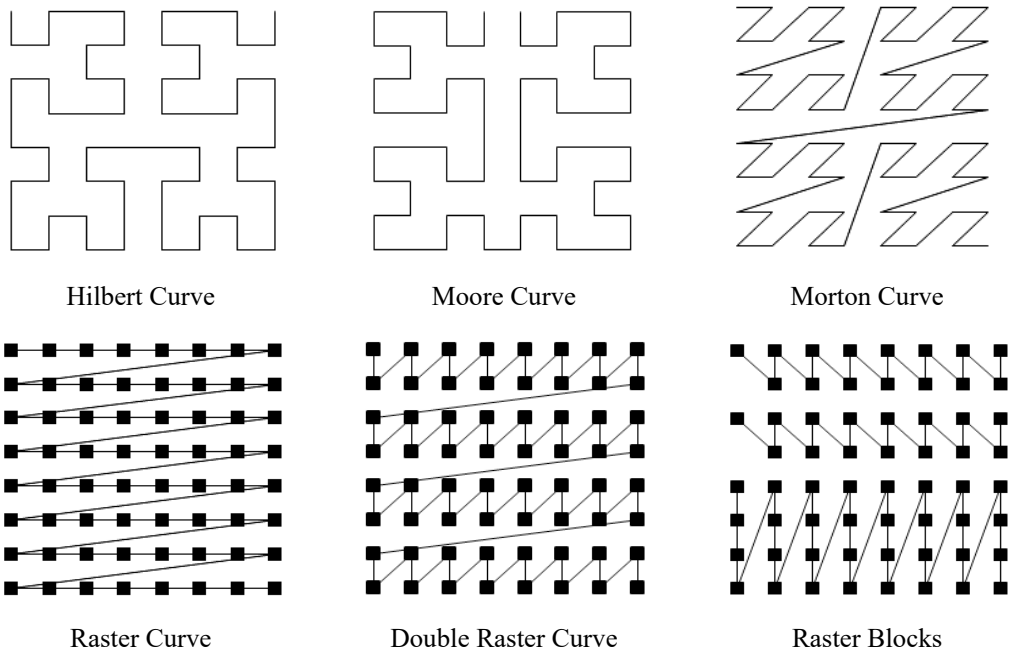
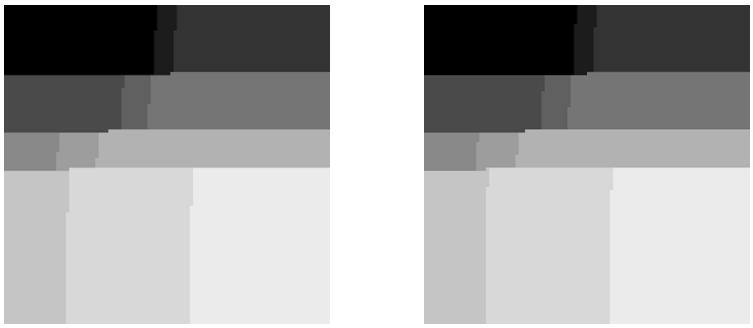


Fig. 14. Space-filling Curves



a) *b)*

Fig. 15. 2D domains formed using layered algorithm

References

1. Top500 Supercomputer Sites. (In Russ.). URL: http://top500.org (accessed: 21.02.2024).

2. Elizarova T.G., Chetverushkin B.N. Application of multiprocessor transputer systems to solve problems of mathematical physics. *Mathematical Modeling*. 1992;4(11):75–100. (In Russ.).

3. Galushkin A.I., Tochenov V.A. Transputer systems — the beginning of the formation of mass parallel computers in Russia. *Neurocomputers: development, application*. 2005;3:17–28. (In Russ.).

4. Transputer Systems – the Beginning of the Formation of Mass Parallel Computers in Russia. *Collection of abstracts of reports NSCF'2014*. Pereslavl-Zalessky: IPS named after A.K. Aylamazyan RAS; 2014. 45 p. (In Russ.). URL: https://2014.nscf.ru/TesisAll/0_PostMoore_Plenar/01_008_GalushkinAI.pdf (accessed: 21.02.2024).

5. Hybrid Computing Cluster K-100. (In Russ.). URL: http://www.kiam.ru/MVS/resourses/k100.html (accessed: 21.02.2024).

6. Davydov A.A., Latsis A.O., Lutsky A.E., Smolyanov Yu.P., Chetverushkin B.N., Shilnikov E.V. Multiprocessor computing system of hybrid architecture "MVS-Express". *Reports of the Academy of Sciences*. 2010;434(4):459–463. (In Russ.).

7. Center for Collective Use of IPM named after M.V. Keldysh RAS. (In Russ.). URL: http://ckp.kiam.ru (accessed: 26.02.2024).

8. Chetverushkin B.N., Churbanova N.G. On the application of the principle of geometric parallelism for the (α−β)-iterative algorithm. *Mathematical Modeling.* 1991;3(3):123–129. (In Russ.).

9. Abalakin I.V., Chetverushkin. B.N. Kinetic consistent difference schemes as a model for describing gas dynamic flows. *Mathematical Modeling*. 1996;8(8):17–36. (In Russ.).

10. Chetverushkin B.N. *Kinetically-consistent schemes in gas dynamics: a new model of viscous gas, algorithms, parallel implementation, applications*. Moscow: Moscow State University Publishing House; 1999. 232 p. (In Russ.).

11. Chetverushkin B.N. Hyperbolic quasi-gas dynamics system. *Mathematical Modeling*. 2018;30(2):81–98. (In Russ.).

12. Chetverushkin B.N., D'Aschenzo N., Saveliev A.V., Saveliev V.I. Kinetic model and equations of magnetic gas dynamics. *Computational Mathematics and Mathematical Physics*. 2018;58(5):716–725. (In Russ.).

13. Saveliev V.I., Chetverushkin B.N. Modeling problems of magnetohydrodynamics on high-performance computing systems. *Mathematical Modeling*. 2020;32(12):3–13. (In Russ.).

14. Hubble Space Telescope Snapshot. (In Russ.). URL: https://apod.nasa.gov/apod/ap000706.html (accessed: 26.02.2024).

15. Cappello F. Fault Tolerance in Petascale/ Exascale Systems: Current Knowledge, Challenges and Research Opportunities. *International Journal of High Performance Computing Applications*. 2009;23(3):212–226.

16. Chetverushkin B.N., Yakobovsky M.V. Computational algorithms and fault tolerance of hyperexascale computing systems. *Reports of the Academy of Sciences*. 2017;472(1):1–5. (In Russ.).

17. Chetverushkin B.N., Yakobovsky M.V. Computational algorithms and architecture of high-performance systems. *Preprints of the M.V. Keldysh Institute of Applied Mathematics*. 2018;52:12. (In Russ.).

18. Chetverushkin B.N., Yakobovsky M.V., Kornilina M.A., Semenova A.V. Numerical Algorithms for HPC Systems and Fault Tolerance Communications. *Computer and Information Science*. 2019;1063:34. (In Russ.). https://doi.org/10.1007/978-3-030-28163-2_3

19. Kornilina M.A., Yakobovsky M.V. Modeling the evolution of complex nonlinear systems on multiprocessor computing complexes. *Journal of Physical Chemistry*. 1995;69(8):1545–1548. (In Russ.).

20. Dorodnitsyn L.V., Kornilina M.A., Chetverushkin B.N., Yakobovsky M.V. Modeling gas flows in the presence of chemically active components. *Journal of Physical Chemistry*. 1997;71(12):2275–2281. (In Russ.).

21. Samarsky A.A. *Theory of Difference Schemes*. Moscow: Nauka; 1989. 616 p.

22. Yakobovsky M.V. Processing grid data on distributed computing systems. *Issues of atomic science and technology. Series Mathematical modeling of physical processes*. 2004;2:40–53. (In Russ.).

23. Yakobovsky M.V. Incremental graph decomposition algorithm. *Bulletin of Lobachevsky University of Nizhny Novgorod. Series Mathematical Modeling and Optimal Control*. 2005;1(28):243–250. (In Russ.).

24. Fiedler M. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*. 1975;25(100):619–633. URL: http://www3.math.tu-berlin.de/Vorlesungen/SS14/MatricesGraphsPDEs/paper_for_students/CZMathJ-25-1975-Fiedler%20copy.pdf (accessed: 27.02.2024).

25. Hendrickson B., Leland R. A Multilevel Algorithm for Partitioning Graphs. *Supercomputing '95 Proceedings*. San Diego, CA, 1995. URL: http://www.leonidzhukov.net/hse/2016/networks/papers/MultilevelAlgorithmPartitioningGraphs.pdf (accessed: 27.02.2024).

26. Karypis G. Family of Graph and Hypergraph Partitioning Software URL: http://glaros.dtc.umn.edu/gkhome/views/metis/ (accessed: 27.02.2024).

27. Pothen A., Simon H.D., and Kang-Pu P.L. Partitioning Sparse Matrices with Eigenvectors of Graphs. Report RNR-89-009, July 1989. URL: http://snap.stanford.edu/class/cs224w-readings/Pothen89Partition.pdf (accessed: 27.02.2024).

28. Sukhinov A.A. *Mathematical Modeling of Impurity Transport Processes in Liquids and Porous Media*. Candidate of Physical and Mathematical Sciences dissertation. Moscow, 2009. 24 p. (In Russ.).

29. Sukhinov A.A. Reconstruction of an Ecological Disaster in the Azov Sea Based on Mathematical Models. *Mathematical Modeling*. 2008;20(6):15–22. (In Russ.).

30. Kornilina M.A., Yakobovsky M.V. Overhead Costs Assessment for Calculations on Locally Refined Grids. *Preprints of the M.V. Keldysh Institute of Applied Mathematics*. 2022;102:36 p. (In Russ.). https://doi.org/10.20948/prepr-2022-102

31. Grigoriev S.K., Zakharov D.A., Kornilina M.A., Yakobovsky M.V. Dynamic Load Balancing Using Adaptive Locally Refined Grids. *Mathematical Modeling*. 2023;35(12):69–88. (In Russ.). https://doi.org/10.20948/mm-2023-12-05

Received 19.02.2024
Received 07.03.2024
Accepted 11.03.2024

About the Authors:

Mikhail V. Yakobovskiy, Deputy Director for Scientific Work, Institute of Applied Mathematics named after M.V. Keldysh of the Russian Academy of Sciences (4, Miusskaya Sq., Moscow, 125047, RF), ORCID, MathNet, ScopusID, lira@imamod.ru

Marina A. Kornilina, Researcher, Institute of Applied Mathematics named after M.V. Keldysh of the Russian Academy of Sciences (4, Miusskaya Sq., Moscow, 125047, RF), ORCID, MathNet, ScopusID, mary@imamod.ru

Contributions of the co-authors:

M.V. Yakobovskiy — direct participation in all described research activities except for modeling gravitational potential.

M.A. Kornilina — involvement in work on fault tolerance and combustion modeling, analysis of materials of the dissertation council, preparation of manuscript materials.

Conflict of interest statement

The authors do not have any conflict of interest.

All authors have read and approved the final manuscript.

Поступила в редакцию 19.02.2024
Поступила после рецензирования 07.03.2024
Принята к публикации 11.03.2024

Об авторах:

Якобовский Михаил Владимирович, заместитель директора по научной работе, Институт прикладной математики им. М.В. Келдыша Российской академии наук (РФ, 125047, Москва, Миусская пл., 4), ORCID, MathNet, ScopusID, lira@imamod.ru

Корнилина Марина Андреевна, научный сотрудник, Институт прикладной математики им. М.В. Келдыша Российской академии наук (125047, Москва, Миусская пл., 4), ORCID, MathNet, ScopusID, mary@imamod.ru

Заявленный вклад соавторов:

М.В. Якобовский — личное участие в проведении всех описываемых исследований за исключением моделирования гравитационного потенциала.

М.А. Корнилина — участие в работах по отказоустойчивости и моделированию задачи горения, анализ материалов деятельности диссертационного совета, подготовка материалов статьи.

Конфликт интересов

Авторы заявляют об отсутствии конфликта интересов.

Все авторы прочитали и одобрили окончательный вариант рукописи.