

Суперкомпьютерный консорциум университетов России  
Российская академия наук



# **СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ**

Труды международной конференции

27–28 сентября 2021 г., Москва



---

МОСКВА – 2021

УДК 519.7  
ББК 22.18  
С89

Под редакцией члена-корреспондента РАН *Вл. В. Воеводина*

**Суперкомпьютерные дни в России** : Труды международной конференции. 27–28 сентября 2021 г., Москва / Под. ред. Вл. В. Воеводина. – Москва : МАКС Пресс, 2021. – 192 с.

ISBN 978-5-317-06698-7 e-ISBN 978-5-317-06697-0  
<https://doi.org/10.29003/m2454.RussianSCDays2021>

Данный сборник содержит полные статьи на русском языке, короткие статьи и аннотации стендовых докладов, включенных в программу Международной конференции «Суперкомпьютерные дни в России».

УДК 519.7  
ББК 22.18

**Russian Supercomputing Days: Proceedings of the International Conference. September 27–28, 2021, Moscow, Russia** / Ed. by Vl. Voevodin. – Moscow : MAKS Press, 2021. – 192 p.

ISBN 978-5-317-06698-7 e-ISBN 978-5-317-06697-0  
<https://doi.org/10.29003/m2454.RussianSCDays2021>

The book contains full papers in Russian, short papers and poster abstracts included in the agenda of the International Conference “Russian Supercomputing Days”.

*Подробную информацию о конференции можно найти в сети Интернет  
по адресу <http://RussianSCDays.org>*

ISBN 978-5-317-06698-7  
e-ISBN 978-5-317-06697-0  
<https://doi.org/10.29003/m2454.RussianSCDays2021>

© Авторы статей, 2021  
© МГУ имени М. В. Ломоносова, 2021  
© Оформление. ООО «МАКС Пресс», 2021



**Полные и короткие статьи**

# Approaches, services, and monitoring in a distributed heterogeneous computing environment for the MPD experiment\*

A.A. Moshkin, I.S. Pelevanyuk, D.V. Podgainy, O.V. Rogachevsky,  
O.I. Streltsova, M.I. Zuev

Joint Institute for Nuclear Research

The article discusses some issues of creating a heterogeneous computing environment for processing and storing data from the MPD experiment of the NICA megaproject, which is being implemented at the Joint Institute for Nuclear Research. The creation of such an environment requires combining many different concepts and techniques based on the paradigm of distributed computing and a hierarchical data processing and storage system. The created environment must meet the requirements of high performance, reliability and availability for various groups of users, which is achieved, among other things, through an advanced monitoring system. The DIRAC Interware platform, which supports computing at all levels: workload management, data management, workflow management, user management, configuration management, etc., was chosen as the basis for creating such an environment. Owing to DIRAC, computational resources and the hierarchical hyperconverged data processing and storage system of the “Govorun” supercomputer were implemented in the heterogeneous computing environment. Moreover, the “Govorun” supercomputer not only plays a key role in the created environment, but also, due to its flexibility, provides an opportunity to test in practice both the latest architectural solutions and software solutions in the field of computing and data processing.

*Keywords:* data processing and storage, middleware for distributed computing, user job monitoring, high-performance computing, computing for megascience projects.

## 1. Introduction

One of the main challenges in scientific computing is the processing of large amounts of data for experiments in high-energy physics, which can be classified as one of the most demanding scientific tasks in terms of computational resources. The amount of data obtained in modern experiments reaches several tens of petabytes per year. The data processing procedure includes a number of operations, such as the reconstruction of events, simulation of the installation, and data analysis, each of which is a rather complex task and requires significant computational resources. For example, in the ATLAS experiment at the Large Hadron Collider (LHC), an average of 600 events are written per second, which results in a constant data stream with a transfer rate of 300 MB/s, demanding further processing. The reconstruction time for one event is 10 s, therefore, only for the reconstruction of events at the same rate, the simultaneous operation of 6,000 processors is required [1]. Even more resources are needed for experimental installation simulation tasks, which are an integral part of data analysis. As a result, the computing resource demands for LHC data processing went far beyond the capabilities of a single, even large, data center and led to the creation of a distributed data processing system. At the same time, the time required to process data and quickly obtain physical results directly depends on the performance of the computing environment as a whole. With this in mind, computing is currently being created for the NICA (Nuclotron-based Ion Collider Facility) megaproject, which is being implemented at the Joint Institute for Nuclear Research (JINR) [2]. NICA is a new accelerator complex designed to study the properties of dense baryonic matter. After putting the NICA collider into operation, JINR scientists will be able to create in the Laboratory a special state of matter in which our Universe stayed shortly after the Big Bang, i.e., Quark-Gluon Plasma (QGP). The first experiment at the NICA complex will be MPD (Multi-Purpose Detector) [3].

---

\* The studies in this direction were supported by the RFBR special grant (“Megascience – NICA”), No.18-02-40101.



The MPD apparatus is designed as a  $4\pi$  spectrometer capable of detecting charged hadrons, electrons and photons in heavy-ion collisions at high luminosity in the energy range of the NICA collider. To reach this goal, the detector will comprise a precise 3D tracking system and a high-performance particle identification (PID) system based on time-of-flight measurements and calorimetry.

The basic design parameters are determined by physical processes in nuclear collisions at NICA and by several technical constraints guided by a trade-off of effective tracking and PID against a reasonable material budget. At the design luminosity, the event rate in the MPD interaction region is about 7 kHz; the total charged particle multiplicity exceeds 1,000 in the most central Au+Au collisions at  $\sqrt{s_{NN}} = 11$  GeV.

The main program blocks of the experiment are the following systems:

- Data acquisition (DAQ, online);
- Multilevel data selection (trigger, online);
- Data processing (particle tracks recognition and reconstruction, online+offline);
- Physical analysis (offline).

The expected data flow is up to 50 GB per second. Two sessions of the accelerator operation per year, each lasting up to 4 months, are planned. The total volume of incoming data per year can reach 1 exabyte. To reduce the size of the required storage, it is necessary to carry out as many stages of data processing in real time as possible, i.e., to implement in real time (during data acquisition) not only a trigger, but also the maximum possible number of stages of the recognition and reconstruction of elementary particle tracks.

For efficient data processing of the MPD experiment, a heterogeneous, geographically distributed computing environment is currently being created on the basis of the DIRAC Interware [4]. The DIRAC Interware is an open-source development platform for the integration of heterogeneous computing and storage resources. It was originally developed for the LHCb computing infrastructure, but later DIRAC was released as a general-purpose solution for scientific groups. Right now, it is used by many experiments in high-energy physics, particle physics and astronomy: LHCb, Belle-II, BES-III, CTA, CLIC, ILC. The purpose of DIRAC is to ensure access to various computing and storage resources through standard interfaces, i.e., web, command line, API, and REST. Another purpose is to provide a set of standard tools for workload management, data management, accounting, workflow management, user management, etc. An important point for the stability of such an environment is the advanced monitoring of tasks performed on various computing sites and data flows between the sites.

At the moment, the heterogeneous, geographically distributed computing environment includes the Tier1 and Tier2 grid sites, the JINR MICC cloud component, the computing clusters of VBLHEP (Veksler and Baldin Laboratory of High Energy Physics JINR) and UNAM Mexico, however, the key element of the environment being created is the “Govorun” supercomputer [5, 6], which allows one to carry out massive calculations, as well as, due to the flexibility of its architecture, IT research in order to create an effective computing model for the NICA megaproject [7].

This article briefly describes an approach to creating a geographically distributed computing environment that currently allows automating the process of modeling physical events for the future MPD experiment, and after the launch of the NICA accelerator complex, it will be used to process real experimental data. The article also presents the main properties of DIRAC Interware as a basic software element for creating such an environment and describes the data processing model on the “Govorun” supercomputer as one of the key elements of this environment.

## 2. Data processing model on the “Govorun” supercomputer

The main HPC resource of JINR is the “Govorun” supercomputer, which comprises two components, GPU and CPU [5, 6]. The GPU component is based on 5 NVIDIA DGX-1 servers with 8 NVIDIA Tesla V100 GPUs. The CPU component contains 21 RSC Tornado nodes based on Intel® Xeon Phi™ and 88 RSC Tornado nodes based on Intel® Xeon® Platinum 8268 and has two Intel® SSDs DC P4511 (NVMe, M.2), 2TB each. Additionally, the “Govorun” supercomputer has 4 special nodes with 12 high-speed, low-latency solid state drives Intel® Optane™ SSD DC P4801X 375GB M.2 series with Intel® Memory Drive Technology (IMDT), which allows getting 4.2 TB for very hot data per node.

The internal network of the “Govorun” supercomputer consists of three main parts: a communication and transport network, a control and monitoring network and a task control network. The communication and transport network uses Intel OmniPath 100 Gbps technology. The network is built on a “thick tree” topology based on 48-port Intel OmniPath Edge 100 Series switches with full liquid cooling. The control and monitoring network enables the unification of all compute nodes and the control node into a single Fast Ethernet network. This network is built using Fast Ethernet switches HP 2530-48. The job control network connects all compute nodes and the control node into a single Gigabit Ethernet network. The job control network is built using HPE Aruba 2530 48G switches.

The total peak performance of the “Govorun” supercomputer is 780 TFlops for double precision (12th and 21st places in the current TOP50 list <http://top50.supercomputers.ru/list>). The supercomputer also has local Lustre storage with a capacity of 288 TB. The “Govorun” supercomputer took 31st place in the IO500 list (<https://io500.org/>) with a bandwidth of more than 35 GiB/s and a metadata performance of more than 230 kIOP/s.

It should be noted here that the CPU component of the “Govorun” supercomputer is a hyperconverged software-defined system and ranks 16th in “10 node challenge” in the current edition of the IO500 list (<https://io500.org/list/isc21/ten>) under the DAOS (Distributed Asynchronous Object Storage) operation. The property of hyperconvergence makes the “Govorun” system very flexible for customizing the user’s job, ensuring the most efficient use of the computing resources of the supercomputer.

It is noteworthy that modern HPC systems are used not only as traditional computing environments for performing massively parallel calculations, but also as systems for Big Data analysis and artificial intelligence tasks that arise in different scientific and applied problems. At the same time, despite the increase in supercomputer performance, memory and data storage bandwidths are becoming bottlenecks. To accelerate work with data for tasks of different types, solved on the “Govorun” supercomputer, a hierarchical hyperconverged data processing and storage system with a software-defined architecture was developed and implemented. This system is based on the disaggregated RSC [8] solution for data processing and storage, which enhances the efficiency of solving user problems, as well as increases the efficiency of using both computational resources and data storage resources. It realizes a new paradigm for working with data, i.e., the integration of computing elements and novel types of data storage elements (Intel Optane PMem, Intel SSD) into a unified computing environment.

According to the speed of accessing data, the system is divided into levels available for the user’s choice, namely, a hot layer implemented on the basis of Intel Optane PMem, a warm layer based on Intel SSD NVMe under the management of the Lustre file system, a warm layer implemented as “an on-demand storage system”, which can be managed by different file systems defined by the user, a cold layer implemented on HDD of sufficient volume, which ensures data storage, but does not meet the peak requirements of a computational task. Each layer of the data storage system under development can be used both independently and as part of data processing workflows. It should be pointed out that a part of the cold storage is managed by the geographically distributed EOS file system (<https://eos-web.web.cern.ch/eos-web/>), which allows connecting the data processing and storage system implemented on the “Govorun” supercomputer to geographically distributed storages, the so-called Data-Lakes. The hierarchical hyperconverged data processing and storage system with a software-defined architecture represents an extremely convenient tool for processing large amounts of data, which can cardinaly speed up work with data and is already actively used for the experiments of the NICA mega-project (Figure 1).

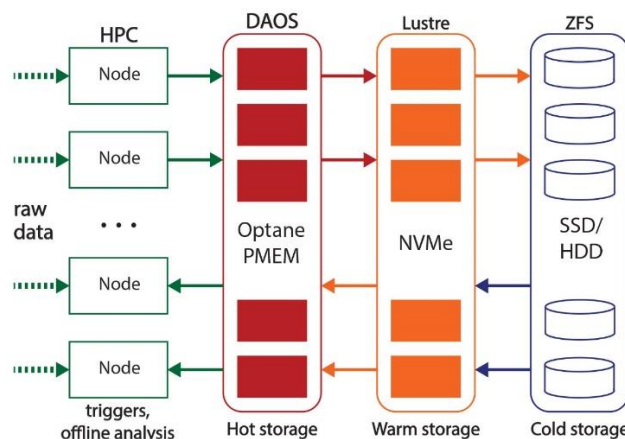


Figure 1. Multilevel system for processing and storing data on the “Govorun” supercomputer

It can be illustrated by the experience of data mass production tasks for the NICA MPD experiment. First of all, the use of Intel SSD allowed us to organize two fast and stable on-demand storages under the Lustre and NFS file systems on the basis of RSC technologies. Secondly, the use of Intel Optane PMem made it possible to utilize all computational cores on the nodes to solve MPD tasks that require a large amount of RAM per core, while without Intel Optane PMem less than half of the computational cores on the nodes can be used for such tasks. And thirdly, the use of an expanded test site on DAOS (Distributed Asynchronous Object Storage) showed a significant acceleration of working with data (up to two times) and the possibility of forming a unified data buffer for various JINR computing clusters.

### 3. DIRAC at JINR

A service based on this platform was deployed and configured at the Joint Institute for Nuclear Research in 2016. Various tests were carried out to estimate its possibilities and performance. By 2019, standard workflow tests related to general Monte-Carlo generation were successfully completed, which proved the possibility of using DIRAC for real scientific computations [9]. At that time, MPD started experiencing the necessity of massive computing to perform centralized Monte-Carlo generation for the needs of scientific groups. At first, only Tier1 and Tier2 were used for that work, but later the “Govorun” supercomputer and the VBLHEP cluster were integrated into the system. The JINR EOS storage system was integrated and accessed via the root protocol. Authentication is based on x509 certificates, and authorization is regulated by both DIRAC and VOMS (Virtual Organization Membership Service).

There are several reasons why DIRAC Interware platform has been chosen as a tool for organization of distributed computing for MPD. First of all, it is the only solution that supports computing on all levels: workload management, data management, workflow management, user management, configuration management, etc. Second, DIRAC proved to be able to handle hundreds of thousands of jobs daily on LHCb infrastructure. It supports horizontal scaling by design: with increasing number of jobs running it is possible to setup additional copies of core services responsible for jobs and data processing. And third, it has wide and active community of users across different organizations and experiments.

Right now at JINR, DIRAC is handling two important aspects of massive computing, i.e., workload management and data management. Workload management relies on the pilot mechanism. The main idea of this approach is that user jobs are never sent directly to a queue on the computing resources. Instead, a special program called “Pilot” is sent. Its function is, firstly, to occupy a slot in the queue, and secondly, to start execution on a particular work node. When it starts, it checks various parameters of the work node it is running on. These parameters are then sent to the DIRAC Matcher service with a request for a user job. DIRAC Matcher chooses a job that meets the specified parameters and submits it to the pilot. After that, the pilot initiates the user job as a child process. It enables the monitoring of the execution process, and in the case of a failure, a corresponding message is sent to DIRAC to mark the job as failed. Another feature of this approach is the simplification of the integration of heterogeneous computing resources into a unified system. Integrating a computing resource means finding a way to run DIRAC Pilot there.

The DIRAC Data Management system is also actively used during MPD generation campaigns. Surely, all files generated by user jobs can be saved to a local storage, however, it would require an additional step to go through all local storages and collect result data. The use of the DIRAC Data Management system allows writing data to a central storage element accessible via the root protocol. After all, the files are present in a storage element from where they can be accessed either via grid protocols or through local access.

Currently, DIRAC at JINR provides all the necessary tools to support massive and efficient high-throughput job submission. It can be improved by making more extensive use of DIRAC complex features, such as the Workflow Management system, and by integrating more heterogeneous resources.

#### **4. Organization of data processing in a distributed heterogeneous computing environment**

Each production (mass data physical event generation and reconstruction) goes through several steps. At each step, there are specific tools and approaches to its implementation. The following steps are distinguished:

1. Physicists make a request for data generation in the forum thread <https://mpdforum.jinr.ru/c/mcprod/26>. This request should contain all information about which software to use, which parameters to apply, how much data to generate, as well as other information relevant to generation.

2. When a request appears, the production manager needs to check if there is enough information to start generation. The first step is to generate a small fraction of the requested data and submit it to physicists for review. This step is important since without it the whole generation can go wrong and end up useless. Partial generation can be performed directly on one computing resource without using a distributed infrastructure for the sake of simplicity. In our case, it is usually done on the “Govorun” supercomputer. If the results are satisfactory, the next step should be taken.

3. Preparation for distributed execution. At this step, the production manager needs to analyze the profile of the computational task and decide how to run jobs on different resources. The first question is which software to use. At best, the software is already hosted in CERN Virtual Machine File System (CVMFS, <https://cernvm.cern.ch/fs/>). However, this is not always the case, and the production manager is responsible for compiling and testing the applied software on all resources that will be used during the production campaign. Another important aspect is to study the profile of an average computational job. By this we mean figuring out how long it takes to complete a job, what is RAM and network consumption, how much space is required in a local storage on the work node. With this information, it is possible to configure the execution of a workload on heterogeneous resources. To support this process, a special system was developed. This system will be described in more detail in the next section.

4. Distributed execution on heterogeneous computing resources. At this step, the production manager generates jobs to submit in the DIRAC system. It is important to keep in mind that some jobs can be completed with an error for various reasons. The recomputation of failed jobs should be easily accomplished. It can be achieved either by rescheduling failed jobs directly in the DIRAC system, or submitting new jobs with some fixes. During this step, the production manager keeps an eye on DIRAC Job Monitoring to identify any malfunctioning resources. Previous misbehavior is noticed, and fewer jobs need to be resubmitted or rescheduled, which means better resource utilization. During this step, it is good to have an approach and a tool to monitor the performance of different resources. A special approach was elaborated to provide this kind of data to production managers, resource administrators, and management. This approach will be described in the section 5.

5. When the generation campaign is done, the production manager needs to verify that all files are correctly placed in a storage. If not, he needs to resubmit jobs that for some reason have not written data to the storage. This work usually takes little time and in some cases can be done directly on one of the computing resources without using DIRAC to manage the jobs. After that, a message about the completion of the generation campaign should be posted on the forum, and physicists can use the generated data for their research.

## 5. Individual user job monitoring

When running a computer task on a local computer, it is quite easy to check its CPU load, RAM usage, and network use with simple built-in tools. However, when jobs are running in a distributed system, this problem is not so easy to tackle. It is possible to login to the work node for manual verification, but there is no direct access to many resources. Another issue is that it is usually useful to have a history of the load, i.e., a profile. Standard tools do not provide this.

To provide such a tool to DIRAC users, a special approach was proposed. It was necessary to give control over monitoring to users in those cases when they really needed it. The issue with total monitoring is related to data flow and future data analysis. For each job, CPU load, RAM usage, network I/O usage metrics should be collected and written every 30 seconds. This means that an average job that runs for 6 hours produces around 3,000 records. While for an average production campaign with 30-40 thousand jobs, the number of records may reach 100 million.

To solve this problem, a special Python application was developed to collect metrics from a running process. The procedure for enabling this monitoring requires the user to run a computational process that should be monitored in a background mode. The PID of the process should be passed to the developed application. The application will check the metrics of the running process and its children and submit them to the InfluxDB database. The collected data is accessed via the standard Chronograf interface (Figure 2).

With such monitoring, it is clearly visible that the job runs in three stages: the first short stage is generation, the second stage is simulation (it produces a large amount of data), and the third stage is reconstruction, which reads all the data and produces the result. During the reconstruction part, a memory leak in the software was detected.

The monitoring service enables users to check how their workload is behaving on the work node. It can be used when needed, and the results will be presented in a web interface with the possibility to form custom queries to the database to find jobs with the highest RAM usage, to define the average CPU usage of each job, or I/O consumption per one job.



Figure 2. Job monitoring. The plot demonstrates RAM and I/O usage.

## 6. Analysis of computing resources performance

When real user jobs run on a distributed infrastructure, execution information is a valuable resource for performance analysis and the comparison of different computing resources. A new approach to analyze and visualize this data was proposed at JINR [10].

Each user job submitted by the user runs not directly on a work node, but under a special process called DIRAC Pilot. When the pilot process starts, it checks all the relevant parameters of the work node it is running on: total RAM, free RAM, free disk space and CPU performance. CPU performance is measured by a special benchmark called DIRAC Benchmark 12 (DB12). This benchmark provides an estimate of the speed of a single CPU core. The data is saved to the internal DIRAC database. After

completing the user job, DIRAC saves information about its wall-time duration. When there are thousands of similar jobs running in the system, it is possible to compare their durations on different resources.

The approach was elaborated for extracting data on completed jobs from the DIRAC database and presenting it in a comprehensive form. The data is visualized in a two-axis scatter plot. The X axis represents the DB12 test result for a particular job. The Y axis represents the wall time of a particular job. Thus, each job can be represented by a single dot in the plot. To add even more information, each dot is colored according to the resource the job was completed on. An example of the plot is shown in Figure 3.

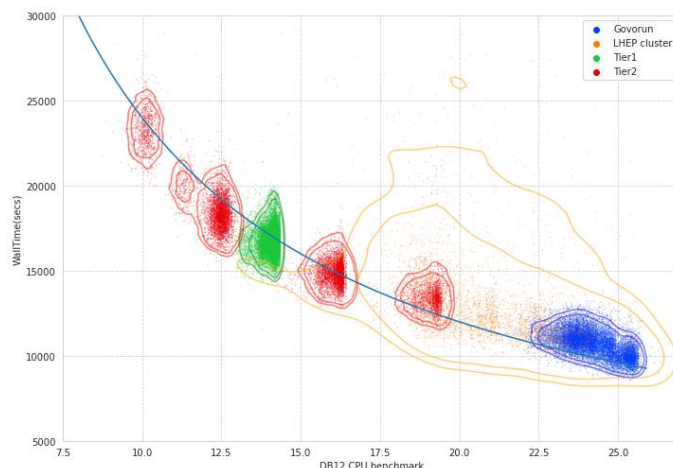


Figure 3. Example of the scatter plot with information about job duration, DB12 results, and resource ownership

The plot demonstrates that the DB12 results correlate well with the wall time. The blue line in the plot was empirically chosen to show where the dots should be placed in the ideal condition. From this plot, we can see that most of the resources give consistent duration time, except for the LHEP cluster. This is due to the lower network speed between the LHEP cluster and the storage system where result data should be placed. The two colored borders around the dot clusters represent the area with 90% and 95% of all jobs. It means that we can predict the duration of a particular job on different resources and plan the duration of the generation campaign more precisely.

This approach has a great advantage since it does not require submitting an artificial workload to the computing resources. Right now, it is required to prepare data and upload it to a Jupyter notebook for visualization. A dedicated system providing this functionality is currently under development. It should help to compare different processors and work nodes with each other on a real user workload.

## 7. Conclusion

The creation of a distributed heterogeneous computing environment based on the DIRAC software with advanced monitoring tools has pursued two main goals, namely, to meet the current need for the generation and reconstruction of model data and to create computing elements for the real MPD experiment. The created environment has made it possible to perform almost 857,000 jobs for the MPD experiment to date. Each job ran on average 6 hours 15 minutes on one core. The total use of computing power was 6 million Mega HEP-SPEC06 days. This effort has generated more than 650 million events, over 250 million of which have also been reconstructed (over 270 TB in storage). The practice of using various computing resources of JINR and other institutes of the MPD collaboration has shown that at the moment the most flexible resource is the “Govorun” supercomputer. This property is achieved due to the unique hardware design of the “Govorun” supercomputer, which comprises a hierarchical data processing and storage system and compute nodes with a large amount of RAM, as well as the RSC [8] software, enabling the creation of systems on demand for a specific user task. About 40% of the total number of tasks has been performed directly on the “Govorun” supercomputer.

The solution to the computing problem of the NICA megaproject required the creation of a number of research polygons, which would allow testing in practice both the latest architectural solutions and



software solutions in the field of computing and data processing. The “Govorun” supercomputer plays the role of the main platform for such research. At the moment, a polygon of 8 servers containing Intel Optane PMem has been created, on which DAOS has been deployed. Research carried out at this testing site has shown a substantial performance increase on the same hardware. The IO500 benchmark data on IOPS has grown significantly from the Luster 50-clients run to the Intel DAOS 10-clients run, with a more profound advantage for more irregular (“hard”) operations, up to more than 10x. The DAOS technology looks promising in terms of enhancing the speed of working with data, and in the near future it is planned to integrate it into the distributed heterogeneous computing environment for the MPD experiment.

## References

1. ATLAS experiment. URL: <https://atlas.cern> (Date of reference: 30.08.2021).
2. NICA – Nuclotron-based Ion Collider fAcility. URL: <https://nica.jinr.ru> (Date of reference: 30.08.2021).
3. Multi-Purpose Detector (MPD). URL: <https://nica.jinr.ru/projects/mpd.php> (Date of reference: 30.08.2021).
4. Tsaregorodtsev A. and the DIRAC Project. DIRAC Distributed Computing Services // J. Phys.: Conf. Ser. 2014. Vol. 513, No. 3. DOI: 10.1088/1742-6596/513/3/032096.
5. Heterogeneous platform “HybriLIT”. URL: <http://hlit.jinr.ru> (Date of reference: 30.08.2021).
6. Belyakov D.V., Nechaevskiy A.V., Pelevanuk I.S. et al. “Govorun” Supercomputer for JINR Tasks // CEUR Workshop Proceedings. 2020. No. 2772. P. 1-12. URL: <http://ceur-ws.org/Vol-2772/1-12-paper-1.pdf> (Date of reference: 30.08.2021).
7. Kutovskiy N., Mitsyn V., Moshkin A. et al. Integration of Distributed Heterogeneous Computing Resources for the MPD Experiment with DIRAC Interware // PEPAN. 2021. Vol. 52, No. 4.
8. RSC Group. URL: <https://rscgroup.ru> (Date of reference: 30.08.2021).
9. Pelevanyuk I. Performance Evaluation of Computing Resources with DIRAC Interware // AIP Conference Proceedings. 2021 (to be published).
10. Korenkov V., Pelevanyuk I., Tsaregorodtsev A. Integration of the JINR Hybrid Computing Resources with the DIRAC Interware for Data Intensive Applications // Data Analytics and Management in Data Intensive Domains. 2020. P. 31-46. DOI: 10.1007/978-3-030-51913-1\_3 (Date of reference: 30.08.2021).

## Automatic analysis of large-scale CT databases for fatty liver disease screening: the proof of concept

N.S. Kulberg<sup>1,2</sup>, V.A. Gombolevskiy<sup>1</sup>, A.P. Gonchar<sup>1</sup>, R.V. Reshetnikov<sup>1,3</sup>, S.S. Repin<sup>1</sup>, S.B. Prokudaylo<sup>1</sup>, V.P. Novik<sup>1</sup>, M.R. Kodenko<sup>1</sup>, M.A. Gusev<sup>1</sup>, I.D. Fateev<sup>4</sup>

<sup>1</sup>Moscow Center for Diagnostics and Telemedicine,

<sup>2</sup>Federal Research Center “Informatics and Control” of Russian Academy of Sciences,

<sup>3</sup>Sechenov First Moscow State Medical University,

<sup>4</sup>Research Computing Center, Lomonosov Moscow State University

Metabolic syndrome (MS) is a well-known risk factor for cardiovascular diseases (CVD), the leading cause of death worldwide, including the Russian Federation. Disorders associated with MS are reversible with timely detection and therapy. Identification of fatty liver disease (FLD), the essential sign of MS, is possible by analyzing the quantitative values of liver tissue density on computed tomography (CT) images. Due to the COVID-19 pandemic, we collected more than 140,000 chest CT scans, which often contain complete or partial images of the liver. The processing of these data using a supercomputer will allow collecting statistical information on the FLD signs prevalence among different cohorts and determine the risk groups for MS. This paper presents preliminary research of automatic liver density analysis on chest CT scans from the publicly available “CTLung500-Ca” dataset. We performed the analysis of the segmentation quality and estimated the liver density distribution depending on the patient sex and age. The computations were performed at the Lomonosov-2 supercomputer as a batch of processes run by Slurm cluster management and job scheduling system. The results of the research will serve as reference values for the subsequent automatic screening of CT studies from the COVID-19 database.

*Keywords:* Metabolic syndrome, fatty liver disease (FLD), steatosis, cardiovascular diseases, computed tomography (CT), tomographic liver densitometry, automatic liver segmentation, artificial intelligence, big data, opportunistic screening

### 1. Introduction

Metabolic syndrome (MS), including abdominal obesity, type 2 diabetes mellitus (T2D), arterial hypertension, and dyslipidemia, is a powerful risk factor for cardiovascular diseases, which are the leading cause of death in the Russian Federation and worldwide [1]. Disorders associated with MS are reversible if treatment is started promptly, thus preventing cardiovascular events, T2D, and disability [2]. Therefore, the early diagnosis of this condition and its manifestations is especially important.

Identification of one of the important signs of metabolic syndrome – fatty liver disease (FLD) – is possible by analyzing the numerical values of density in the computed tomography (CT) liver images [3, 4, 5]. There is no consensus on the liver density cut-off that would reliably classify patients with FLD [6]. However, as a rule, moderate-to-severe FLD is observed for the patients with liver attenuation <40 Hounsfield units (HU), constituting about 5.3% of the population [7]. In terms of the FLD early detection, chest CT can provide more valuable information than abdominal one, since the latter is usually performed for people who have already developed liver disease. Chest CT is one of the most frequently performed tomographic examinations. As a rule, it contains a complete or partial image of the liver, which provides the opportunity to assess its condition. An additional contribution was made by the COVID-19 pandemic, which resulted in unprecedented CT data accumulation for all social and age groups [8, 9]. When interpreting these CT scans, the main priority of radiologists was to look for pathologies in the lungs, without considering hepatic CT attenuation. Automatic liver segmentation and density analysis on these images will allow opportunistic screening of the MS, retrieving previously unavailable epidemiological data for the FLD.

The article deals with the initial stage of the research. It includes the study of the dependencies between demographic factors (gender, age) and liver density values using the publicly available chest



CT dataset “CTLung500-Ca”. The dataset simulates features of the large-scale database from COVID-19 collection: partial view of the liver with lung-specific scanning protocols.

## 2. Materials and Methods

To develop the concept of automatic analysis of the liver CT-density, we processed the data of an open-access chest CT dataset “CTLung500-Ca” [10, 11]. The dataset contains 536 chest CT scans of patients of Moscow outpatient clinics from the lung cancer risk group (patients age  $62.7 \pm 6.6$  years, 60% of female patients). The CT images with slice thickness greater than 1 mm were rejected.

The data set was processed at the Lomonosov-2 supercomputer [12]. The computations were performed as a batch of processes run by Slurm cluster management and job scheduling system [13]. The job of Slurm was to run a particular process from batch of planned processes, where each process computed liver density of a single CT DICOM study. Computation of a single CT DICOM study folder were done using 8 processor threads per task. Processing of a single task took about 200 sec of computing time and about 4 GBytes of RAM.

Automatic liver segmentation and CT density analysis were performed using the proprietary CTLiverExam software [14]. The processing consisted of two stages: 1. Automatic liver segmentation. 2. Automatic determining of the liver CT-density based on the of stage 1 results. The program has previously been tested for liver detection sensitivity and specificity [15, 16]. The accuracy of the automatic CT density measurement was also tested under various conditions [17, 18].

Statistical analysis of the data was performed using the dplyr [19] and pROC [20] packages for R 3.6.3 [21], the significance level was taken equal to 0.05.

## 3. Results

Automatic liver segmentation was performed, then segmentation results were assessed visually, assigning each case to one of the six categories:

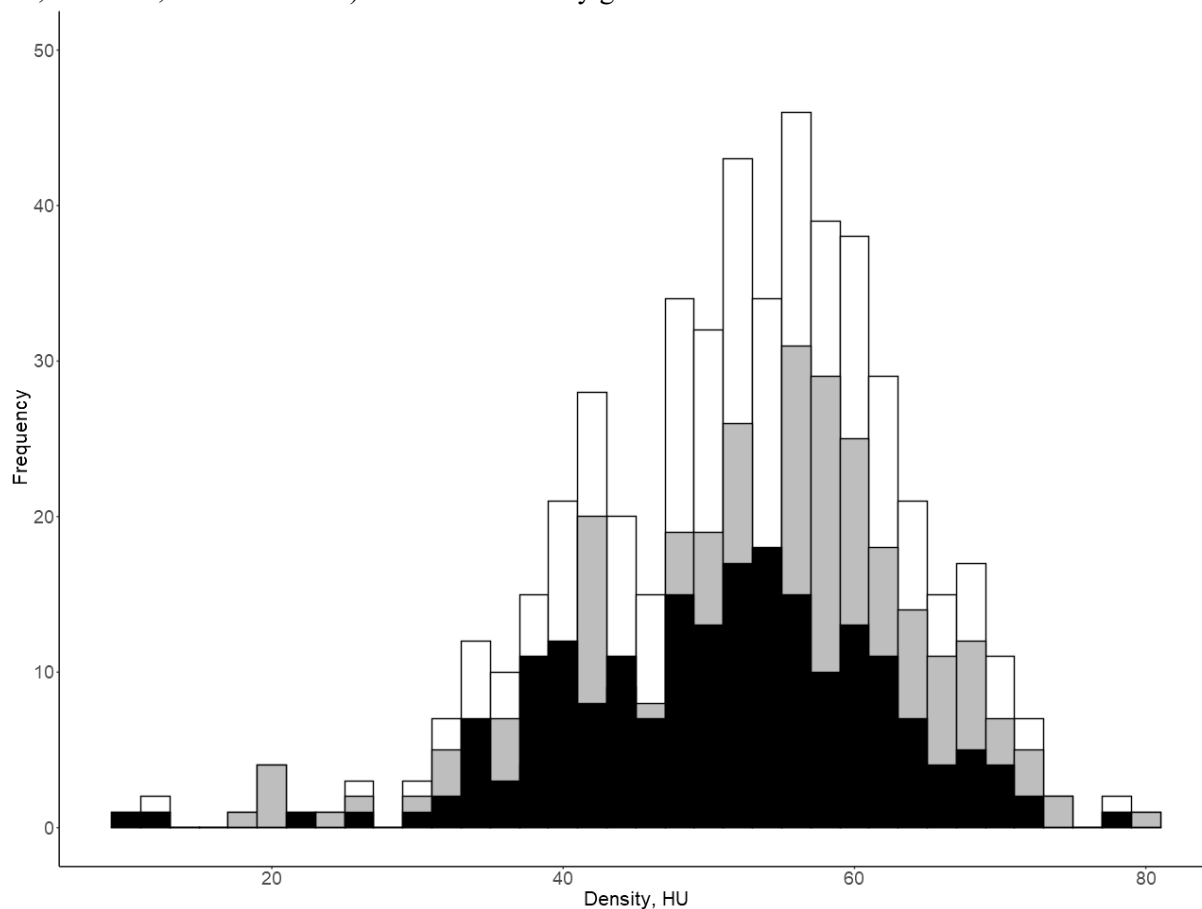
- 1) True positive (TP): the contours of the liver were determined correctly;
- 2) True negative (TN): the liver was not presented at the CT scan, and nothing was segmented;
- 3) False-positive (FP): the liver was not presented at the CT scan, but some other organs were recognized as liver;
- 4) False negative (FN): the liver was presented at the CT scan, but its contours were not determined properly;
- 5) FP+FN: the liver was presented at the CT scan, but some other organs were recognized as liver;
- 6) Not applicable (NA): CT scan is unsuitable for density analysis due to the excessive noise level in the liver area.

Based on the assessment results, 516, 11, and 9 scans were classified as TP, FP+FN, and NA, respectively. One scan in the TP category was excluded due to an unsuitable scanning protocol (slice thickness exceeded the recommended value of 1 mm).

The average liver CT density in the 515 remaining CT scans was  $52.2 \pm 11.1$  HU, which corresponds to the range of normal organ density: 40 ... 60 HU (Fig. 1), but significantly lower than in the study of the American population, for which opportunistic screening for hemochromatosis was performed ( $59.4 \pm 12.7$  HU) [22]. There was a statistically significant difference ( $p = 0.02$ ,  $t = -2.29$ ,  $df = 437.79$ ) between the mean liver density values for women ( $53.1 \pm 11.2$  HU) and men ( $50.8 \pm 10.8$  HU) (fig. 1). Liver density less than 40 HU was found in 16.4% of men (33 patients) and 11.8% of women (37 patients), but the size of the existing sample was insufficient to recognize this difference as statistically significant ( $p = 0.15$ ,  $t = 1.45$ ,  $df = 383.12$ ). Besides that, 12.9% of men (26 patients) and 10.5% of women (33 patients) had close to FLD threshold density values (liver attenuation in the range 40...45 HU). Analysis of ROC curves at the 40 HU threshold did not reveal any dependency between the patients age and liver density: the AUCs were 0.572 and 0.532 for men and women, respectively.

The dataset included 158 CTs containing no lung lesions. This provided an opportunity to study whether lung cancer is the FLD risk factor. We did not find any statistically significant difference in liver CT density between patients with lung nodules and patients without them ( $p=0.24$ ,  $t=-1.19$ ,  $df=314.28$ ).

The exceeded liver CT density is also associated with undesirable health effects such as hemochromatosis and other disorders related to liver iron overload [22]. The dataset contained 55 CT scans (10.7%, 16 males, and 39 females) with a liver density greater than 65 HU.



**Figure 1.** Liver CT density distribution in the dataset: white for all CT scans; gray for women; black for men

## 4. Discussion

Opportunistic screening has revealed a high prevalence of FLD signs among patients who provided their CT scans for the CTLung500-Ca dataset. For the research population, the prevalence was 13.6%, which is significantly higher than the same indicator (5.3%) for a sample of 170 participants in the MS-ELCAP study (Mount Sinai Early Lung and Cardiac Action Program, New York, USA). Each of the participants in the MS-ELCAP study was at risk of developing lung cancer and had a long history of smoking; the average age of patients was 62.0 years (55% females) [23]. All these indicators bring the CTLung500-Ca and MS-ELCAP populations closer together; however, the incidence of FLD signs in the Russian sample was ~2.6 times higher than in the American sample. Previously, Chen et al. found an increased incidence of FLD (16.2%) in victims of a terrorist attack on the WTC exposed to hepatotoxic substances [7]. In this context, our results suggest that the Russian population in general and the Moscow population, in particular, are subjected to risk factors that significantly increase the likelihood of developing FLD and metabolic syndrome.

In addition to FLD, the results of the research allow us to draw attention to the diseases related to iron overload (hemochromatosis and others), which are associated with increased (>65 HU) liver CT density. For this cut-off value, the proportion of patients with high liver attenuation in the CTLung500-Ca dataset (10.7%) was significantly lower than for a sample of 3357 patients from US clinics who underwent abdominal CT scan for rectal cancer screening (30.6 %) [22]. Note the different liver density distribution in the CTLung500-Ca dataset ( $52.2 \pm 11.1$  HU), in the MS-ELCAP study

( $57.6 \pm 9.3$  HU) [23], and among rectal cancer screening participants ( $59.4 \pm 9.3$  HU) [22]. This may indicate that the American and Russian populations potentially differ significantly in this parameter, which limits the possibility of direct mutual transfer of treatment recommendations and preventive measures and necessitates further research in this area.

This research is preliminary due to the limited size of the sample used. The research also has the following limitations:

- The CT densities of the liver and spleen were not compared, which may result in additional density distortions associated with the CT device calibration issues [24, 25].

- The research involved only 50+ years old patients from the lung cancer risk group, whose data may differ significantly from the population average [23].

However, these limitations will be consistently overcome at further stages of the research. We plan to screen FLD signs in the expanded CT dataset of the lung cancer risk group (about 7,000 patients). Another research direction is to check the published hypotheses on the dependency of COVID-19 and liver CT density [26]. Initially, this will be done using the open-source MosMedData dataset ([https://mosmed.ai/datasets/covid19\\_1110](https://mosmed.ai/datasets/covid19_1110)), consisting of 1,100 chest CT scans with varying degrees of lung damage [27].

The main measurements, which will allow us to obtain statistics on a wider population, will be carried out on the CT scans collected in Moscow clinics during the 2020–21 COVID-19 pandemic. More than 140,000 non-contrast chest CT scans will be automatically processed using the Lomonosov-2 supercomputer. The analysis of the liver CT density will make it possible to update the statistics on the prevalence of FLD signs among the different ages, which will allow identifying separate risk groups for MS prevention and treatment. The obtained statistics will indicate main effort directions to develop both treatment and preventive programs.

## 5. Conclusions

515 chest CT scans containing a partial liver image were automatically segmented and analyzed. A decreased density of less than 40 HU was found in 16% of men and 11% of women. A conditional risk group was identified with a density approaching the threshold value (40–45 HU), which includes another 12.9% of men and 10.5% of women. No significant relationship between hepatic density and age was detected for the studied age group 50–75 years. A significant difference in the distribution of liver density between the populations of Russia and the USA was revealed, which reasons require further research.

Based on this preliminary research, a methodology for the automatic liver CT density analysis for large samples has been developed, which will be used for gathering FLD statistics from large chest CT datasets.

## 6. Acknowledgement

The Research Computing Center of Moscow State University provided computer resources for this research. The supercomputer “Lomonosov-2” (<https://parallel.ru/cluster>) was used for automatic liver segmentation and analysis.

## 7. References

1. The GBD 2015 Obesity Collaborators. Health Effects of Overweight and Obesity in 195 Countries over 25 Years // *New England Journal of Medicine*. 2017. Vol. 377, No. 1. P. 13–27. DOI: 10.1056/nejmoa1614362
2. Neuschwander-Tetri, B.A. Non-alcoholic fatty liver disease // *BMC Medicine*. 2017. Vol. 15, No. 1. P. 45. DOI: 10.1186/s12916-017-0806-8

3. Iwasaki, M., Takada, Y., Hayashi, M., et al. Noninvasive Evaluation of Graft Steatosis in Living Donor Liver Transplantation // *Transplantation*. 2004. Vol. 78, No. 10. P. 1501–1505. DOI: 10.1097/01.tp.0000140499.23683.0d
4. Park, S.H., Kim, P.N., Kim K.W., et al. Macrovesicular Hepatic Steatosis in Living Liver Donors: Use of CT for Quantitative and Qualitative Assessment // *Radiology*. 2006. Vol. 239, No. 1. P. 105–112. DOI: 10.1148/radiol.2391050361
5. Jirapatnakul, A., Reeves, A. P., Lewis, S., et al. Automated measurement of liver attenuation to identify moderate-to-severe hepatic steatosis from chest CT scans // *European Journal of Radiology*. 2020. Vol. 122. P. 108723. DOI: 10.1016/j.ejrad.2019.108723
6. Pickhardt, P. J., Park, S. H., Hahn, L., et al. Specificity of unenhanced CT for non-invasive diagnosis of hepatic steatosis: implications for the investigation of the natural history of incidental steatosis // *European Radiology*. 2012. Vol. 22, No. 5. P. 1075–1082. DOI: 10.1007/s00330-011-2349-2
7. Chen, X., Ma, T., Yip, R., et al. Elevated prevalence of moderate-to-severe hepatic steatosis in World Trade Center General Responder Cohort in a program of CT lung screening // *Clinical Imaging*. 2020. Vol. 60, No.2. P. 237–243. DOI: 10.1016/j.clinimag.2019.12.009
8. Morozov, S., Ledikhova, N., Panina, E., et al. Re: Controversy in coronaViral Imaging and Diagnostics (COVID) // *Clinical Radiology*. 2020. Vol. 75, No. 11. P. 871–872. DOI: 10.1016/j.crad.2020.07.023
9. Morozov, S.P., Kuzmina, E.S., Ledikhova, N.V., et al. Mobilizing the academic and practical potential of diagnostic radiology during the COVID-19 pandemic in Moscow // *Digital Diagnostics*. 2020. Vol. 1, No. 1. P. 5–12. DOI: 10.17816/DD51043
10. Morozov S.P., Kulberg N.S., Gombolevskiy V.A., et al. Tagged results of lung computed tomography scans (RU 2018620500). 2018. URL: [https://mosmed.ai/en/datasets/ct\\_lungcancer\\_500/](https://mosmed.ai/en/datasets/ct_lungcancer_500/)
11. Morozov, S.P., Gombolevskiy, V.A., Elizarov, A.B., et al. A Simplified Cluster Model and a Tool Adapted for Collaborative Labeling of Lung Cancer CT Scans // *Computer Methods and Programs in Biomedicine*. 2021. Vol. 206, P. 106111. DOI: 10.1016/j.cmpb.2021.106111
12. Voevodin V.V., Antonov A.S., Nikitenko D.A., et al. Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community // *Supercomputing Frontiers and Innovations*. 2019. Vol. 6, No. 2. DOI: 10.14529/jsfi190201
13. Yoo A., Jette M., Grondona M. Slurm: Simple Linux Utility for Resource Management // *Job Scheduling Strategies for Parallel Processing in Lecture Notes in Computer Science*. 2003. Vol. 2862. P. 44-60
14. Kulberg N.S., Elizarov A.B., Kovbas V.S. Program of liver image segmentation and determining radiodensity of the liver CTLiverExam. Certificate of state registration of the computer program No. 2019660983. 2019 (In Russian).
15. Kulberg, N.S., Elizarov, A.B., Novik, V.P., et al. Automatic segmentation and determining radiodensity of the liver in a large-scale CT database // *ArXiv*. 2019. URL: <http://arxiv.org/abs/1912.13290>
16. Kulberg N.S., Elizarov A.B. et. al. Automatic batch determining radiodensity of the liver to detect subclinical liver cases // *Radiologia-Praktika*. 2020. Vol. 3, No. 81. P. 50-61 (in Russian).
17. Gonchar A.P., Elizarov A.B., Kulberg N.S., et al. Automatic Measurement of Liver Density by Computed Tomography and Ultra-Low-Dose Computed Tomography // *Novosti Khirurgii*. 2020. Vol. 28, No. 6. P. 636–647. DOI: 10.18484/2305-0047.2020.6.636
18. Gonchar A.P., Gombolevskij V.A., Elizarov A.B., et al. Liver density in routine and low-dose computed tomography: the effect of image noise on measurement accuracy // *Medical Visualization*. 2020. Vol. 24, No. 1. P. 39–47. DOI: 10.24835/1607-0763-2020-1-39-47
19. Wickham, H., François, R., Henry, L., Müller, K. dplyr: A Grammar of Data Manipulation. // *R package version 1.0.4*. 2021. URL: <https://cran.r-project.org/package=dplyr>
20. Robin, X., Turck, N., Hainard, A., et al. pROC: an open-source package for R and S+ to analyze and compare ROC curves // *BMC Bioinformatics*. 2011. Vol. 12, No. 1. P. 77. DOI: 10.1186/1471-2105-12-77
21. R Core Team. R: A Language and Environment for Statistical Computing // *R Foundation for Statistical Computing*. 2020. URL: <https://www.r-project.org/>

22. Lawrence E.M., Pooler B.D., Pickhardt P.J. Opportunistic Screening for Hereditary Hemochromatosis with Unenhanced CT: Determination of an Optimal Liver Attenuation Threshold // *American Journal of Roentgenology*. 2018. Vol. 211, No. 6. P. 1206–1211. DOI: 10.2214/AJR.18.19690
23. Chen X., Li K., Yip R., et al. Hepatic steatosis in participants in a program of low-dose CT screening for lung cancer // *European Journal of Radiology*. 2017. Vol. 94. P. 174–179. DOI: 10.1016/j.ejrad.2017.06.024
24. Gonchar A.P., Gombolevskij V.A., Elizarov A.B., et al. Possibilities of liver density estimation according to non-contrast computed tomography // *Journal of Radiology and Nuclear Medicine*. 2020. Vol. 101, No. 1. P. 58–66. DOI: 10.20862/0042-4676-2020-101-1-58-66
25. Gromov A.I., Petraikin A.V., Kulberg N.S., et al. The Problem of X-Ray Attenuation Estimation Accuracy in Multislice Computed Tomography // *Medical Visualization*. 2016. Vol. 6. P. 133–142. URL: <https://medvis.vidar.ru/jour/article/view/368/356>
26. Jothimani, D., Venugopal, R., Abedin, M. F., et al. COVID-19 and the liver // *Journal of Hepatology*. 2020. Vol. 73, No. 5. P. 1231–1240. URL: 10.1016/j.jhep.2020.06.006
27. Morozov S.P., Andreychenko A.E., Blokhin I.A., et al. MosMedData: data set of 1110 chest CT scans performed during the COVID-19 epidemic // *Digital Diagnostics*. 2020. Vol. 1, No. 1. P. 49–59. DOI: 10.17816/DD46826

# НРС TaskMaster – система мониторинга эффективности задач суперкомпьютера

П.С. Костенецкий, А.Б. Шамсутдинов, Р.А. Чулкевич, В.И. Козырев

Национальный исследовательский университет "Высшая школа экономики"

Определение эффективности использования ресурсов суперкомпьютеров является проблемой, актуальной с момента появления суперкомпьютерной отрасли. Универсальной системы для определения эффективности не существует, так как все суперкомпьютеры в своей степени уникальны за счет различных технических характеристик и программного обеспечения. Суперкомпьютер *cHARISMa* в НИУ ВШЭ также имеет свою особенность - на одном вычислительном узле могут быть запущены несколько десятков задач одновременно. Настройки позволяют более 500 пользователям суперкомпьютера НИУ ВШЭ быстро выполнять свои научные расчеты не мешая друг другу. В данной статье описывается система мониторинга собственной разработки НИУ ВШЭ для суперкомпьютера *cHARISMa* - *НРС TaskMaster*. Данная система автоматически оценивает эффективность выполнения задач пользователей суперкомпьютера и определяет неэффективные задачи, тем самым позволяя существенно экономить дорогостоящее машинное время. Пользователи могут просматривать отчеты о выполнении своих задач вместе с выводами об их работе и интерактивными графиками. Данная система построена на базе открытого программного обеспечения, что позволит в дальнейшем установить ее на других вычислительных кластерах.

*Ключевые слова:* суперкомпьютер, мониторинг, эффективность

## 1. Введение

Система мониторинга эффективности задач является важным инструментом для обнаружения некорректно запущенных вычислений, влекущих за собой недостаточно эффективное использование ресурсов суперкомпьютера. В данной работе описывается новая система мониторинга эффективности задач *НРС TaskMaster*, разработанная в Национальном исследовательском университете «Высшая школа экономики» для суперкомпьютера *cHARISMa* (Computer of HSE for Artificial Intelligence and Supercomputer Modelling). Данная система предназначена для мониторинга задач пользователей суперкомпьютера и обнаружения неэффективно используемых ресурсов.

Разработанная система позволяет пользователям просматривать отчеты о выполнении их задач вместе с интерактивными графиками выполнения, а также автоматически определять задачи, которые работали неэффективно. Имея доступ к результатам анализа своих задач, в дальнейшем пользователи могут запускать свои задачи более эффективно, что позволит существенно экономить машинное время суперкомпьютера.

Среди ошибок пользователей суперкомпьютера, наиболее распространенными являются:

- выделение недостаточного или избыточного объема ресурсов для задачи;
- запуск на CPU задач, поддерживающих вычисления на GPU;
- запуск непараллельной задачи на нескольких ядрах CPU или графических процессоров;
- выделение мощностей вычислительного узла без запуска расчетов.

Ключевой особенностью суперкомпьютера НИУ ВШЭ является то, каким образом он выделяет ресурсы для задач пользователей. Вместо выделения целого вычислительного узла для одной задачи, пользователю дается определенное количество ядер процессора и GPU, тем самым на вычислительном узле могут выполняться сразу несколько десятков

задач, оптимизируя таким образом использование ресурсов суперкомпьютера. Из-за описанной выше особенности *sHARISMa*, готовые решения для мониторинга ресурсов системы, такие как [1] и [2], не подходят для данного суперкомпьютера, и необходимо разрабатывать новую систему мониторинга, опираясь уже на имеющиеся инструменты для мониторинга в открытом доступе. В [3] описано, как использование комбинации таких программ, как Telegraf, InfluxDB, Slurm и Grafana позволяет быстро настроить и запустить систему мониторинга ресурсов суперкомпьютера. Так как все программы, за исключением Telegraf, уже установлены на *sHARISMa*, было решено использовать данный подход при разработке системы.

Помимо мониторинга суперкомпьютерных ресурсов, система должна производить анализ эффективности выполнения задач суперкомпьютера. Известной отечественной системой создания отчетов об эффективности задач является [4]. JobDigest производит анализ собранных интегральных значений и на его основе выдает задаче тег, описывающий свойство задачи (например, низкая загрузка GPU). Несмотря на то, что использование тегов удобно для поиска и фильтрации задач, не всегда теги по отдельности могут предоставить общую картину об эффективности работы задачи. Для того, чтобы однозначно определить, как эффективно работала задача на суперкомпьютере *sHARISMa*, на основе собранных тегов должен быть создан *вывод* о ее работе. Вывод также должен учитывать тип задачи при анализе, т.к. для различных научных приложений показатели эффективного использования ресурсов могут быть разными. Администратор системы мониторинга эффективности задач суперкомпьютера *sHARISMa* должен иметь возможность создавать новые выводы без помощи разработчиков, чтобы оперативно определять новые типы неэффективно запускаемых задач. По этим причинам для *sHARISMa* была разработана собственная система мониторинга эффективности задач *HPC TaskMaster*, с использованием открытого программного обеспечения [5].

## 2. Описание суперкомпьютера sHARISMa

Технические характеристики суперкомпьютера *sHARISMa* приведены в таблице 1.

**Таблица 1.** Технические характеристики суперкомпьютера *sHARISMa*

Количество узлов/CPU/ядер/GPU	40/80/1816/116
Модель процессоров	Intel Xeon Gold 6248R, 6240R, 6152
Модель GPU	NVIDIA Tesla V100 32 GB NVLink
Оперативная память кластера	34 ТБ
Параллельная система хранения данных	СХД на базе Lustre 840 ТБ
Тип вычислительной сети	InfiniBand EDR (2x100 Гбит/с)
Тип управляющей сети	Gigabit Ethernet
Пиковая производительность	1 Петафлоп/с
Производительность на тесте LINPACK	653,7 Терафлоп/с
Операционная система	Linux Centos 7.6

Высокопроизводительный вычислительный кластер *sHARISMa* занимает 6 место по производительности в списке ТОП-50 суперкомпьютеров СНГ по состоянию на 2021 год [6]. Ресурсы вычислительного кластера предназначены для поддержки проведения фундаментальных научных исследований и организации учебного процесса, а также для выполнения научных и научно-практических проектов, требующих использования суперкомпьютерных систем. На данный момент, более 500 пользователей работают на суперкомпьютере по 193 научным и учебным проектам НИУ ВШЭ. Подробные характеристики суперкомпьютера описаны в статье [7]. В 2021 планируется расширение суперкомпьютера НИУ ВШЭ новыми типами вычислительных узлов с 8 графическими картами NVIDIA Tesla A100 80GB SXM в каждом, что позволит существенно увеличить количество проводимых научных исследований мирового уровня.

### 3. Проектирование системы

Для проектирования системы мониторинга эффективности задач были определены следующие функциональные требования.

1. Система должна собирать следующие данные для каждой задачи:
  - утилизация конкретных ядер CPU, выделенных для задачи;
  - утилизация GPU, на которых выполняется задача;
  - утилизация видеопамати GPU;
  - энергопотребление GPU;
  - утилизация оперативной памяти, создаваемая задачей;
  - использование сети *InfiniBand*;
  - использование файловой системы;
2. Система должна анализировать собранные данные и на их основе определять, эффективно ли работала задача;
3. Система должны предоставлять пользователям суперкомпьютера доступ к списку выполненных задач, отчет об их выполнении и графики их выполнения при помощи веб-приложения;

Опираясь на приведенные выше функциональные требования, был проведен анализ работающих инструментов для мониторинга на *sHARISMa* и обозначены следующие нефункциональные требования для новой системы.

1. Для запуска задач на суперкомпьютере используется планировщик задач *Slurm*. Данные о задачах *Slurm* сохраняются двумя способами: основные данные записываются в реляционную базу данных *MySQL* при помощи фонового процесса *slurm database*, а метрики задач записываются в базу данных временных рядов *InfluxDB* [8] при помощи плагина *acct gather*. Данный плагин собирает такие данные, как утилизация CPU, утилизация памяти, использование сети *InfiniBand*, при этом он сохраняет метрики для каждой отдельной задачи.
2. Для сбора недостающих показателей утилизации конкретных ядер центральных процессоров и графических процессоров был установлен фоновый процесс *Telegraf* [9], который имеет встроенные плагины для сбора метрик утилизации ядер центральных и графических процессоров. Таким образом, имея ID центральных процессоров и графических процессоров, назначенных для задачи, система может собрать метрики для задействованных компонентов, вследствие чего может проанализировать эффективность выполнения задачи. Собранные метрики при помощи *Telegraf* сохраняются в БД *InfluxDB*.



3. В качестве инструмента для визуализации графиков на суперкомпьютере *cHARISMa* установлена *Grafana* [10]. *Grafana* предоставляет большие возможности для настройки и оформления графиков, а также имеет поддержку создания их при помощи API. Данный API позволяет автоматизировать создание графиков для каждой задачи.

Преимуществом использования комбинации *Telegraf*, *InfluxDB* и *Grafana* является возможность установки и настройки данных инструментов на любом суперкомпьютере. Помимо этого, данные инструменты делают систему мониторинга достаточно гибкой. При необходимости, можно собирать дополнительные данные, используя встроенные плагины *Telegraf* или разработав свои.

Диаграмма компонентов системы *HPC TaskMaster* представлена на рис. 1.

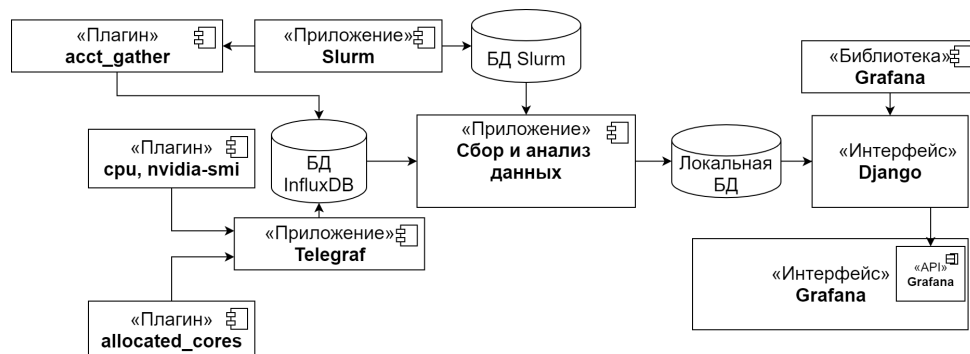


Рис. 1. Диаграмма компонентов системы

## 4. Алгоритм работы системы

В данном разделе приведены основные алгоритмы работы системы *HPC TaskMaster*.

### 4.1. Алгоритм сбора и анализа статистики системы

Алгоритм работы системы мониторинга эффективности задач представлен на рис. 2.

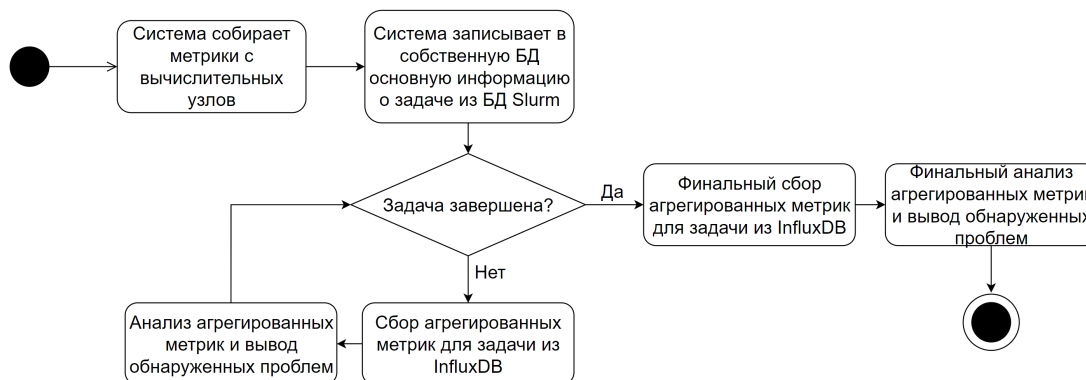


Рис. 2. Алгоритм работы системы мониторинга эффективности задач

Система мониторинга эффективности задач работает по следующему принципу:

- на каждом вычислительном узле собираются метрики при помощи *Telegraf* и сохраняются в БД *InfluxDB* на головном узле. Также, в *InfluxDB* загружаются метрики из плагина *acct gather*;
- система актуализирует свою локальную базу данных *MySQL*, сравнивая ее задачи с задачами из базы данных *Slurm*;

- пока задача находится в состоянии работы, для нее с определенным периодом собираются агрегированные метрики из БД InfluxDB;
- собранные агрегированные метрики анализируются системой и для задачи создается список обнаруженных проблем;

На основе имеющихся данных о задаче, при запросе пользователя для нее автоматически строятся графики в Grafana.

## 4.2. Алгоритм работы автоматической генерации графиков в Grafana

Для автоматического создания графиков для каждой задачи в *Grafana*, был разработан специальный алгоритм, представленный на рис. 3.

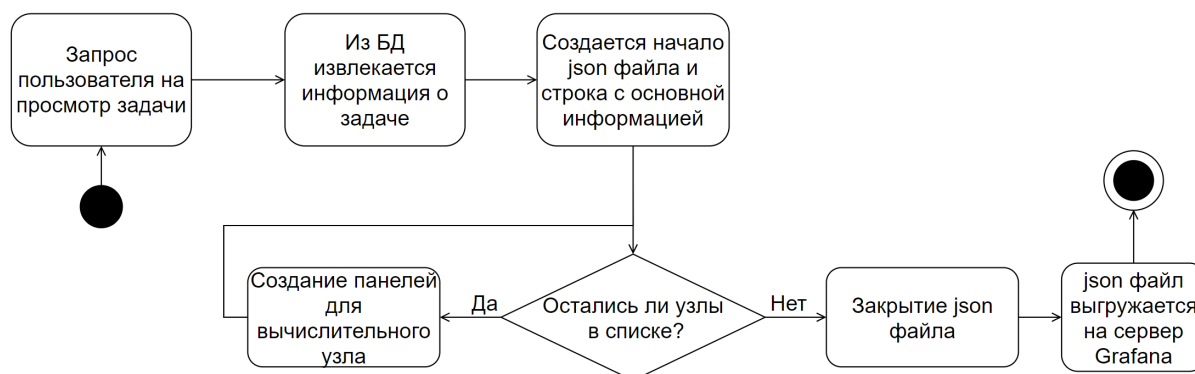


Рис. 3. Алгоритм работы автоматической генерации графиков

В дальнейшем, созданные графики используются для отображения на сайте системы при помощи технологии *iframe*, где пользователь может в интерактивном режиме ознакомиться с графиками, просматривая показатели за определенный промежуток времени. Также, система создает графики как для выполненной, так и для работающей задачи, тем самым пользователь имеет возможность наблюдать работу своей задачи в реальном времени. Пример автоматически созданных графиков для задачи приведен на рис. 4.



Рис. 4. Графики утилизации вычислительных ресурсов задач

## 5. Обнаружение неэффективных задач при помощи системы

Чтобы определить, работает ли задача неэффективно, необходимо оценить утилизацию задействованных задач компонент. Для этого вводится понятие *индикатор проблем* - если значение утилизации компонента попадает в границы утилизации, заданные администратором, то в отчете появляется индикатор проблемы утилизации данного компонента, у которого есть уровень проявления от 0 до 1 (от наименьшего к наибольшему). Примером индикатора низкой утилизацией ядра процессора может быть значение утилизации ядра равное 20%, в то время как минимальная граница, заданная администратором равна 0%, а максимальная - 30%. В этом случае уровень проявления индикатора будет равен 0.33.

На данный момент, реализовано распознавание следующих типов индикаторов проблем:

- низкая утилизация как минимум одного из ядер центрального процессора или графического процессора во время выполнения задачи;
- низкая утилизация ядер центрального процессора при высокой утилизации графических процессоров;
- низкая утилизация графических процессоров при высокой утилизации ядер центрального процессора;
- низкое использование сети *InfiniBand* при использовании нескольких узлов.

На выходе система получает список индикаторов проблем для данной задачи. Следующим шагом является генерация вывода о работе задачи на основе имеющихся индикаторов. В настройках системы администратор указывает параметры, на основе которых генерируется вывод - среди всех типов индикаторов, администратор выбирает нужные для определенного вывода и задает границы уровня проявления проблем, а также указывает длительность и имя приложения задачи. Если при анализе задачи все обнаруженные индикаторы совпали по уровню проявления проблем с параметрами вывода, то данный вывод применяется к задаче. У каждого вывода есть свой приоритет, наивысший отдается выводам для определенных приложений.

Алгоритм обнаружения неэффективных задач приведен на рис. 5.



Рис. 5. Алгоритм обнаружения неэффективных задач

Пример отчета о выполнении задачи с созданным выводом приведен на рис. 6.


Общая информация:		Индикаторы проблем:	
ID задачи:	358910	<b>Индикаторы:</b>  <ul style="list-style-type: none"> <li> CPU: низкая средняя загрузка                             <ul style="list-style-type: none"> <li>node cn-002: 0.38% (значение уровня: 0.99)</li> </ul> </li> <li> CPU: низкая загрузка отдельных ядер                             <ul style="list-style-type: none"> <li>Node: cn-002</li> </ul> </li> <li> GPU: низкая средняя загрузка                             <ul style="list-style-type: none"> <li>Node: cn-002                                     <ul style="list-style-type: none"> <li>gru-0: 1.66% (значение уровня: 0.92)</li> </ul> </li> </ul> </li> </ul> <b>Вывод:</b> <ul style="list-style-type: none"> <li> Неэффективное использование Jupyter Notebook                             <ul style="list-style-type: none"> <li>Приложение Jupyter Notebook работает неэффективно!</li> </ul> </li> </ul>	
Имя задачи:	jupyter-notebook		
Состояние задачи:	таймаут		
Пользователь:			
Дата старта:	8 сентября 2021г. 16:16:53		
Дата завершения:	9 сентября 2021г. 16:17:01		
Длительность:	1д 0ч 00м 00с		
Количество узлов:	1		
Количество ядер CPU:	32		
Количество GPU:	1		
Выделенные узлы и их типы:	type_a: cn-[002]		
Команда запуска:	/usr/local/bin/run_notebook		

Рис. 6. Фрагмент отчета о выполнении задачи

## 6. Заключение

Разработанная система мониторинга эффективности задач *HPC TaskMaster* позволяет пользователям суперкомпьютера просматривать информацию о своих задачах, а также обнаруживает случаи, когда выделенные задачей ресурсы были использованы неэффективно. Благодаря использованию в качестве подсистем открытого программного обеспечения, система отличается гибкостью, и в дальнейшем может быть внедрена на различные суперкомпьютеры.

В качестве дальнейшего развития системы планируется: 1) расширить список отслеживаемых проблем в задачах пользователей, 2) внедрить анализ временных рядов при помощи математических методов или путем использования нейронных сетей, 3) подготовить систему для публикации в открытом репозитории.

Исследование выполнено с использованием суперкомпьютерного комплекса *sHARISMa* НИУ ВШЭ [7].

## Литература

1. Nagios - The Industry Standard In IT Infrastructure Monitoring. (n.d.). Retrieved March 24, 2021, from <https://www.nagios.org/>
2. Zabbix :: Open source решение распределенного мониторинга корпоративного класса. (n.d.). Retrieved March 24, 2021, from <https://www.zabbix.com/ru>
3. Chan N. A resource utilization analytics platform using grafana and telegraf for the Savio supercluster // ACM International Conference Proceeding Series. Association for Computing Machinery, 2019.
4. Nikitenko D. и др. JobDigest – Detailed System Monitoring-Based Supercomputer Application Behavior Analysis // Communications in Computer and Information Science. Springer Verlag, 2017. Т. 793. С. 516–529.
5. Шамсутдинов А.Б., Костенецкий П.С. Разработка системы мониторинга эффективности задач на суперкомпьютере sHARISMa // Параллельные вычислительные технологии ПаВТ'2021, г. Волгоград.
6. Top50 | Суперкомпьютеры. (n.d.). Retrieved September 2, 2021, from <http://top50.supercomputers.ru/list>

7. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // Journal of Physics: Conference Series. 2021. Т. 1740, № 1.
8. Fadhel M., Sekerinski E., Yao S. A comparison of time series databases for storing water quality data // Advances in Intelligent Systems and Computing. 2019. Т. 909.
9. Rattanatamrong P. и др. Overhead study of telegraf as a real-time monitoring agent // JCSSE 2020 - 17th International Joint Conference on Computer Science and Software Engineering. Institute of Electrical and Electronics Engineers Inc., 2020. С. 42–46.
10. Beermann T. и др. Implementation of ATLAS Distributed Computing monitoring dashboards using InfluxDB and Grafana // EPJ Web Conf. EDP Sciences, 2020. Т. 245. С. 03031.

# Improving SL-AV Global Atmosphere Model Computational Efficiency with I/O and Algorithmic Optimizations\*

Mikhail Tolstykh, Rostislav Fadeev, Gordey Goyman, and Vladimir Shashkin

<sup>1</sup> Marchuk Institute of Numerical Mathematics Russian Academy of Sciences, Moscow, Russia

<sup>2</sup>Hydrometcentre of Russia, Moscow, Russia

<sup>3</sup>Moscow Institute of Physics and Technology, Dolgoprudny, Russia

Numerical weather prediction is a time-critical application routinely using many thousands of processor cores. In this paper, the results of further optimization of SL-AV global numerical weather prediction model having the resolution of about 11 km at a massively parallel Cray XC40 computer are presented. We concentrate here mostly on I/O optimizations having the goal to achieve the elapsed time of 24-hours forecast of less than 20 minutes at approximately 3000 processor cores.

**Keywords:** Numerical weather prediction, Global atmosphere models, Massively parallel computations, Parallel I/O

## 1. Introduction

Numerical weather prediction is a time-critical application that routinely uses many thousands of processor cores. Many computer systems of the world weather forecasting centers are in supercomputer Top500 list [1]. The elapsed time of a global numerical medium-range weather forecast crucially depends on the parallel efficiency of the numerical algorithm and its implementation.

SL-AV is Russian global atmosphere model developed at Marchuk Institute of Numerical mathematics Russian Academy of Sciences and Hydrometcentre of Russia [2]. The model is based on the atmospheric equations in hydrostatic approximations. The distinct feature of this model is the use of absolute vorticity equation as one of prognostic variables. This model is applied for operational medium-range and long-range weather forecasts. Algorithms and their parallel implementation are described in [3]. One-dimensional MPI decomposition and OpenMP loop parallelization is applied in the SL-AV code written in Fortran language [4]. The code demonstrated 53% efficiency at 9072 processor cores for  $3024 \times 1513 \times 126$  grid [5]. We have then developed a new version of this model with horizontal resolution of about 10 km and 104 vertical levels. We further improved a lot the parallel implementation of the SL-AV model [6, 7], including a partial use of the single precision instead of double, OpenMP vector length optimizations. As a result, the elapsed time of a 1-day forecast of the model with the abovementioned resolution without I/O has reached 32 minutes at 4000 processor cores of Cray XC40 [6] installed at Roshydromet Main Computing Center.

However, this is not sufficient for operational application of this version of the SL-AV model. Moreover, it is quite difficult to carry out complex tuning of all model parameterizations for subgrid-scale processes that are mostly resolution-dependent. Such a tuning requires multiple numerical experiments involving a series of forecasts for different seasons and so is quite time consuming.

We concentrate here mostly on I/O optimizations having the goal to achieve the elapsed time of 24-hours forecast of less than 20 minutes at approximately 3000 processor cores. Some algorithmic improvements are also briefly considered.

---

\* The study, except for Subsection 3.3, was supported by the Russian Science Foundation (project 21-71-30023).

## 2. Organization of I/O in the Atmosphere and Earth System Models

For a modern global model with 7-10 km horizontal resolution the I/O efficiency became an issue. Indeed, the size of the problem is  $10^9$ . From 10 to 12 3D variables and about 20 2D variables need to be stored in a file with forecast initial conditions so the typical size of the initial data file is about 26 GB. 3D variables in a modern atmosphere model include wind speed components, temperature, specific humidity, 4-5 hydrometeors (i.e. rain droplets and ice particles concentrations), ozone concentration, turbulent kinetic energy. Then the forecast information with a size of 3 GB needs to be stored every 1-3 hours depending on forecast lead-time. The forecast information usually consists of five 3D fields defined at isobaric surfaces (geopotential, temperature, wind speed components, relative humidity) and 2D fields (precipitation, near surface temperature, wind components, relative humidity, snow depth etc).

It is known since long ago that the implementation of parallel input-output of data in an atmospheric model can significantly accelerate its execution, see, for example [8]. Gradual establishment of MPI-IO moved the focus towards interfaces convenient for atmosphere and Earth system models. So the incorporation of parallel capabilities based on MPI-IO into NetCDF freeware library commonly used in Earth system models and its model components [9] was natural. NetCDF file contains metadata information making it portable and searchable. This format is supported by many software packages including plotting software.

While NetCDF format is widely used in climate modeling [10], historically, the GRIB (and GRIB2) formats [11] are generally accepted in the numerical weather prediction community. This format allows to significantly compress data thus reducing the file size. Typically, a file in GRIB2 format is 2-3 times smaller than the file with the same information written in NetCDF4 format without compression. Unfortunately, the compression algorithm used in GRIB is essentially sequential. At the same time, recent NetCDF libraries (starting from version 4.7.4) include parallel compression [12] so the file size becomes comparable to GRIB2 file size.

There are advanced parallel I/O systems based on NetCDF format applied for many Earth system models coupling many component models (atmosphere, ocean, sea ice, etc) [13, 14].

## 3. The Model Improvements

### 3.1. Brief Description of the Model Configuration

The work described in this paper concerns the recent version of SL-AV model with the horizontal resolution  $0.1^\circ$  in longitude, variable resolution in latitude between 7 km in Northern hemisphere to 13 km in Southern hemisphere and 104 vertical levels (SLAV10). The grid dimensions are  $3600 \times 1946 \times 104$ . The implementation improvements for this version presented in [6] have reduced the elapsed time from 42 to 31 minutes at 3880 processor cores of Cray XC40. However, this timing does not include the time necessary to read initial data and write forecast data. Moreover, operational environment requires to reduce the amount of processor cores from 3880 to 2916. This means that further optimizations are needed to bring the elapsed time of 24-hour forecast below 20 minutes, the limit dictated by operational application.

### 3.2 Implementation of I/O

For a modern global model with 7-10 km horizontal resolution the I/O efficiency became an issue. Indeed, the size of the problem is  $10^9$ . From 10 to 12 3D variables and about 20 2D variables need to be stored in a file with forecast initial conditions so the typical size of the initial data file is about 26 GB. 3D variables in a modern atmosphere model include wind speed components, temperature, specific humidity, 4-5 hydrometeors (i.e. rain droplets and ice particles concentrations), ozone

concentration, turbulent kinetic energy. Then the forecast information with a size of 3 GB needs to be stored every 1-3 hours depending on forecast lead-time. The forecast information usually consists of five 3D fields defined at isobaric surfaces (geopotential, temperature, wind speed components, relative humidity) and 2D fields (precipitation, near surface temperature, wind components, relative humidity, snow depth etc).

It is known since long ago that the implementation of parallel input-output of data in an atmospheric model can significantly accelerate its execution, see, for example [8]. Gradual establishment of MPI-IO moved the focus towards interfaces convenient for atmosphere and Earth system models. So the incorporation of parallel capabilities based on MPI-IO into NetCDF freeware library commonly used in Earth system models and its model components [9] was natural. NetCDF file contains metadata information making it portable and searchable. This format is supported by many software packages including plotting software.

While NetCDF format is widely used in climate modeling [10], historically, the GRIB (and GRIB2) formats [11] are generally accepted in the numerical weather prediction community. This format allows to significantly compress data thus reducing the file size. Typically, a file in GRIB2 format is 2-3 times smaller than the file with the same information written in NetCDF4 format without compression. Unfortunately, the compression algorithm used in GRIB is essentially sequential. At the same time, recent NetCDF libraries (starting from version 4.7.4) include parallel compression [12] so the file size becomes comparable to GRIB2 file size.

There are advanced parallel I/O systems based on NetCDF format applied for many Earth system models coupling many component models (atmosphere, ocean, sea ice, etc) [13, 14].

### 3.3. Algorithmic Optimizations

Some changes in numerical algorithm of the SL-AV model dynamical core were introduced. The major change was moving the computations of the horizontal diffusion of divergence after the temperature computation where the divergence is used. This algorithmic change has eliminated a numerical noise in some prognostic fields (surface pressure and temperature) visible near steep mountains. This, in turn, allowed to increase the time step of the model from 135 to 300 s without degrading the forecast quality. The reduction in number of time steps necessary for computation of 24-hours forecast, in turn, resulted in respective reduction of elapsed time.

## 4. Numerical Experiments

We run SL-AV model at Cray XC40 system installed at Roshydromet's Main Computing Center. The system consists of 936 nodes with two Intel Xeon E2697v4 18-core CPUs and 128 GB memory. All the nodes are connected with Cray ARIES interconnect. The peak performance is 1.29 PFlops. The system includes Lustre parallel file system.

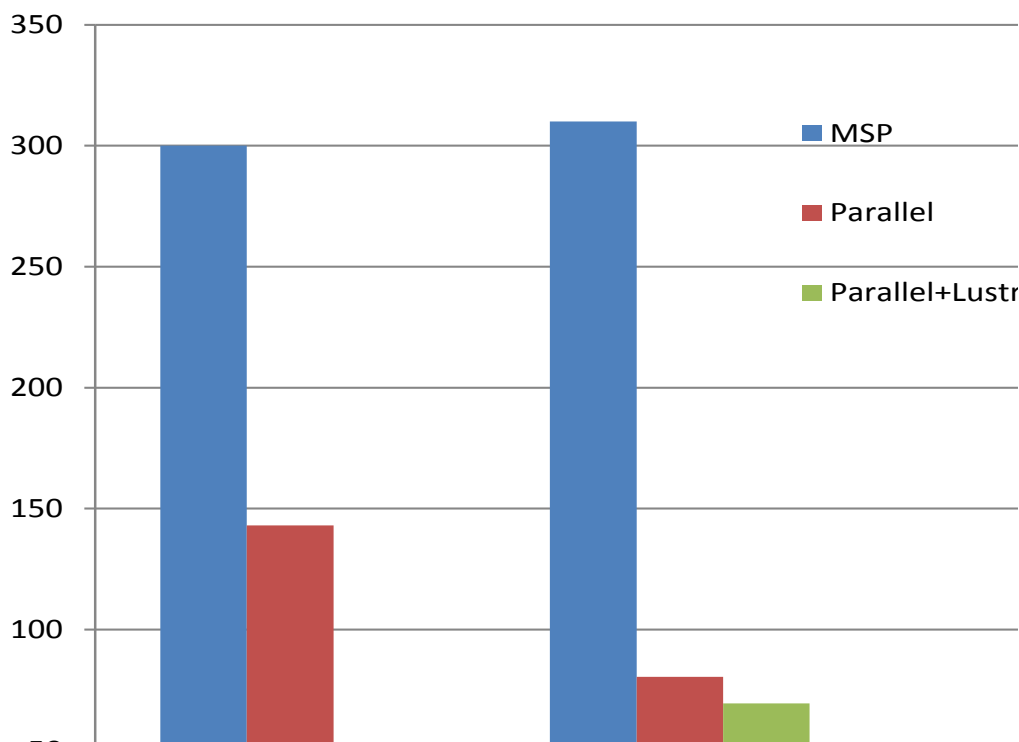
Initially, in 2019 the elapsed time necessary to compute the 24-hours forecast with the SL-AV model with the resolution mentioned above using about 4000 processor was 42 minutes without time for I/O. The efforts undertaken in 2020 have reduced this time to less than 31 min [6]. As mentioned above, this was still too much as the operational requirements dictate the limitation of no more than 20 minutes. The number of processor cores had to be reduced as well. Now we use in our experiments 81 nodes each having 36 processor cores. Each node runs 6 MPI processes, each of them running 6 OpenMP threads. This use of a Cray XC40 node was found optimal in earlier works. Thus, 2916 processor cores are used. The elapsed time for different I/O tasks in the SL-AV model using sequential and parallel I/O is shown in Fig.1. The results of using Lustre file system capabilities for accelerating parallel I/O are also shown there.

One can see that using parallel I/O significantly accelerates this part of the code. Further acceleration is achieved while using Lustre file system options. For example, the procedure of reading the file



with initial data is accelerated by a factor of 2.1 for parallel I/O alone and by factor of 10 if this file is physically located at different hard disk drives as set by `lfs setstripe` command.

Now the breakdown between different I/O components is as follows. Reading initial data at the beginning of the forecasts takes approximately 30 seconds. 70 seconds is required to write a file similar to the file with initial data which is used as a first guess file at the next forecast cycle (6-hour forecast), and just 14 seconds is needed to write postprocessed data at pressure levels for the forecasts at all other lead times. This can be compared with the elapsed time of the usual model time step that takes 3.4 s for the time steps without radiation computations and 5.8 s for the time step including radiation calculations (this is every eighth step). Given that the output of forecasted fields at pressure levels is required every three hours, the I/O elapsed time per forecast day is reduced from 440 sec to 120 sec for all forecast days other than the first one. The similar numbers for the first forecast day are 715 and 182 sec, respectively.



**Fig.1.** Elapsed time in seconds for used in different I/O steps of SL-AV model code while using 2916 cores at Cray XC40. ML means writing the information at model levels; the content mostly coincides with the initial data file. PL means writing prognostic information at pressure levels. MSP means sequential I/O at the master process with gather/scatter data from/to all other processes.

Algorithmic changes have allowed to reduce the number of time steps for 24-hours forecasts from 640 to 288 with the time step increased from 135 to 300 s. This means that the wall-clock time of computational part for 24-hour forecast (except for the first forecast day) has shrunk from 2370 to 1070 sec. Now the elapsed time of full 24-hour forecast is about 20 min at 2916 processor cores.

## 5. Conclusions

In this short paper, we have presented the works on further optimization of the potential future operational Russian medium-range weather forecast model, SLAV10, having the horizontal resolution of about 10 km, and 104 vertical levels. We have implemented parallel I/O based on parallel NetCDF

standard library routines. This resulted in 10 minutes saving in the elapsed time of the full SL-AV forecast. We have also incorporated some algorithmic improvements into the model allowing to increase the time step of the model. As of the end of 2020, the wall-clock time of 24-hours forecast was 31 min at 3888 processor cores at Cray XC40 system without I/O. Now the wall-clock time (including I/O) to complete 24-hour forecast time at 2916 processor cores is 20 min. This time allows operational application of this version of the SL-AV model and makes it easier to work on its complex tuning.

One can expect a similar acceleration of SL-AV code running at other computer systems as free-ware parallel NetCDF library is supported on most known platforms.

For this reason, I/O optimizations described in this paper can be also implemented for a wide range of applications.

Our further optimizations plans include experiments with single precision calculations in other parts of the SLAV10 model.

**Acknowledgements.** The SL-AV model has been developed in a collaboration between Marchuk Institute of Numerical Mathematics RAS (INM) and Hydrometcentre of Russia. The authors are grateful to Vassily Mizyak, Radomir Zaripov, Svetlana Travova, Vladimir Rogutov from Hydrometcentre who participate in model development and research. Design, debugging and preliminary tests of the parallel I/O and SL-AV dynamical core modifications were carried out using cluster installed at INM RAS. The numerical experiments with the SL-AV model having 0.1° resolution are carried out using Cray XC40 installed at the Roshydromet Main Computer Center (MCC). This study was performed at Marchuk Institute of Numerical Mathematics RAS.

## References

1. TOP500 Supercomputer sites. <https://www.top500.org/>
2. Tolstykh, M.A., Fadeev, R.Y., Shashkin, V.V., Goyman G.S., Zaripov, R.B., Kikteev, D.B., Makhnorylova, S.V., Mizyak, V.G., Rogutov, V.S.: Multiscale Global Atmosphere Model SL-AV: the Results of Medium-range Weather Forecasts. *Russ. Meteorol. Hydrol.* **43**, 773–779 (2018). <https://doi.org/10.3103/S1068373918110080>
3. Tolstykh, M., Goyman, G., Fadeev, R., Shashkin, V. Structure and algorithms of SL-AV atmosphere model parallel program complex. *Lobachevskii Journal of Mathematics.* **39** 587-595 (2018).
4. Tolstykh, M., Shashkin, V., Fadeev, R., Goyman, G.: Vorticity-divergence semi-Lagrangian global atmospheric model SL-AV20: dynamical core. *Geosci. Model Dev.* **10**, 1961-1983 (2017), doi:10.5194/gmd-10-1961-2017
5. Tolstykh, M., Goyman, G., Fadeev, R., Shashkin, V., Lubov, S.: SL-AV Model: Numerical Weather Prediction at Extra-Massively Parallel Supercomputer. In: Voevodin, V., Sobolev, S. (eds.) *RuSCDays 2018*. CCIS, vol. 965, pp. 379-387. Springer, Cham (2019). doi: 10.1007/978-3-030-05807-4\_32
6. Tolstykh, M., Goyman, G., Fadeev, R., Shashkin, V. Implementation of SL-AV Global Atmosphere Model with 10 km Horizontal Resolution. In: Voevodin, V., Sobolev, S. (eds.) *RuSCDays 2020*. CCIS, vol. 1331, pp. 216-225. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-64616-5\\_19](https://doi.org/10.1007/978-3-030-64616-5_19)
7. Tolstykh, M., Goyman, G., Fadeev, R., Travova, S., Shashkin, V. Development of the global multiscale atmosphere model: computational aspects. *J. Phys.: Conf. Ser.* **1740**, 012074 (2021). doi:10.1088/1742-6596/1740/1/012074

8. Parallelization of HIRLAM Model Parallelization and Asynchronous I/O  
<https://www.ecmwf.int/sites/default/files/elibrary/2004/14140-parallelization-hirlam-model-parallelization-and-asynchronous-io.pdf>
9. <https://www.unidata.ucar.edu/software/netcdf/>
10. Dennis, J. M., Edwards, J., Loy, R., Jacob, R., Mirin, A. A., Craig, A. P., Vertenstein, M. An application-level parallel I/O library for Earth system models. *The International Journal of High Performance Computing Applications*, **26**(1), 43–53 (2012).  
<https://doi.org/10.1177/1094342011428143>
11. Manual on Codes - International Codes, Volume I.2, Annex II to the WMO Technical Regulations: Part B – Binary Codes, Part C – Common Features to Binary and Alphanumeric Codes. ISBN 978-92-63-10306-2 [https://library.wmo.int/doc\\_num.php?explnum\\_id=10310](https://library.wmo.int/doc_num.php?explnum_id=10310)
12. Learning HDF5. <https://portal.hdfgroup.org/display/HDF5/Learning+HDF5>
13. Yang, R., Ward, M., Evans, B.: Parallel I/O in Flexible Modelling System (FMS) and Modular Ocean Model 5 (MOM5), *Geosci. Model Dev.* **13**, 1885–1902 (2020),  
<https://doi.org/10.5194/gmd-13-1885-2020>.
14. XIOS <https://portal.enes.org/models/software-tools/xios>

# Optimization and regularization of the inverse problem for stochastic differential equation using graphics accelerators \*

O.I. Krivorotko<sup>1,2</sup>, A.V. Neverov<sup>2</sup>, T. Hohage<sup>3</sup>

<sup>1</sup>Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch of the Russian Academy of Sciences

<sup>2</sup>Novosibirsk State University

<sup>3</sup>Institute for Numerical and Applied Mathematics

The problem of drift and volatility parameters identification in stochastic differential equations (SDEs) using additional measurement of single trajectory of stochastic process is investigated. The classical way for solving such a problem is to reduce it to a Fokker-Planck equation and minimize a suitable data fidelity functional, what is carried out sequentially. In addition to that, such inverse problems are ill-posed, i.e. their solutions are unstable. For higher-dimensional systems of SDEs, the numerical solution of the Fokker-Planck equations becomes infeasible since the solution depends on  $n + 1$  variables for drift coefficient and  $(n + 1)^2$  for covariance matrix, and computational complexity is  $O(n^3)$ . We propose regularized Landweber iteration algorithm for easier paralleling of problem. The key idea is introduction of solution-dependant parameters, what allows us to introduce implicit time dependency. The benefit of such approach consists in that the adjoint problem is deterministic, gradient of fidelity functional has the integral form and consists of mathematical expectations, that allow us to effectively parallelize algorithm with Monte-Carlo approach. Moreover, dependency is implemented with Fourier series that helps to reduce number of variables and compute gradient independently for each basis function. The algorithm determines the stable space of parameters. We conduct this process on synthetic data for validation of an algorithm and regularization for variety of input data.

*Keywords:* inverse problem, stochastic differential equation, Landweber iteration, regularization, optimization, stability analysis, parallelization.

## 1. Introduction

The recent advances in computing technologies, supercomputer modeling, cloud computing make it possible to analyze more and more complex and realistic models in economy, engineering, and the natural sciences. Real economic, social or biological systems will always be exposed to external influences that are not completely understandable or inaccessible to explicit models (hormonal fluctuations, variations in blood pressure, respiration, variable neuronal control of muscle activity, enzymatic processes, energy needs, cellular metabolism, smoking, stress and etc.), and therefore there is a growing need to expand deterministic models to models that encompass more complex variations in dynamics. A natural continuation of models of deterministic differential equations are systems of stochastic differential equations (SDEs), where the corresponding parameters are modeled as suitable random processes, or stochastic processes are added to the equations of the motion system

$$d\mathbf{X}_t = \mu(t, \mathbf{X}_t)dt + \sigma(\mathbf{X}_t)d\mathbf{W}_t, \quad \mathbf{X}_0 = x_0, \quad t \in [0, T]. \quad (1)$$

Here  $\mathbf{X}_t = \mathbf{X}(t)$  is a family of random variables with values in  $\mathbb{R}^n$ ,  $\mathbf{W}_t = \mathbf{W}(t)$  is a Wiener random process described the Brownian motion,  $\mu : [0, T] \times \mathbb{R}^n \rightarrow \mathbb{R}^n$  is a drift coefficient and  $\sigma : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$  is a volatility or diffusion. That coefficients characterize the transfer rates, rates of consumption and discounting in economics, population specific and disease propagation in

---

\*Supported by the Russian Science Foundation (project no. 18-71-10044).

biological applications. There is an interest to estimate the drift or the diffusion either non-parametrically or parametrically to gain a better understanding of the modeled process using observation of a single trajectory:

$$\mathbf{X}_t = f_t, \quad t \in (0, T). \quad (2)$$

The variable  $\mathbf{X}_t$  has a probability density  $\rho(\cdot)$ , which solves the stationary Fokker-Planck equation

$$\operatorname{div} \left( -\mu\rho + \frac{1}{2}\sigma\sigma^T \operatorname{grad} \rho \right) = 0. \quad (3)$$

Suppose that  $\sigma$  is known and  $\mu$  should be estimated. Then we can define the deterministic coefficient-to-solution nonlinear operator

$$G(\mu) = \rho(\cdot). \quad (4)$$

The proposed project focuses on the refinement of the model coefficients (functions  $\mu$  and  $\sigma$ ) from (1) by some additional information (2) about system states (the inverse problem). One way to solve the problem of identification of the coefficients in (1)–(2) is to reduce the inverse problem for the SDEs (1) to an inverse problem (4) for the Fokker-Planck equation (3) [18].

A natural approach to estimate  $\mu$  is then to minimize a suitable data fidelity functional

$$J(\mu) := \mathcal{S}(f_t, G(\mu)) \quad (5)$$

measuring the "closeness" of  $G(\mu)$  to the observed data  $f_t$ .

It can be shown that these inverse problems are ill-posed, i.e. their solution is unstable. This means that a simple maximum likelihood estimator, i.e. a minimizer of  $\mathcal{S}(f_t, G(\mu))$  over  $\mu$  in some convex set, is unstable. Therefore, regularization algorithms should be applied [13, 15].

Secondly, for the minimization of fidelity functional (5) iteration approaches are implemented that depend on solution of SDEs problem (1) on each iteration. For higher-dimensional systems of SDEs the numerical solution  $\rho(x)$  of the Fokker-Planck equations becomes infeasible since the solution  $\rho$  depends on  $n + 1$  variables. However, it is still possible to solve the SDE (1) many times to sample from  $\rho$ . Corresponding algorithms can be parallelized with the number of trajectories of the SDE and possibly also with the number of dimensions [1, 16]. A challenge that we will address in this paper is to design regularization methods that can deal with such an approximate random evaluation of the forward operator.

The paper is organized as follows. Section 2 demonstrates the short historical review in regularization of inverse problems for SDEs. The variation problem statement as well as Landweber iteration algorithm for solving inverse problem is formulated in Section 3. The numerical results of solving the inverse problem for synthetic data, its parallelization and regularization are demonstrated in Section 4.

## 2. Short historical review

The modeling by stochastic differential equations became standard in financial econometrics since the work of F. Black and M.S. Scholes [3]. More recent works on nonparametric estimation of the drift are those by M. Hoffmann [10] using wavelets, V.G. Spokoiny [20] using kernel methods, F. Comte, V. Genon-Catalot and Y. Rozenholc [6] using penalized least squares, E. Schmisser [19] applying penalized least squares to high dimensional problems, O. Paspiliopoulos et al. [17] using Bayesian methods. A parametric estimator was developed by A. Hurn, J. Jeismann and K. Lindsay [12]. They propose a maximum likelihood estimator which relies on the numerical solution of Fokker-Planck equation by finite elements. Due to a parametric model for  $\mu$  their problem is not ill-posed. T. Hohage and F. Dunker [7] derived convergence

rates for estimating parameters in SDE by variational regularization methods. However, in each situation the assumptions of the convergence theorems have to be checked, which may not always be an easy task. The iteratively regularized Gauss-Newton method with quadratic penalty and quadratic data fidelity term was suggested by A.B. Bakushinskii [2], and further analyzed by B. Blaschke, A. Neubauer and O. Scherzer [4] and T. Hohage [11] for low order Hoelder or logarithmic source conditions, respectively. Further references can be found in the monograph of B. Kaltenbacher, A. Neubauer and O. Scherzer [15]. Regularization with general convex penalty terms has recently been investigated in a number of papers [5, 8].

### 3. Methods and approaches

Our approach for solving inverse problems for SDEs involves approximation of parameters with Fourier series

$$\mu \approx \bar{\mu} = \sum_{m=1}^M q_{m\mu} \varphi_m(x), \quad \sigma \approx \bar{\sigma} = \sum_{m=1}^M q_{m\sigma} \varphi_m(x). \quad (6)$$

Here  $q = (q_{0\mu}, q_{1\mu}, \dots, q_{M\mu}, q_{0\sigma}, \dots, q_{M\sigma}) \in \mathbb{R}^{2M+2}$ ,  $\{\varphi_m\}_{m \in \mathbb{N}}$  is an orthonormal basis in  $L^2$ .

We consider forward operator (4) to be smooth and decomposable into series

$$G(\mu) \approx \sum_{m=1}^M \varphi_m(x) E[\varphi_m(\mathbf{X}_t(\mu))], \quad G(\mu^\dagger) \approx \sum_{m=1}^M \varphi_m(x) \varphi_m(f_t)$$

and fidelity functional to be  $L^2$ -distance between data  $G(\mu^\dagger)$  and  $G(\mu)$ , where  $\mu^\dagger$  is exact solution of the inverse problem (1)-(2):

$$J_M(\mu) = \frac{1}{2} \sum_{t=1}^T \int \left| \sum_{m=0}^M \varphi_m(x) E[\varphi_m(\mathbf{X}_t(\mu))] - \varphi_m(x) \varphi_m(f_t) \right|^2 dx.$$

Since  $\{\varphi_m\}_{m \in \mathbb{N}}$  is an orthonormal basis, fidelity functional (5) has form

$$J_M(q) = \frac{1}{2} \sum_{t=1}^T \sum_{m=1}^M (E[\varphi_m(\mathbf{X}_t(q))] - \varphi_m(f_t))^2. \quad (7)$$

Here,  $E[\varphi_m(\mathbf{X}_t(q))]$  is an expectation from  $\varphi_m(\mathbf{X}_t(q))$  where  $\mathbf{X}_t(q)$  demonstrates the dependence of SDE solution of Fourier coefficients  $q$  for functions  $\mu$  and  $\sigma$  at moment  $t$ . The choice of the number of basis functions  $M$  needs to be properly balanced with the number of observations  $N$ . The specific form of the data fidelity functional as a kind of  $L^2$ -distance with statistics allows to apply gradient methods with early stopping as a regularization technique and parallelization of algorithm.

The approach consists in minimizing the data fidelity functional (7) by a gradient method, which leads to a stochastic version of the classical nonlinear Landweber iteration [9]

$$\mu_k = \mu_{k-1} + (G'(\mu_{k-1}))^* (f - G(\mu_{k-1})).$$

For high-dimensional SDEs only probabilistic approximations of the forward operator  $G$  are available. Moreover, it is possible to characterize the adjoint of the Fréchet derivative  $G'(\mu_{k-1})^*$  by a backward (adjoint) SDE. These representations give rise to a Monte-Carlo implementation which can also be applied for high-dimensional problems, where the numerical evaluation of the Fokker-Planck equation is otherwise numerically challenging or even impossible. Moreover, the representations do not rely on any properties of the Brownian motion and hence can be applied to SDE driven by suitable non-Markovian processes such as fractional Brownian motion.

Convergence and stability of this algorithm in more general form for noisy data is investigated in paper [14]. Its result states monotonic decrease of fidelity functional for a finite amount of steps, because of noise. Theoretical estimation of critical amount of steps is not available without information about noise, thus we investigate another early stopping rules.

### 3.1. Landweber iteration algorithm

We implement stochastic version of the classical gradient algorithm called Landweber iteration for solving minimization problem of data fidelity functional. The gradient of data fidelity functional is derived in [14] based on solutions of a series of deterministic adjoint problems. In numerical calculations we use synthetic data  $f_t$  that is a result of direct problem (1) solution with an exact functions  $\mu^\dagger(x)$  and  $\sigma^\dagger(x)$ , that is solved with Euler-Maruyama method (see Section 4). The *Landweber iteration algorithm* is follows:

Step 1. Choose the orthonormal basis  $\varphi_m(x)$ . In our experiments  $\varphi_m(x) = \cos(m\pi x/10)$ ,  $x \in [-3, 3]$ ,  $m = 1, \dots, M$ .

Step 2. Solve (1) for exact drift term  $\mu^\dagger$  and volatility  $\sigma^\dagger$  using Euler-Maruyama scheme with  $N = 10^2$  steps in time. Synthetic data will be  $f_t = \mathbf{X}_t$ .

Step 3. Put  $q^0 = 0$ .

Step 4. Suppose that we have  $q^{k-1}$ . If  $|(J_M(q^{k-2}) - J_M(q^{k-1})) J_M^{-1}(q^{k-2})| < \varepsilon$ , then  $q^* = q^{k-1}$ . Otherwise go to Step 5.

Step 5. Generate  $P$  trajectories  $X^{q^{k-1}, p}$  as solutions of direct problem (1) with approximate parameters  $\bar{\mu}^{q^{k-1}}, \bar{\sigma}^{q^{k-1}}$ .

Step 6. For each trajectory  $X^{q^{k-1}, p}$  solve an adjoint problem

$$\begin{cases} \frac{d}{dt} \lambda_{m,t}^p = -\mathcal{D}\mu(\mathbf{X}_t^{q^{k-1}, p}) \lambda_{m,t}^p, & t \in (0, T); \\ \lambda_{m,T}^p = \text{grad } \varphi_m(\mathbf{X}_T^{q^{k-1}, p}). \end{cases}$$

Here  $\text{grad } \varphi_m(x) = -\frac{m\pi}{10} \sin \frac{m\pi x}{10}$ ,  $\mathcal{D}\mu(x) = -\sum_{m=0}^M q_m \mu \varphi'_m(x)$ .

Step 7. Calculate the gradient for directions of  $q^{k-1}$  [14]:

$$J'_M(q) \partial q^{k-1} = \sum_{t=1}^T \sum_{m=1}^M (E[\varphi_m(\mathbf{X}_t)] - \varphi_m(f_t)) \cdot E \left[ \int_0^t \lambda_{m,s}^\top [\bar{\mu}' \partial q^{k-1}](\mathbf{X}_s^{q^{k-1}}) ds \right],$$

where

$$E \left[ \int_0^t \lambda_{m,s}^\top [\bar{\mu}' \partial q^{k-1}](\mathbf{X}_s^{q^{k-1}}) ds \right] = \frac{1}{P} \sum_{p=1}^P \int_0^t \lambda_{m,s}^{p\top} [\bar{\mu}' \partial q^{k-1}](\mathbf{X}_s^{q^{k-1}, p}) ds.$$

Step 8. Find the next approximation of  $q$  using formula:

$$q^k = q^{k-1} - J'_M(q^{k-1}) \partial q^{k-1}, \quad \partial q = |J'|^{-2}.$$

Return to Step 4.

The Tikhonov regularization of an inverse problem (1)-(2) which is formulated as a minimization problem for the regularized function

$$T_{M,\alpha}(q) = J_M(q) + \alpha \sum_{m=0}^M q_m^2 \quad (8)$$

and analysis of a regularization parameter of iteration algorithm is investigated in Section 4.2.

## 4. Numerical results

In this section we analyze numerical results of solving inverse problem (1)-(2) with synthetic data for some exact solution. Firstly we describe the parallelization algorithm for graphics accelerators. Then we validate algorithm with initialised synthetic data. The experiments were performed on Nvidia GeForce 940MX videocard with 384 CUDA cores.

## 4.1. Parallelization of an algorithm

Mathematical expectations in algorithm allows us to parallelize its computation with respect to generated trajectory  $\mathbf{X}$ . It was implemented with Numba library for Python 3.

Specialty of parallelization is that computational complexity is not that important as amounts of transferred data, because it takes a lot more time to deliver data from video card to host.

For example of such program we will take a look into couple of our functions (also called kernels). First of all we estimate expected value of solution for (1) with known  $\sigma$  and  $\mu$ .

```
@cuda.jit # this wrapper turns function into kernel
def avarage(res, N_time, num_of_paths, x0, q_mu, q_sigma,
           rng_states):
    # find thread that we are in.
    thread_id = cuda.grid(1)
    #generate path with additional func.
    euler_maruyama(res[thread_id], N_time, x0, q_mu, q_sigma,
                  rng_states)
```

**Figure 1:** Avarage function for CUDA.

Firstly, we may notice, that our kernel does not return any value, as everything we pass to it will be returned, that's why resulting value is also passed. Secondly we called `cuda.grid(1)`. This function returns coordinate we are in, what helps us to write data to different places and not to overwrite data with different threads at the same time. Finally we generate trajectory for (1) in `euler_maruyama` function.

```
@cuda.jit(device=True)
def euler_maruyama(path, x0, N_time, q_mu, q_sigma, rng_states):
    thread_id = cuda.grid(1)
    path[0] = x0
    dt = 1
    for j in range(1, N):
        # generating standard normal variable.
        dw = xoroshiro128p_normal_float32(rng_states, thread_id)
        path[j] = path[j-1] + mu(path[j-1], q_mu) * dt + \
                sigma(path[j-1], q_sigma) * dw
```

**Figure 2:** Euler-Maruyama parallel realization.

Here, we show that this function is used only in video card device, and it can not be called from main program, so it must be used in kernels or other device functions.

To generate random variables, one should use built-in generator, based on xoroshiro128 algorithm. To use it we should create states for iterating through generating random sequence. `rng_states = create_xoroshiro128p_states(num_of_paths, seed=42)`, where `num_of_paths` stand for number of states to be simultaneously iterated at the same time and `seed` may be any constant integer, that serves to reproduce the same random sequences every run. At last, we calculate  $\mu$  and  $\sigma$  according to their basis decomposition.

Moreover, if one run out of cores to calculate with, we may improve `avarage` function

In figure 3, we create a local array that is used only in one thread. Note that vector operations are not supported and they are must be done in cycles or additional kernels.

### 4.1.1. Analysis of effectiveness of parallelization

Table 1 shows times of computations of one iteration and per core efficiency coefficients for parallelization for different amount of cores as well as problem difficulty. Note the linear dependency on amount of steps in time and quadratic on amount of basis functions, thus we acquired quadratic computation difficulty for amount of unknowns.



```
@cuda.jit
def avarage(res, num_of_paths, x0, q_mu, q_sigma, rng_states, paths):
    # find thread that we are in.
    thread_id = cuda.grid(1)
    # create local array on videocard.
    path = cuda.local.array(N_time, nb.float32)
    for i in range(kernel_cycles):
        euler_maruyama(path, x0, N, q_mu, q_sigma, rng_states)
        for j in range(N):
            res[thread_id, j] += path[j]
    for j in range(N):
        res[thread_id, j] /= kernel_cycles
```

Figure 3: Avarage function for small amount of cores.

		Per core efficiency								Time of one iteration (seconds)								
		Amount of cores								Amount of cores								
$N_T$	$M$	2	4	8	16	32	64	128	256	1	2	4	8	16	32	64	128	256
100	5	1.00	0.99	0.99	0.94	0.88	0.78	0.51	0.29	20.2	10.1	5.05	2.54	1.33	0.71	0.4	0.31	0.27
100	10	1.00	0.99	0.99	0.95	0.94	0.86	0.65	0.36	83.4	41.7	20.9	10.4	5.48	2.77	1.5	1.0	0.89
100	20	0.99	0.99	0.99	0.95	0.94	0.89	0.69	0.41	320	160	80.3	40.1	20.9	10.6	5.62	3.6	3.06
500	5	0.98	0.98	0.94	0.91	0.86	0.67	0.35	0.12	102	51.8	26.1	13.6	7.03	3.69	2.37	2.25	3.17
500	10	0.99	0.99	0.96	0.92	0.88	0.73	0.43	0.17	414	208	104	53.6	28.1	14.5	8.76	7.4	9.36
$10^3$	5	0.98	0.97	0.91	0.88	0.80	0.53	0.22	0.07	206	104	52.5	28.0	14.5	8.01	5.95	7.07	11.1

Table 1: Efficiency of parallelization per core for different amount of cores and problem statements, where  $N_T$  – amount of steps in time,  $M$  – amount of basis functions for parameters approximation.

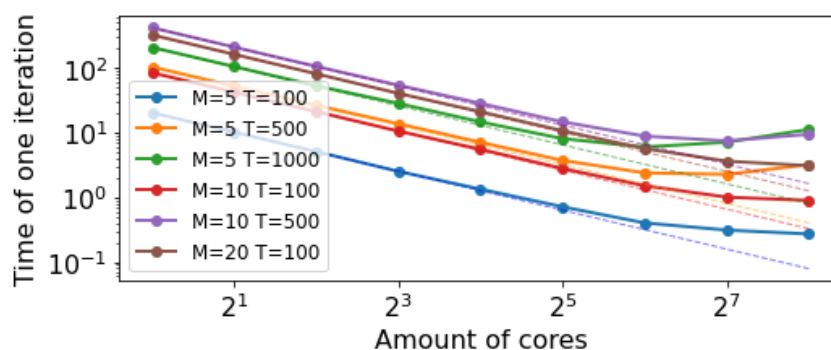


Figure 4: Time (in seconds) of computation of one iteration for different amount of cores, where  $T$  is an amount of steps in time,  $M$  is an amount of basis functions for parameters approximation.

Figure 4 shows that after  $2^5$  amount of cores parallelization starts to be ineffective, or even more time-consuming, due to big amounts of data, that is simultaneously transferred to videocard memory.

#### 4.2. The results of solving of an inverse problem

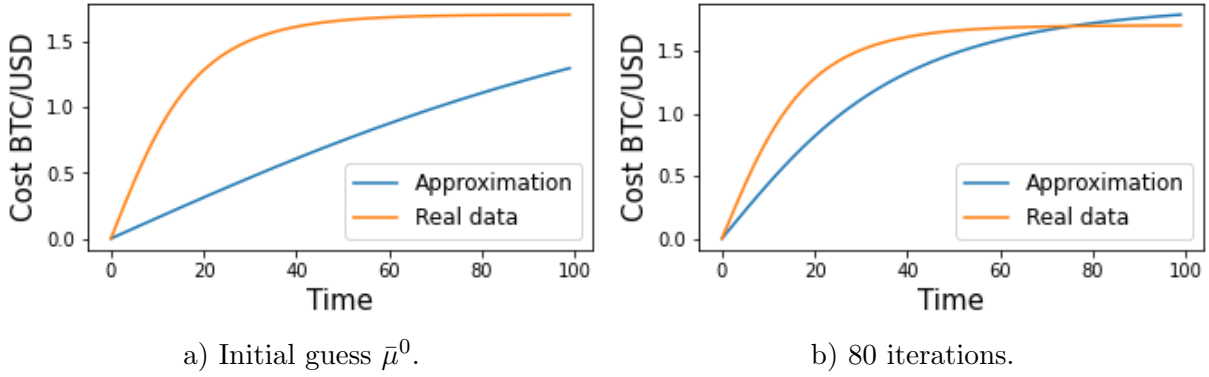
For an initial test of algorithm, let  $N = 100$  steps in time,  $P = 1980$  trajectories for finding mathematical expectations. Firstly, we run a Landweber algorithm for non-regularized data fidelity functional (7). At the second step, we apply Tikhonov regularization (minimization of function  $T_{M,\alpha}(q)$ ) for avoiding unstable effects in solution recovery.

#### 4.2.1. Validation with synthetic data.

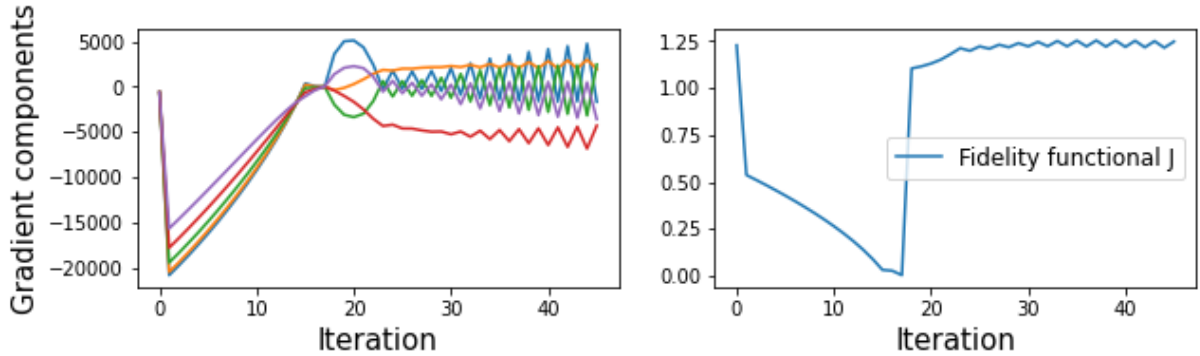
First of all, we generate synthetic data  $f_t = X(\mu^\dagger, \sigma^\dagger)$  as a solution of (1) with known parameters of the same form of Fourier series (6) to check if algorithm is correctly implemented.

We start with parameter  $\sigma^\dagger = 0$  and set  $\mu^\dagger$  with  $q_{m\mu}^\dagger = \sin \frac{m\pi}{10}$  and let number of basis functions be  $M = 5$  and synthetic data points  $T = 100$ .

Initial guess is taken linear  $\bar{\mu}^0 = 0.01$ . The solution of SDE (1) for the approximate  $\bar{\mu}$  by Landweber iterations in comparison with synthetic data  $f_t$  are displayed in fig. 5.



**Figure 5:** Data approximation  $X(\bar{\mu}, \bar{\sigma})$  (blue curve) for Landweber iteration with non-regularized data fidelity function with  $\partial q = |J'|^{-2}$ . The orange curve is the synthetic data  $f_t$ .

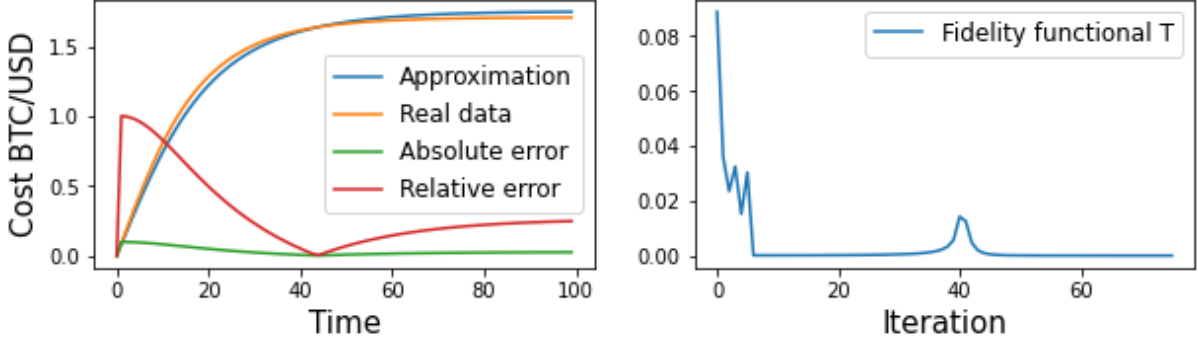


**Figure 6:** The gradient (8) of fidelity function depends on iterations (left). Non-regularized fidelity functional (7) evolution for Landweber iteration without stopping rule (right).

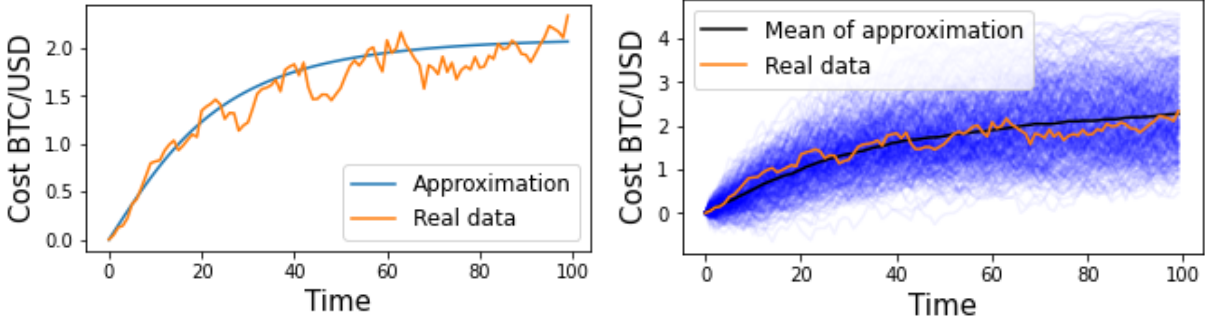
To regularise unstable steps with  $\partial q = |J'|^{-2}$  (see fig. 6) we apply smaller steps  $\partial q = J|J'|^{-2}$  as fidelity functional and gradient approach zero and with Tikhonov regularization with parameter  $\alpha = 0.1$ . The results of these regularizations are shown in fig. 7. This leads us to better results with  $J(\bar{\mu}) = 6.87 \cdot 10^{-6}$  (see fig. 7), while previously we got just  $J(\bar{\mu}) = 1.66 \cdot 10^{-2}$  (see fig. 5). Thus we consider method to be able to find exact solution of deterministic problem.

Now, we add to unknown  $\mu^\dagger$  the identification of volatility  $\sigma^\dagger$ , that is set with  $q_{m\sigma}^\dagger = \cos \frac{m\pi}{10}$ . To identify them, we execute algorithm for  $\bar{\mu}$  only. After its determination we try to identify  $\bar{\sigma}$ . As soon as fidelity functional consists of mathematical expectations and does not consider deviations explicitly, noise is not identifiable (see fig. 8b). Moreover, noise moved data sufficiently, thus  $\mu$  may be identified only with error about amplitude of noise.

For further inspection of error on noise dependence, we choose constant noise parameter  $\sigma^\dagger$



**Figure 7:** Restoration of noise-free data using Landweber iteration with  $\partial q = J|J'|^{-2}$  (left). Here blue curve is  $X(\bar{\mu}, \sigma^\dagger)$ , orange curve – synthetic data  $f_t$ , green line – absolute error  $|X(\bar{\mu}, \sigma^\dagger) - f_t|$ , red curve – relative error. The regularized data fidelity function (8) evolution is demonstrated from the right.



a) only mean  $\bar{\mu}$  restoration.

b) noise  $\bar{\sigma}$  after mean  $\bar{\mu}$  restoration.

**Figure 8:** Restoration of noisy data of 10% using Landweber iteration for regularized function  $T_{M,\alpha}(\mu), \alpha = 0.1$ .

and observe maximum relative errors:

$$E_\mu = \max_{x \in [-3,3]} \left| \mu^\dagger(x) - \bar{\mu}(x) \right| / \left| \mu^\dagger(x) \right|, \quad E_q = \max_{m=1,\dots,M} \left| q_m^\dagger - q_m \right| / \left| q_m^\dagger \right|.$$

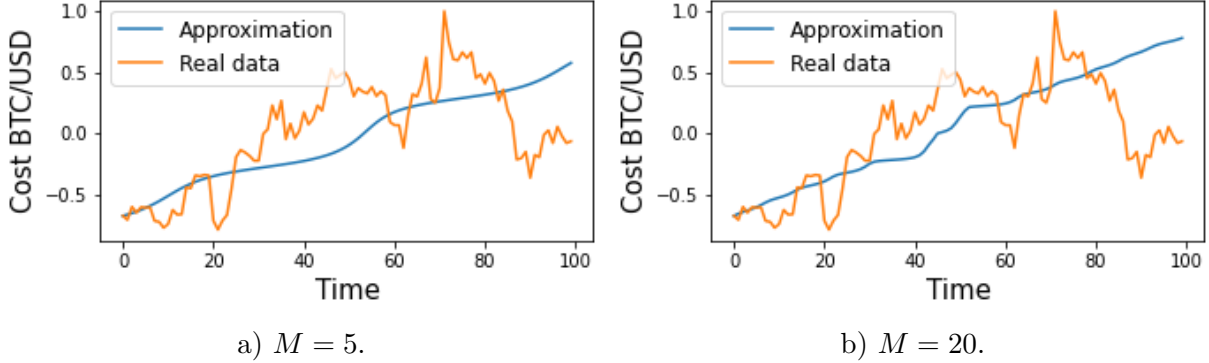
	$\sigma = 0$	$\sigma = 0.025$	$\sigma = 0.05$	$\sigma = 0.1$	$\sigma = 0.15$
$E_q$	0.13306	0.20600	0.33770	0.75031	1.22765
$E_\mu$	0.05151	0.19826	0.31381	0.58208	1.23405
$T_{M,\alpha}$	$1.6601 \cdot 10^{-6}$	$5.3638 \cdot 10^{-5}$	$2.0925 \cdot 10^{-5}$	$1.1478 \cdot 10^{-4}$	$1.2214 \cdot 10^{-2}$

**Table 2:** Relative errors and achieved fidelity functional  $T_{M,\alpha}$  with different volatility level  $\sigma$ .

From Table 2 we may practically confirm estimation of error in relation to data noise, that is linear [14]. Achieved fidelity functional is growing with growth of noise in measurements due to smooth approximation of noisy discontinuous data that is the less smooth, the more noise coefficient is set.

#### 4.2.2. Real Bitcoin data

The Bitcoin prices in period since 01.04.2020 until 01.08.2020 were used. The real data was normalized with subtracting mean and dividing by maximum value. This lets us apply the same set of basis functions to all data sets.



**Figure 9:** Real data approximations (blue curves) with initial guess  $\bar{\mu}^0 = 0$  using Landweber iteration for regularized data fidelity term (8). The orange curves are the real Bitcoin prices.

For small amount of basis functions we can not describe all data details and linear part prevails over other parts of approximation (see fig. 9). From the other side, too large choice of  $M$  may result in description of every jump of data, including all the noise, what is also not acceptable. Moreover, initial linear guess lets us find different local minimums of fidelity functional, and its right choice may help avoid local minimums.

#### 4.2.3. Analysis of results

One may notice that approximations are monotonic. This effect takes place because of form of parameters we chose. If we consider equation (1) with  $\sigma = 0$ , we may notice that if  $\mu(X_{t_0}) = 0$  at some point  $t_0$  then  $X_t = X_{t_0}$  for  $t > t_0$  as  $dX = 0$ . These points of equilibrium are stable if every eigenvalue of Jacobian matrix of  $\mu(X_{t_0})$  has negative real part, or  $\mu'(X_{t_0}) < 0$  in our situation. This fact adds additional requirement for data. It must be path of Wiener process, that has strictly monotonic mean (in case there are none stable points) or mean is constant after some point in time.

## 5. Conclusion and discussions

We presented an implementation of Landweber iteration algorithm for drift and volatility coefficients identification in stochastic differential equation with Wiener process. This method is preferable instead of classical methods based on probability density satisfied to Fokker-Planck equation due to more straightforward computation process (especially in multidimensional SDEs). The expression for gradient of data fidelity functional is connected with the solution of an deterministic adjoint problem for SDE. Moreover, iteration algorithm allows one to parallel the computational process using graphics accelerators, that was shown to be effective.

This problem is solved with applying of Tikhonov regularization and limitation of iteration steps. Nevertheless, existence of multiple solutions remains and Tikhonov regularization just delays the unstability of process and does not resolves it.

Even though the method is flexible and allows to add multiple parameters, lack of input data does not allow us identify all of them. In our case, noise was not identifiable, thus it was also affecting recovering mean.

It was shown, that method is applicable to narrow set of one dimensional functions. Due

to existence of stable points, their mean must be monotonic, otherwise estimation of mean does not describe data, instead data is completely estimated by noise.

## References

1. Artemiev S., Korneev V. Numerical solution of stochastic differential equations on supercomputers. *Numerical Analysis and Applications* 4, 1–11, 2013.
2. Bakushinskii A. On a convergence problem of the iterative-regularized gauss-newton method. *Zhurnal Vychislitelnoi Matematiki i Matematicheskoi Fiziki* 32(9), 1503–1509, 1992.
3. Black F., Scholes M. S. The pricing of options and corporate liabilities. *Journal of Political Economy*, 1973.
4. Blaschke B., Neubauer A., Scherzer O. On convergence rates for the iterativelyregularized gauss-newton method. *IMA Journal of Numerical Analysis* 17, 1997.
5. Burger, M., Osher, S. Convergence rates of convex variational regularization. *Inverse Problems* 20(5), 1411–1421, 2004.
6. Comte F., Genon-Catalot V., Rozenholc Y. Penalized nonparametric mean square estimation of the coefficients of diffusion processes. *Bernoulli*, 13, 2007.
7. Dunker F., Hohage T. On parameter identification in stochastic differential equations by penalized maximum likelihood. *Inverse Problems*, 30(9), 095001, 2014.
8. Eggermont P. Maximum entropy regularization for fredholm integral equations of the first kind. *SIAM J. Math. Anal.* 24, 1557–1576, 1993.
9. Hanke M., Neubauer A., Scherzer O. A convergence analysis of the Landweber iteration for nonlinear ill-posed problems. *Numerical Mathematics*, 1995.
10. Hoffmann M. Adaptive estimation in diffusion processes. *Stochastic Process, Appl.*, 1999.
11. Hohage T. Logarithmic convergence rates of the iteratively regularized Gauss-Newton method for an inverse potential and an inverse scattering problem. *Inverse Problems* 13, 1279–1299, 1997.
12. Hurn A., Jeisman J., Lindsay, K. Teaching an old dog new tricks: Improved estimation of the parameters of stochastic differential equations by numerical solution of the Fokker-Planck equation. *NCER Working Paper Series* 9, 2007.
13. Kabanikhin S. *Inverse and Ill-posed Problems. Theory and Applications*, de Gruyter, 2011.
14. Kaltenbacher B., Schopfer F., Schuster T. Iterative methods for non-linear ill-posed problems in Banach spaces: convergence and applications to parameter identification problems. *Inverse Problems*, 25(6):065003, 19, 2009.
15. Kaltenbacher B., Neubauer A., Scherzer O. *Iterative Regularization Methods for Nonlinear ill-posed Problems*. Radon Series on Computational and Applied Mathematics, 2008.
16. Mil'shtejn G. N. *Approximate integration of stochastic differential equations. Theory of Probability and Its Applications*.
17. Papaspiliopoulos O., Pokern Y., Roberts G., Stuart A. *Biometrika. Statistics* 99(3), 2012.

18. Risken H. The Fokker-Planck equation. Methods of solution and applications. Springer Series in Synergetics, Springer-Verlag, 1989.
19. Schmitter E. Penalized nonparametric drift estimation for a multidimensional diffusion process. *Statistics* 47(1), 61–84, 2013.
20. Spokoiny V. Adaptive drift estimation for nonparametric diffusion model. *Ann. Statist.* 28(3), 815–836, 2000.

# VaLiPro: валидатор решений задач линейного программирования для кластерных вычислительных систем\*

Л.Б. Соколинский, И.М. Соколинская

Южно-Уральский государственный университет

В статье представлен параллельный алгоритм валидации решений задач линейного программирования. Идея метода состоит в том, чтобы генерировать регулярный набор точек на гиперсфере малого радиуса, центрированной в точке тестируемого решения. Целевая функция вычисляется для каждой точки валидационного множества, принадлежащей допустимой области. Если все полученные значения меньше или равны значению целевой функции в точке, проверяемой как решение, то эта точка считается корректным решением. Параллельная реализация алгоритма VaLiPro выполнена на языке C++ с использованием параллельного BSF-каркаса, инкапсулирующего в проблемно-независимой части своего кода все аспекты, связанные с распараллеливанием программы на базе библиотеки MPI. Приводятся результаты масштабных вычислительных экспериментов на кластерной вычислительной системе, подтверждающие эффективность предложенного подхода.

*Ключевые слова:* линейное программирование, валидатор решений, VaLiPro, параллельный алгоритм, кластерные вычислительные системы, параллельный BSF-каркас.

## 1. Введение

Эра больших данных [1,2] привела к появлению задач линейного программирования (ЛП) сверхбольших размерностей [3]. Подобные задачи возникают в экономике, промышленности, логистике, статистике, квантовой физике и других областях. Решение таких сверхбольших задач невозможно без масштабируемых параллельных алгоритмов, ориентированных на кластерные вычислительные системы. В соответствии с этим в последние годы интенсифицировались усилия по разработке новых и модернизации известных параллельных алгоритмов решения задач ЛП. В качестве примеров можно привести работы [4–8]. При разработке новых масштабируемых алгоритмов ЛП возникает необходимость их тестирования на различных задачах. Источниками таких задач могут быть как эталонные репозитории, например Netlib-Lp [9], так и генераторы случайных задач, см. например [10,11], в которых решение заранее известно. При этом на практике часто встречаются классы задач с неизвестными решениями. При тестировании ЛП-решателя на таких классах задач возникает необходимость валидации, сертификации и уточнения полученного решения. Проблеме сертификации и уточнения решения задачи ЛП посвящен ряд работ. В статье [12] был предложен метод, проверяющий вычисленное решение на его принадлежность допустимой области и оптимальность на основе рациональной арифметики. Указанный подход впоследствии был реализован с более высокой эффективностью Кохом (Koch) в [13] и Апплгейтом (Applegate) с соавторами в [14]. Кох разработал программу верификации *perPlex*, базирующуюся на симплекс-методе и использующую разреженную LU-декомпозицию с выбором ведущего элемента по Марковицу (Markowitz pivoting). Апплгейт с соавторами разработали программу *QSopt\_ex* на основе известной библиотеки GMP (GNU Multi Precision), позволяющей выполнять вычисления с произвольной точностью. Программа стартует, начиная с точности, поддерживаемой целевой вычислительной системой, затем происходит переход к 128 разрядам, и затем разрядность динамически увеличивается на 50%, пока не будет получено решение, удовлетворяющее всем ограничениям с требуемой точностью. Программа также может использоваться для проверки задачи ЛП на неразрешимость или неограниченность. Основным недостатком программы *QSopt\_ex* является то, что использование

\* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-07-00092-а.

дробно-рациональных вычислений в случае больших и сложных задач ЛП требует больших временных затрат. Панюков и Горбик в работе [15] попытались преодолеть этот недостаток путем использования параллельных вычислений на распределенной памяти. Ими предложены два метода распараллеливания симплекс-метода. Первый метод основан на декомпозиции симплекс-таблицы по столбцам. Второй метод базируется на методе обратной матрицы и использует декомпозицию исходной матрицы по столбцам, а обратной – по строкам. Однако результаты проведенных экспериментов недостаточно убедительны, так как отсутствует сравнение с лучшими последовательными решениями, вычисления проводились только с использованием трех разреженных задач ЛП (число ненулевых элементов не превышало 5%), и, кроме этого, максимальная граница масштабирования ограничивалась 16 процессорами. Еще одно оригинальное решение предложено Глейксером (Gleichner) и соавторами в [16]. В этой работе описывается процедура итеративного уточнения решения задачи ЛП с любой заданной точностью. Точное решение вычисляется путем решения последовательности связанных задач ЛП с использованием арифметики ограниченной точности. Решаемые задачи ЛП имеют ту же матрицу коэффициентов, что и исходная задача, преобразуются только целевая функция, правые части ограничений и границы переменных.

Все рассмотренные выше методы концентрируются на уточнении уже найденного приближенного решения. Если же найденное решение находится слишком далеко от правильного, что означает наличие ошибки в алгоритме, то применение этих методов становится нецелесообразным. Кроме того, все указанные алгоритмы имеют высокую вычислительную сложность и при этом не допускают эффективного распараллеливания на больших кластерных вычислительных системах. Метод, предлагаемый в данной статье, ориентирован на отладку и валидацию новых алгоритмов решения задач ЛП на кластерных вычислительных системах. Он реализован в виде параллельной программы *VaLiPro (Validator of Linear Program)*, обладающей хорошей масштабируемостью на многопроцессорных вычислительных системах с распределенной памятью. Статья организована следующим образом. В разделе 2 дается формальное описание предлагаемого метода валидации решений задач ЛП и приводится последовательная версия алгоритма *VaLiPro*. В разделе 3 рассматривается параллельная версия алгоритма *VaLiPro*. В разделе 4 предлагается ее реализация с использованием параллельного BSF-каркаса и приводятся результаты масштабных вычислительных экспериментов на кластерной вычислительной системе, подтверждающие эффективность предложенного подхода. В разделе 5 суммируются полученные результаты и приводятся планы по использованию валидатора *VaLiPro* для разработки искусственной нейронной сети, способной решать задачи ЛП большой размерности.

## 2. Метод валидации решений задач ЛП

Пусть в евклидовом пространстве  $\mathbf{R}^n$  задана задача линейного программирования

$$\bar{x} = \arg \max \{ \langle c, x \rangle \mid Ax \leq b, x \in \mathbf{R}^n \}, \quad (1)$$

где  $c$  – вектор коэффициентов целевой функции. Здесь и далее  $\langle \square, \square \rangle$  обозначает скалярное произведение двух векторов. Обозначим  $M = \{ x \in \mathbf{R}^n \mid Ax \leq b \}$  – множество допустимых точек задачи (1). По определению множество  $M$  является выпуклым. Везде далее мы будем предполагать, что  $M$  является непустым, ограниченным (и, следовательно, замкнутым) множеством, то есть задача (1) имеет, по крайней мере, одно решение. Пусть  $\tilde{x} \in \mathbf{R}^n$  – приближенное решение задачи (1), полученное с помощью некоторого ЛП-решателя, и требующее сертификации.

Идея метода валидации *VaLiPro*, описываемого в данной работе, заключается в построении конечного множества точек  $V$ , покрывающих гиперсферу  $S$  малого радиуса  $\rho$  с центром в точке сертифицируемого решения  $\tilde{x}$ :

$$V \subset S = \{ x \in \mathbf{R}^n \mid \|x - \tilde{x}\|^2 = \rho^2 \}.$$



Здесь и далее  $\|\cdot\|$  обозначает евклидову норму. На множестве точек  $V \cap M$  вычисляется максимум целевой функции:

$$\bar{v} = \arg \max \{ \langle c, v \rangle \mid v \in V \cap M \}.$$

Если  $|\langle c, \bar{v} \rangle - \langle c, \tilde{x} \rangle| < \varepsilon$ , то приближенное решение  $\tilde{x}$  считается корректным. В противном случае  $\tilde{x}$  считается некорректным решением. Здесь  $\varepsilon \in \mathbf{R}_{>0}$  – некоторая малая положительная константа, являющаяся параметром алгоритма валидации.

Опишем метод построения валидационного множества  $V$ . Известно [17], что координаты любой точки  $v = (v_1, \dots, v_n)$ , лежащей на поверхности гиперсферы  $S$ , задаваемой уравнением

$$\|x - \tilde{x}\|^2 = \rho^2,$$

могут быть представлены в виде

$$\begin{aligned} v_1 &= \rho \cos \phi_1, \\ v_j &= \rho \cos \phi_j \prod_{i=1}^{j-1} \sin \phi_i, \quad (j = 2, \dots, n-2), \\ v_{n-1} &= \rho \sin \theta \prod_{i=1}^{n-2} \sin \phi_i, \\ v_n &= \rho \cos \theta \prod_{i=1}^{n-2} \sin \phi_i, \end{aligned} \tag{2}$$

где  $0 \leq \phi_j \leq \pi$  ( $j = 1, \dots, n-2$ ),  $0 \leq \theta < 2\pi$ . Алгоритм генерации валидационного множества  $V$  проиллюстрируем на трехмерной сфере (см. рис. 1). Зафиксируем *нечетное число параллелей*  $d \geq 3$  (полюса исключаются). Положим

$$\varphi = \pi/d. \tag{3}$$

В плоскости  $(x_1, 0, x_3)$  отложим от оси  $(0, x_1)$  углы  $0, \varphi, \dots, (d-1)\varphi$ . Получившиеся лучи в пересечении со сферой дадут множество точек  $\{v_0, \dots, v_{d-1}\}$ , которые однозначно определяют  $d$  параллелей. Теперь в плоскости  $(x_1, 0, x_2)$  аналогичным образом отложим от оси  $(0, x_1)$  углы  $0, \varphi, \dots, (d-1)\varphi$  и определим  $d$  меридианов. Точки, получающиеся на пересечении параллелей и меридианов, за исключением точек полюсов, образуют валидационное множество точек на трехмерной сфере. В общем виде описанный метод генерации точек валидационного множества для  $n \geq 3$  представлен в виде алгоритма 1а. Вложенные циклы с заголовками на шагах 3, 5, 8, 10 фактически генерируют сферические координаты очередной валидационной точки:

$$(\rho, \phi_1, \phi_2, \dots, \phi_{n-2}, \theta). \tag{4}$$

На шагах 12-19 сферические координаты преобразуются в декартовы по формулам (2). Используя границы переменных в заголовках циклов на шагах 3, 5, 8, 10, легко подсчитать, что алгоритм 1а порождает  $2d(d+1)^{n-2}$  точек. Недостатком алгоритма 1а является то, что он генерирует некоторые точки с повторениями. Проведенный вычислительный эксперимент показал, что для размерности  $n=4$  при количестве параллелей  $d=5$  алгоритм 1а порождает 189 дубликатов

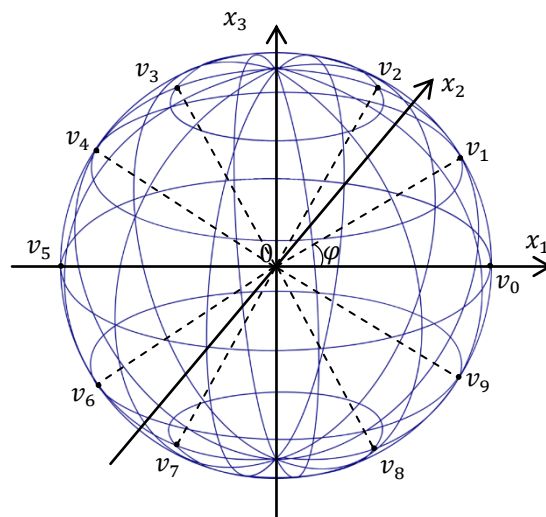


Рис. 1. Построение точек валидационного множества  $V$  на трехмерной сфере ( $n = 3, d = 5$ ).

при

Алгоритм 1. Генерация точек валидационного множества	
а) С дубликатами	б) Без дубликатов
<pre> 1: input d 2: <math>\varphi := \pi/d</math> 3: for <math>j_{n-1} = 0 \dots (2d - 1)</math> do begin 4:   <math>\theta := j_{n-1}\varphi</math> 5:   for <math>j_{n-2} = 0 \dots d</math> do begin 6:     <math>\phi_{n-2} := j_{(n-2)}\varphi</math> 7:     ... 8:     for <math>j_2 = 0 \dots d</math> do begin 9:       <math>\phi_2 := j_2\varphi</math> 10:      for <math>j_1 = 0 \dots d</math> do begin 11:        <math>\phi_1 := j_1\varphi</math> 12:        <math>\Pi := 1</math> 13:        <math>v_1 := \rho \cos(\phi_1)</math> 14:        for <math>l = 2 \dots n - 2</math> do begin 15:          <math>\Pi := \sin(\phi_{l-1})\Pi</math> 16:          <math>v_l := \rho \cos(\phi_l)\Pi</math> 17:        end 18:        <math>v_{n-1} := \rho \sin(\theta)\Pi</math> 19:        <math>v_n := \rho \cos(\theta)\Pi</math> 20:        output v 21:      end 22:    end 23:    ... 24:  end 25: end 26: stop </pre>	<pre> 1: input d, <math>\rho</math> 2: <math>\varphi := \pi/d</math> 3: for <math>j_{n-1} = 0 \dots (2d - 1)</math> do begin 4:   <math>\theta := j_{n-1}\varphi</math> 5:   for <math>j_{n-2} = 1 \dots (d - 1)</math> do begin 6:     <math>\phi_{n-2} := j_{n-2}\varphi</math> 7:     ... 8:     for <math>j_2 = 1 \dots (d - 1)</math> do begin 9:       <math>\phi_2 := j_2\varphi</math> 10:      for <math>j_1 = 1 \dots (d - 1)</math> do begin 11:        <math>\phi_1 := j_1\varphi</math> 12:        <math>\Pi := 1</math> 13:        <math>v_1 := \rho \cos(\phi_1)</math> 14:        for <math>l = 2 \dots n - 2</math> do begin 15:          <math>\Pi := \sin(\phi_{l-1})\Pi</math> 16:          <math>v_l := \rho \cos(\phi_l)\Pi</math> 17:        end 18:        <math>v_{n-1} := \rho \sin(\theta)\Pi</math> 19:        <math>v_n := \rho \cos(\theta)\Pi</math> 20:        output v 21:      end 22:    end 23:    ... 24:  end 25: end 26: stop </pre>

общем количестве точек равно 360, что составляет более 50%. Дубликаты порождаются в итерациях, когда  $\phi_i = 0$  или  $\phi_i = \pi$ , что соответствует значениям  $j_i = 0$  и  $j_i = d$  ( $i=1, \dots, n-2$ ). Это вызвано тем, что в этом случае один из сомножителей  $\sin \phi_i$  в (2) окажется равным нулю и, следовательно, вариации других сомножителей уже не смогут изменить значение соответствующей координаты. Решение проблемы дубликатов без серьезной переделки алгоритма 1а достигается путем задания границ переменных циклов  $j_1, \dots, j_{n-2}$  от 1 до  $d-1$ , как это сделано в алгоритме 1б (шаги 5, 8 и 10). При этом вместе с дубликатами теряется и некоторое количество значимых точек. При  $n=4$  и  $d=5$  это количество равно 11, что составляет менее 7% от полного набора после удаления дубликатов. Эксперименты показали, что такая потеря не оказывает существенного влияния на качество работы валидационного алгоритма. Анализ алгоритма 1б показывает, что общее количество точек валидационного множества  $V$ , генерируемых этим алгоритмом, определяется по формуле

$$|V| = 2d(d-1)^{n-2}. \quad (5)$$

Недостатком алгоритма 1б является то, что количество циклов зависит от размерности задачи, что не позволяет использовать размерность как параметр программы. Кроме того, этот подход оказывается неприемлемым для больших размерностей, так как, например, для размерности  $n=1000$  нам пришлось бы написать тысячу циклов for. Для преодоления указанного недостатка мы использовали вектор-функцию  $g: \{0, \dots, 2d(d-1)^{n-2} - 1\} \rightarrow \square^n$ , вычисляющую координаты точки валидационного множества по ее номеру  $k$  (нумерация начинается с нуля) в том поряд-

ке, в котором их генерирует алгоритм 1б. Определение этой функции представлена в виде алгоритма 2.

---

**Алгоритм 2.** Функция  $g$  (вычисление точки  $v$  по ее номеру  $k$ )

---

```

1: function  $g(k)$  begin
2:    $u_{n-1} := \lfloor k/(d-1)^{n-2} \rfloor$ 
3:    $u_n := u_{n-1}$ 
4:    $k := k \bmod (d-1)^{n-2}$ 
5:   for  $j = (n-3) \dots 0$  do begin
6:      $u_j := \lfloor k/(d-1)^j \rfloor + 1$ 
7:      $k := k \bmod (d-1)^j$ 
8:   end
9:    $\Pi := 1$ 
10:   $\varphi := \pi/d$ 
11:   $v_1 := \rho \cos(u_1 \varphi)$ 
12:  for  $j = 2 \dots (n-2)$  do begin
13:     $\Pi = \Pi \sin(u_{j-1} \varphi)$ 
14:     $v_j = \rho \cos(u_j \varphi) \Pi$ 
15:  end
16:   $\Pi = \Pi \sin(u_{n-2} \varphi)$ 
17:   $v_{n-1} := \rho \sin(u_{n-1} \varphi) \Pi$ 
18:   $v_n := \rho \cos(u_n \varphi) \Pi$ 
19:  return  $v$ 
20: end
```

---



---

**Алгоритм 3.** Валидация решения  $\tilde{x}$  задачи ЛП

---

```

1: input  $n, A, b, c, d, \rho, \varepsilon, \tilde{x}$ 
2:    $\varphi := \pi/d$ 
3:   for  $k = 0 \dots 2d(d-1)^{n-2} - 1$  do begin
4:      $v := g(k)$ 
5:     if  $Av \leq b \wedge \langle c, v \rangle > \langle c, \tilde{x} \rangle + \varepsilon$  goto 9
6:   end
7:   output "Solution"  $\tilde{x}$  "is correct."
8:   goto 10
9:   output "Solution"  $\tilde{x}$  "is incorrect."
10: stop
```

---

Финальная реализация метода VaLiPro, использующая вектор-функцию  $g$ , представлена в виде алгоритма 3. В качестве дополнительного параметра этого алгоритма фигурирует малая положительная константа  $\varepsilon$  (по умолчанию  $\varepsilon = 10^{-6}$ ), нивелирующая возможные погрешности вычислений при сравнении значений целевой функции на шаге 5. Кратко прокомментируем шаги алгоритма 3. На шаге 1 вводятся исходные данные задачи ЛП (1), параметры алгоритма и сертифицируемое решение  $\tilde{x}$ . На шаге 2 вычисляется угол  $\varphi$  в соответствии с формулой (3). Шаг 3 открывает цикл по номеру точки  $k$ , изменяющемуся от 0 до  $2d(d-1)^{n-2} - 1$  в соответствии с формулой (5). На шаге 4 с помощью вектор-функции  $g$  (см. алгоритм 2) вычисляется очередная валидационная точка  $v$ . На шаге 5 проверяется, принадлежит ли  $v$  допустимой области задачи (1), и сравниваются значения целевой функции в точке  $v$  и в точке сертифицируемого решения  $\tilde{x}$ . Если целевая функция принимает большее значение в точке  $v$  и эта точка является допустимой, то осуществляется переход на шаг 9, где выводится сообщение, что сертифицируемое решение не является корректным. В противном случае осуществляется переход к следующей итерации цикла. Если цикл завершился естественным образом, управление переходит на шаг 7, где выводится сообщение, что решение  $\tilde{x}$  является корректным, после чего осуществляется переход на шаг 10, где алгоритм завершает свою работу.

### 3. Параллельный алгоритм валидации решений задач ЛП

В соответствии с формулой (5) мощность валидационного множества точек, генерируемого алгоритмом 3, экспоненциально зависит от размерности пространства. Поэтому алгоритм 3 будет иметь высокую вычислительную сложность для больших размерностей. Для сокращения временных затрат на вычисления нами была разработана и реализована параллельная версия алгоритма 3, представленная в виде алгоритма 4. Данный алгоритм разработан в соответствии с моделью параллельных вычислений BSF [18], использующей парадигму «мастер-рабочий» [19]. В соответствии с этой моделью узел-мастер является центром управления и коммуникации. Все узлы-рабочие выполняют один и тот же код, но над разными данными. BSF-модель

требует представления алгоритма в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определяемых формализмом Бирда-Миртенса (Bird–Meertens formalism) [20].

Алгоритм 4. Параллельный алгоритм валидации решения задачи ЛП	
Мастер	$l$ -й рабочий ( $l=0, \dots, L-1$ )
M1:	W1: <b>input</b> $n, A, b, c, d, \rho, \varepsilon, \tilde{x}$
M2:	W2: $W_l := [lK/L, \dots, (l+1)K/L - 1]$
M3:	W3: $Z_l := \text{Map}(f_{\tilde{x}}, W_l)$
M4:	W4: $s_l := \text{Reduce}(\wedge, Z_l)$
M5: $\text{RecvFromWorkers}(s_1, \dots, s_L)$	W5: $\text{SendToMaster}(s_l)$
M6: $s := \text{Reduce}(\wedge, [s_1, \dots, s_L])$	W6:
M7: <b>if</b> $s = \text{false}$ <b>goto</b> M10	W7:
M8: <b>output</b> "Solution" $\tilde{x}$ "is correct."	W8:
M9: <b>goto</b> M11	W9:
M10: <b>output</b> "Solution" $\tilde{x}$ "is incorrect."	W10:
M11: <b>stop</b>	W11: <b>stop</b>

Функция высшего порядка *Map* преобразует исходный список  $W = [w_0, \dots, w_{K-1}]$  в список  $Z = [z_0, \dots, z_{K-1}]$ , применяя к каждому элементу функцию  $f_{\tilde{x}}$ :

$$Z = \text{Map}(f_{\tilde{x}}, W) = [f_{\tilde{x}}(w_0), \dots, f_{\tilde{x}}(w_{K-1})].$$

В данном случае элементами списка  $W$  являются номера точек валидационного множества:

$$W = [0, \dots, K-1] \quad (K = 2d(d-1)^{n-2}).$$

Булева функция  $f_{\tilde{x}} : \{0, \dots, K-1\} \rightarrow \{\text{true}, \text{false}\}$  задается следующей формулой:

$$f_{\tilde{x}}(w) = \begin{cases} \text{true} & | A \cdot g(w) \leq b \wedge \langle c, g(w) \rangle \leq \langle c, \tilde{x} \rangle, \\ \text{false} & | A \cdot g(w) > b \vee \langle c, g(w) \rangle > \langle c, \tilde{x} \rangle, \end{cases}$$

где вектор-функция  $g$  вычисляет координаты валидационной точки по ее номеру  $w$ . Функция  $f_{\tilde{x}}$  возвращает значение «истина» (*true*), если точка  $g(w)$  принадлежит допустимой области задачи (1), и значение целевой функции в этой точке меньше, либо равно значению целевой функции в точке  $\tilde{x}$ . В противном случае функция  $f_{\tilde{x}}$  возвращает значение «ложь» (*false*). Список  $Z = [z_0, \dots, z_{K-1}]$ , таким образом, содержит булевы индикаторы для всех точек валидационного множества. Если хотя бы один элемент в этом списке имеет значение *false*, то точка  $\tilde{x}$  считается некорректным решением задачи (1).

Функция высшего порядка *Reduce* преобразует список  $Z = [z_0, \dots, z_{K-1}]$  в атомарное булево значение  $s$ , применяя операцию конъюнкции  $\wedge$  ко всем элементам списка  $Z$ :

$$s = \text{Reduce}(\wedge, Z) = z_0 \wedge \dots \wedge z_{K-1}.$$

В параллельном алгоритме 4 на шаге W2  $l$ -тый рабочий определяет свою часть  $W_l$  списка  $W$ :

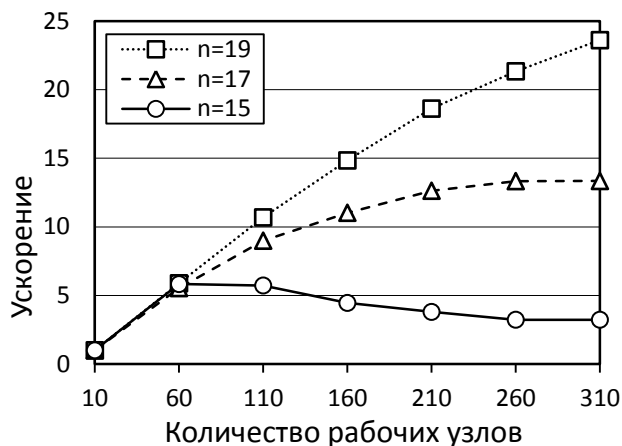
$$W_l = [lK/L, \dots, (l+1)K/L - 1].$$

Здесь  $L$  обозначает количество рабочих. Для простоты мы предполагаем, что  $K$  кратно  $L$ . На шаге W3 рабочий применяет функцию *Map* к своей части списка. Получившийся список булевых значений на шаге W4 редуцируется к одному булеву значению  $s_l$  путем применения функции *Reduce*, первым параметром которой является операция логического «и». Вычисленное таким образом логическое значение  $s_l$  на шаге W5 пересылается мастеру. Мастер на шаге M5 получает от рабочих все вычисленные значения. На шаге M6 список полученных значений ре-

дуцируется к одному булеву значению  $s$  с помощью функции *Reduce*. На шагах M7-M10 анализируется вычисленное булево значение  $s$  и выводится соответствующее заключение.

**Табл. 1.** Характеристики кластера «Торнадо ЮУрГУ».

Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 cores 3.33 GHz)
Количество процессоров в узле	2
Оперативная память узла	24 GB DDR3
Соединительная сеть	InfniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS



**Рис. 2.** Ускорение параллельного алгоритма VaLiPro для различных размерностей.

#### 4. Программная реализация и вычислительные эксперименты

Параллельный алгоритм 4 был нами реализован на языке C++ с использованием параллельного BSF-каркаса [21,22]. BSF-каркас базируется на модели параллельных вычислений BSF [18] и инкапсулирует в проблемно независимой части своего кода все аспекты, связанные с распараллеливанием программы с помощью библиотеки MPI [23]. Исходные коды параллельной программы VaLiPro свободно доступны в сети Интернет по адресу <https://github.com/leonid-sokolinsky/BSF-LPP-Validator>. С использованием указанной программы нами были проведены масштабные вычислительные эксперименты на вычислительном кластере «Торнадо ЮУрГУ» [24], характеристики которого приведены в табл. 1. Для экспериментов использовались случайные задачи ЛП, полученные с помощью программы FRaGenLP [25] при следующих значениях параметров: длина ребра ограничивающего гиперкуба  $\alpha = 200$ , радиус большой гиперсферы  $\theta = 100$ , радиус малой гиперсферы  $\rho = 50$ , верхняя граница «почти параллельности» для гиперплоскостей  $L_{max} = 0.35$ , минимальная допустимая близость для гиперплоскостей  $S_{min} = 100$ , максимальное допустимое абсолютное значение при генерации коэффициентов случайных неравенств  $a_{max} = 1000$ , максимальное допустимое абсолютное значение при генерации правых частей случайных неравенств  $b_{max} = 10000$ . Эксперименты проводились для размерностей  $n = 15$ ,  $n = 17$  и  $n = 19$ . Количество неравенств, соответственно, составляло 46, 52 и 58. Из них случайных: 15, 17 и 19 соответственно. Решения задач получались с помощью апекс-метода [4]. Во всех случаях при запуске валидатора использовались следующие значения параметров:  $d = 5$ ,  $\rho = 1$ ,  $\varepsilon = 10^{-6}$ . Результаты экспериментов представлены на рис. 2. Время верификации решения для случайной задачи ЛП размерности  $n = 19$  на конфигурации из одного узла-мастера и одного узла-рабочего составило 17 минут. На конфигурации из одного узла-мастера и 310 узлов-рабочих верификация решения такой же задачи заняла 4 секунды. Анализ результатов показывает, что граница масштабируемости предложенного алгоритма существенно зависит от размерности задачи (под границей масштабируемости здесь понимается максимум кривой ускорения). При  $n = 19$  параллельная версия алгоритма VaLiPro продемонстрировала масштабируемость, близкую к линейной, вплоть до 310 процессорных узлов. Для задачи размерности  $n = 17$  граница масштабируемости составила примерно 260 узлов, а на задаче размерности  $n = 15$  она уменьшилась до 60 узлов. Это связано с тем, что задача такой размерности

уже не способна загрузить работой большое количество процессорных узлов: время, затрачиваемое на передачу данных по сети, начинает превалировать над временем, затрачиваемым на вычисления, и процессоры начинают простаивать.

## 5. Заключение

В статье представлен параллельный алгоритм VaLiPro для валидации решений задач линейного программирования на кластерных вычислительных системах. Идея валидационного алгоритма заключается в генерации регулярного множества точек на гиперсфере малого радиуса с центром в проверяемом решении. Решение считается корректным, если все точки валидационного множества, принадлежащие допустимой области задачи линейного программирования, характеризуются меньшим значением целевой функции по сравнению с проверяемой точкой. Реализация параллельного алгоритма VaLiPro была выполнена на языке C++ с использованием параллельного BSF-каркаса, инкапсулирующего в проблемно независимой части своего кода все аспекты, связанные с распараллеливанием программы с помощью библиотеки MPI. Исходные коды разработанной параллельной программы свободно доступны в сети Интернет по адресу <https://github.com/leonid-sokolinsky/BSF-LPP-Validator>. Предложенный метод валидации является универсальным и годится для любых задач линейного программирования. Преимуществом параллельного алгоритма VaLiPro является почти линейное ускорение для достаточно больших размерностей пространства, начиная с размерности 19. Существенным недостатком, ограничивающим применение предложенного метода, является экспоненциальный рост количества точек валидационного множества при увеличении размерности пространства, что приводит к экспоненциальному росту временной сложности алгоритма. Описанный алгоритм был использован вместе с алгоритмом FRaGenLP [25] и апекс-методом [4] для автоматической генерации обучающего набора из 70 000 прецедентов, который планируется использовать для разработки искусственной нейронной сети, способной решать многомерные задачи линейного программирования.

## Литература

1. Jagadish H. V. et al. Big data and its technical challenges // *Communications of the ACM*. 2014. Vol. 57, no. 7. P. 86–94. DOI:10.1145/2611567.
2. Hartung T. Making Big Sense From Big Data // *Frontiers in Big Data*. 2018. Vol. 1. P. 5. DOI:10.3389/fdata.2018.00005.
3. Соколинская И.М., Соколинский Л.Б. О решении задачи линейного программирования в эпоху больших данных // *Параллельные вычислительные технологии (ПаВТ'2017)*. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2017. С. 471–484.
4. Соколинский Л.Б., Соколинская И.М. Исследование масштабируемости апекс-метода для решения сверхбольших задач линейного программирования на кластерных вычислительных системах // *Суперкомпьютерные дни в России: Труды международной конференции*. 21-22 сентября 2020 г. Москва: МАКС Пресс, 2020. С. 49–59.
5. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method // *Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science*, vol. 9295 / ed. Zaroliagis C., Pantziou G., Kontogiannis S. Cham: Springer, 2015. P. 281–307. DOI:10.1007/978-3-319-24024-4\_17.
6. Huangfu Q., Hall J.A.J. Parallelizing the dual revised simplex method // *Mathematical Programming Computation*. Springer Verlag, 2018. Vol. 10, no. 1. P. 119–142. DOI:10.1007/s12532-017-0130-5.
7. Tar P., Stágel B., Maros I. Parallel search paths for the simplex algorithm // *Central European Journal of Operations Research*. Springer Verlag, 2017. Vol. 25, no. 4. P. 967–984. DOI:10.1007/s10100-016-0452-9.

8. Yang L., Li T., Li J. Parallel predictor-corrector interior-point algorithm of structured optimization problems // 3rd International Conference on Genetic and Evolutionary Computing, WGEC 2009. 2009. P. 256–259. DOI:10.1109/WGEC.2009.68.
9. Gay D.M. Electronic mail distribution of linear programming test problems // Mathematical Programming Society COAL Bulletin. 1985. no. 13. P. 10–12.
10. Charnes A. et al. On Generation of Test Problems for Linear Programming Codes // Communications of the ACM. 1974. Vol. 17, no. 10. P. 583–586. DOI:10.1145/355620.361173.
11. Arthur J.L., Friendewey J.O. GENGUB: A generator for linear programs with generalized upper bound constraints // Computers and Operations Research. Pergamon, 1993. Vol. 20, no. 6. P. 565–573. DOI:10.1016/0305-0548(93)90112-V.
12. Dhiflaoui M. et al. Certifying and Repairing Solutions to Large LPs How Good are LP-solvers? // SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms. USA: Society for Industrial and Applied Mathematics, 2003. P. 255–256.
13. Koch T. The final NETLIB-LP results // Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI:10.1016/S0167-6377(03)00094-4.
14. Applegate D.L. et al. Exact solutions to linear programming problems // Operations Research Letters. North-Holland, 2007. Vol. 35, no. 6. P. 693–699. DOI:10.1016/j.orl.2006.12.010.
15. Панюков А.В., Горбик В.В. Применение массивно-параллельных вычислений для решения задач линейного программирования с абсолютной точностью // Автоматика и телемеханика. 2012. № 2. С. 73–88.
16. Gleixner A.M., Steffy D.E., Wolter K. Iterative refinement for linear programming // INFORMS Journal on Computing. INFORMS Inst.for Operations Res.and the Management Sciences, 2016. Vol. 28, no. 3. P. 449–464. DOI:10.1287/ijoc.2016.0692.
17. Blumenson L.E. A Derivation of n-Dimensional Spherical Coordinates // The American Mathematical Monthly. 1960. Vol. 67, no. 1. P. 63–66. DOI:10.2307/2308932.
18. Sokolinsky L.B. BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems // Journal of Parallel and Distributed Computing. 2021. Vol. 149. P. 193–206. DOI:10.1016/j.jpdc.2020.12.009.
19. Sahni S., Vairaktarakis G. The master-slave paradigm in parallel computer and industrial settings // Journal of Global Optimization. 1996. Vol. 9, no. 3–4. P. 357–377. DOI:10.1007/BF00121679.
20. Bird R.S. Lectures on Constructive Functional Programming // Constructive Methods in Computing Science. NATO ASI Series F: Computer and Systems Sciences, vol. 55 / ed. Broy M. Berlin, Heidelberg: Springer, 1988. P. 151–216.
21. Соколинский Л.Б. Параллельный программный каркас BSF. Свидетельство о государственной регистрации программы для ЭВМ 2020661344, 22.09.2020.
22. Sokolinsky L.B. BSF-skeleton [Electronic resource]. 2019. URL: <https://github.com/leonid-sokolinsky/BSF-skeleton> (accessed: 09.04.2021).
23. Gropp W. MPI 3 and Beyond: Why MPI Is Successful and What Challenges It Faces // Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol. 7490 / ed. Träff J.L., Benkner S., Dongarra J.J. Berlin, Heidelberg: Springer, 2012. P. 1–9. DOI:10.1007/978-3-642-33518-1\_1.
24. Kostenetskiy P.S., Safonov A.Y. SUSU Supercomputer Resources // Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies (PCT 2016). CEUR Workshop Proceedings. Vol. 1576. 2016. P. 561–573.
25. Соколинский Л.Б., Соколинская И.М. FRaGenLP: генератор случайных задач линейного программирования для кластерных вычислительных систем // Параллельные вычислительные технологии (ПаВТ'2021). Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2021. С. 244–254. DOI:10.14529/pct2021.

# Автоматизация процессов преподавания практики на примере курса по параллельному программированию

А. Ю. Нестеров

Нижегородский государственный университет им. Н. И. Лобачевского

Отрасль разработки программного обеспечения активно развивается, что приводит к увеличению потребности в квалифицированных специалистах. В ответ на это в профильных вузах увеличивается прием на направления подготовки, связанные с тематикой информационных технологий. В результате нагрузка на преподавателей, ведущих программистские дисциплины, существенно возрастает. Одним из способов, ведущих к повышению качества образования в новых условиях, является внедрение автоматизированных процессов, связанных с проверкой практических работ, выполняемых студентами. Данная работа описывает соответствующий опыт, полученный при чтении курса «Параллельное программирование» в Институте информационных технологий, математики и механики ННГУ.

*Ключевые слова:* параллельное программирование, образовательный курс, автоматизация процессов преподавания.

## 1. Введение

Для подготовки специалистов в сфере высокопроизводительных вычислений в Институте информационных технологий, математики и механики (ИТММ) ННГУ многие годы преподается годовой курс по основам параллельного программирования. Этот курс делится на две большие части: методы параллельного программирования для вычислительных систем с распределенной памятью и методы параллельного программирования для систем с общей памятью. Каждая из частей имеет свои сложности и нюансы в освоении материала. Также каждая часть данного курса имеет теоретическую [1, 2] и практическую [3, 4] составляющие.

В данной работе речь пойдет о практической составляющей всего курса, точнее, об автоматизации проверки выполняемых студентами заданий. Сама по себе идея автоматизации проверки широко известна и активно применяется на практике, но стоит отметить, что при чтении курса по параллельному программированию возникают дополнительные сложности, связанные с оценкой производительности и масштабируемости кода. Практика показывает, что преодолеть эти сложности не так просто.

Кратко рассмотрим, какие технологии рассказываются в курсе. Первая часть курса включает в себя изучение наиболее распространённого стандарта интерфейса обмена данными – MPI [5]. Вторая часть курса включает в себя изучение открытого стандарта OpenMP [6], Intel Threading Building Blocks — кроссплатформенной библиотеки шаблонов, разработанной компанией Intel [7], и класса thread [8] стандартной библиотеки шаблонов C++. В целом весь курс базируется на задачах, выбранных из задачника [9], а процесс обучения выглядит следующим образом. Традиционно, студенты посещают лекции и, начиная с некоторого момента времени, приступают к выполнению практических заданий, которые, в последствии, вручную, очно проверяет преподаватель практики.

Данный подход к ведению практических занятий имеет ряд недостатков. Во-первых, вследствие увеличения набора значительно возрастает нагрузка на преподавателей (курс слушает до 200 студентов ежегодно) и понижается качество проверки. Во-вторых, вышеизложенный подход вызывает сложности при работе в дистанционном режиме, технические проблемы с демонстрацией кода остались, а время их исправления в дистанционном режиме в среднем увеличилось вдвое. В-третьих, никто не отменяет существование субъективизма и человеческого фактора, в том числе и в образовательной сфере. Для преодоления указанных проблем были поставлены следующие задачи: построить автоматическую систему проверки работ студентов по



курсу параллельного программирования, а также сохранить и, желательно, увеличить качество проверки работ при росте числа студентов, посещающих курс.

В 2018 году автором данной статьи была представлена библиотека с открытым исходным кодом `parallel_programming_course` [10], реализующая полуавтоматическую проверку работ по основам параллельного программирования. Использование репозитория с исходными кодами библиотеки на протяжении более двух лет позволило сделать некоторые выводы для повышения оценки качества работ, а также непрерывно перестраивать архитектуру данной библиотеки с последующими планами модификаций для улучшения автоматизации. В данной работе представлен подход к организации автоматизированной проверки и первые результаты его применения, полученные за последние несколько лет. Конечной целью является создание системы полной автоматической проверки работ курса по основам параллельного программирования без участия преподавателя практики.

## 2. Жизнеспособность представленной системы

### 2.1 Причины ввода автоматизации

Во введении частично были описаны причины ввода автоматизации проверки работ. Еще раз перечислим данные причины:

1. Увеличение количества студентов и нагрузки на преподавателя.
2. Ухудшение качества проверки работ.
3. Потребность в независимой от преподавателя системе проверки работ.

Данный список причин позволил сформулировать ряд требований к проектируемой системе:

1. Система должна быть устойчивой к проверке работ, имеющих недетерминированный характер. Данное требование основывается на работе параллельных программ и на пункте 2 из перечисленных выше причин.
2. Система должна быть автономной и работающей круглосуточно независимо от дня и времени суток. Данное требование основано на пункте 3 перечисленных выше причин.
3. Программа, написанная студентом, должна быть в рабочем состоянии в течение всего семестра и являться гарантией для допуска к зачету/экзамену курса или очному опросу студента по данной лабораторной работе. Данное требование продиктовано организационными процессами обучения в вузах.

Перечисленные выше причины и требования позволили спроектировать систему, основанную на *принципах жизнеспособности*, описанных далее.

### 2.2 Основные идеи жизнеспособности системы

В данной части будет описан термин *жизнеспособность системы* в контексте данной работы, а также рассказано о том, где получать идеи для дальнейшего развития системы. Рассмотрим ряд примеров, которые помогут прийти к данным определениям.

**Первый пример.** Существуют промышленные проекты, комплексы программ и библиотек. Их жизнеспособность основана на требованиях бизнеса и динамическом развитии промышленной отрасли, все идеи по реализации и проектированию берутся непосредственно из задач, поставленных самим бизнесом или экономически выгодной предметной областью. Промышленный проект жив, пока есть смысл в получении прибыли от него.

**Второй пример.** Существуют студенческие проекты, комплексы программ и библиотек. Их жизнеспособность основана на мотивации самого студента или студентов, а также заинтересованности наставников, одним из примеров наставника является научный руководитель. Идеи дает наставник или сам студент. Студенческий проект жив, пока им занимается сам студент или его последователи.

Под *жизнеспособностью* данной проверочной системы будет пониматься следующее:

1. **Длительность существования курса по параллельному программированию.** Пока существует курс, существует и система для проверки данного курса.

2. **Простота проекта для малого порога вхождения нового преподавательского состава.** Использование новыми преподавателями данной системы продлит жизнь данной системе.
3. **Создание новых идей для модификации системы с каждой новой итерацией курса.** Позволит держать студентов в тонусе. Повторение одной и той же проверки ведет к предсказуемости и недополучению знаний.

Генерация и внедрение новых идей для модификации системы сводится к следующим правилам:

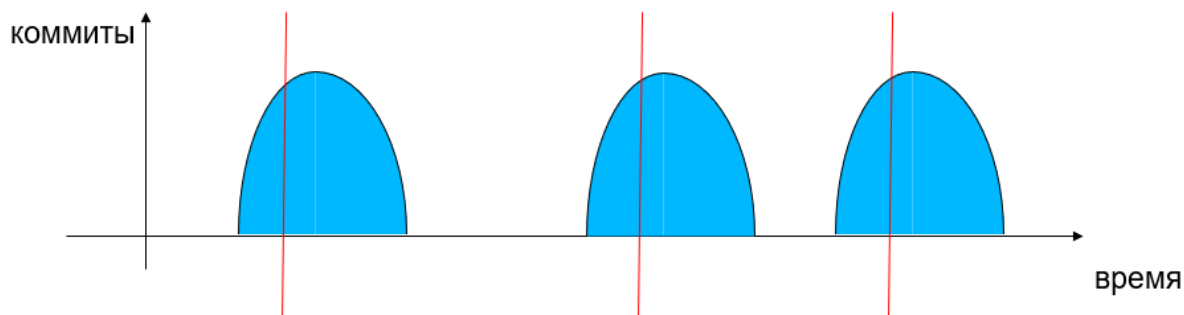
1. Новая идея изменений может возникнуть на основе обмана студентом системы. Недобросовестный студент захочет обойти систему проверок ради личной выгоды. Придуманный им обход проверок будет являться основой для будущих изменений в данной системе.
2. Новая идея изменений может возникнуть в стремлении улучшить критерий оценки качества контроля обучения. Стандартный критерий оценивания (неудовлетворительно, удовлетворительно, хорошо и отлично) не подходит для систем проверок задач программирования по причинам ограниченности диапазона значений. В данной статье и нашей системе проверки работ действует другой критерий оценивания для получения идей и улучшения работоспособности студентов – критерий непрерывности изменения кода.

### 2.3 Критерий непрерывности изменения кода

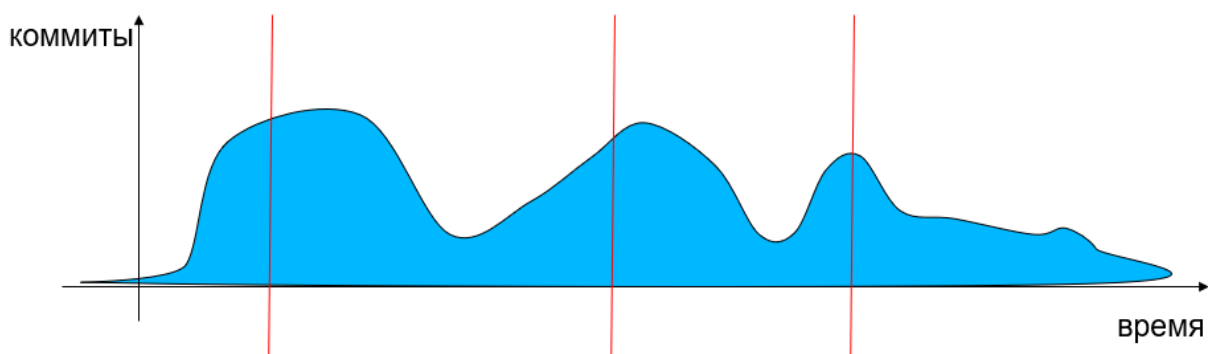
Предположим, что существует следующее утверждение: идеальный код (в любой интерпретации) не может быть написан мгновенно, но к нему можно стремиться на протяжении конкретного количества итераций его создания и изменений. За одну итерацию можно написать в худшем случае наихудшую версию кода. Для того, чтобы его улучшить, нужны непрерывные изменения. Обратимся к терминологии git-технологии [11] и назовем элементарным изменением кода *коммит в репозиторий*. Тогда для того, чтобы код становился лучше и лучше, нужно увеличивать количество коммитов в единицу времени, но заранее оговоримся, что каждый коммит заведомо проверяется и имеет рабочее состояние, тем самым побуждая студента только улучшать текущую версию исходного кода. Вышеизложенный подход назовем критерием непрерывности изменения кода.

Перейдем к примеру студенческой лабораторной работы и предположим, что у студента есть три задачи в семестре, каждая из которых имеет свой крайний срок сдачи. Если студент будет делать все работы в последний момент, то изменения его кода будут выглядеть следующим образом (рис. 1). Это означает ни что иное, как позднюю вовлеченность в работу. А качество сданных работ будет вызывать ряд проблем, среди которых частые падения в общем проекте (в основном падения из-за доступов к памяти, которые контролируются анализатором памяти), а также непредсказуемая работа реализованного параллелизма, отлавливаемая системами непрерывной интеграции. При этом если студент добросовестно отнесется к работе, то после ряда его собственных проверок и модификаций, изменения кода будут выглядеть уже по-другому (рис. 2). Идея добросовестной работы в том, чтобы начинать все задачи как можно раньше и не оставлять предыдущие сделанные задачи без внимания даже после их проверки в течение всего семестра. В идеале баллы по курсу формируются на основе поддержания всех задач в рабочем состоянии на конец семестра.

Можно также предположить, что если выполнять регулярные бесполезные коммиты, то критерий выполнится. Данную проблему в нашем проекте помог решить контроль графиков модификации кода (добавление и удаление участков кода в коммитах). Такие графики в идеале должны быть похожие на рис. 2, а также не должны совпадать или быть инверсией друг с другом. Данная процедура проверки создает сложность для недобросовестного выполнения критерия непрерывности изменения кода.



**Рис. 1.** Статистика работы студента при нарушении крайних сроков сдачи лабораторных работ (красные линии – поставленные срок сдачи).



**Рис. 2.** Статистика работы студента при добросовестном отношении к срокам сдачи лабораторных работ (красные линии).

Возвращаясь к получению идей модификации системы проверки работ, можно придумать способы синтетически приводить график, представленный на первом рисунке к графику на втором рисунке для преподавателя.

### 3. Хронология построения системы проверки работ

#### 3.1 Первоначальный вариант системы

Первоначальный вариант системы основывался на использовании технологий непрерывной интеграции на удаленных машинах и кроссплатформенной разработке на разных операционных системах на языках программирования С и С++. Требовалось разместить исходные коды студента на сайте GitHub.com с помощью системы контроля версий git [11], выбор данной технологии продиктован большим количеством разработчиков в open source среде на текущем сайте. Общим знаменателем для каждой сборки была система CMake [12], где с использованием одного или нескольких конфигурационных файлов можно построить проект и собрать его под различные операционные системы. Также для того, чтобы сборка запускалась на разных удаленных машинах, нужно было написать несколько YML-файлов для конфигурации систем непрерывной интеграции Travis [13] и AppVeyor [14], данные системы выбраны как лучший вариант использования бесплатных удаленных машин в образовательных целях.

Требования для реализации лабораторной работы студентом в первоначальной реализации системы были следующими:

1. Создать свой сpp-файл с функцией main и реализацией своей задачи.
2. Создать конфигурационный файл CMakeLists.txt и указать во всех остальных таких конфигурациях его существование.

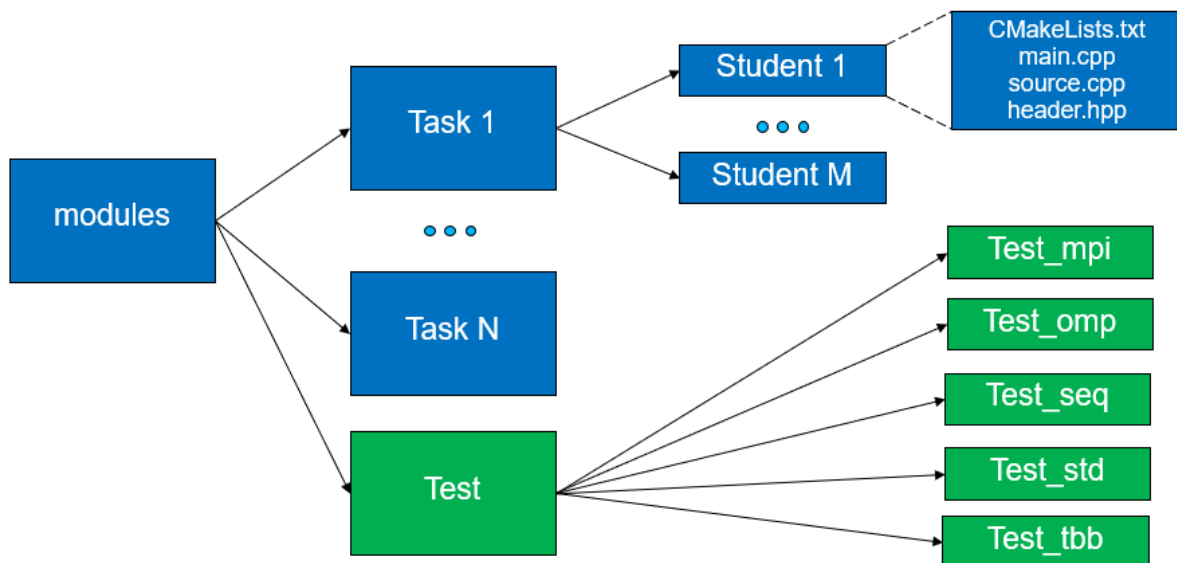
3. Все созданные файлы разложить в соответствующие директории, согласно инструкции.
4. Собрать проект с помощью системы CMake.
5. Указать изменения в конфигурациях систем непрерывной интеграции.
6. Отправить все изменения на сайт GitHub.com, которые автоматически запускают удаленные машины на сборку проекта.

### 3.2 Возникшие проблемы и изменения

Изначально предполагалось использование образовательных и соревновательных систем, таких как Яндекс.Контест, Stepik и других. Малая гибкость и минимальный контроль над студентами в системах-аналогах побудил нас создать собственную систему со своими требованиями к образовательному процессу под конкретный курс. Первоначальный вариант системы менялся с каждой новой итерацией курса. Все изменения были основаны на идеях жизнеспособности данной системы проверки работ, описанных ранее в работе. Из основ жизнеспособности системы возникли следующие проблемы, которые надо было решать на одной из новых итераций:

1. Написание своих реализаций тестирования задач на правильность решения заставляло преподавателя все равно заново перепроверять все работы. Решение данной проблемы заключалось в унификации тестирования. Замена реализации собственного main студента на написание тестов с помощью тестовой системы Google Test Framework [15].
2. Написание собственного конфигурационного файла CMakeLists.txt вызывало ряд проблем с пониманием скриптового языка CMake. Решение данной проблемы заключалось в подготовке шаблонов определенного рода скриптов. Позднее студенту нужно было только выбрать нужный шаблон из представленных в наборе.
3. Изменение YML-файлов для конфигурации удаленных машин также вызывало большие проблемы с пониманием работы проверочной системы у студентов. Данная проблема решалась изменением инструкции по использованию системы проверки. Возникло элементарное требование – назвать все свои файлы по шаблону, представленному в инструкции.
4. Несколько итераций курса по параллельному программированию позволило студентам накопить базу работ для помощи в списывании младшим студентам. Данная проблема решалась с помощью системы проверки кода на плагиат. Использовалась система JPlag [16], которая отлично показала себя в проверке работ более 100 студентов одновременно.
5. Также возникла проблема с качеством написания кода, от стиля кодирования, до проблем работы с памятью. Такие проблемы решались и решаются с помощью следующих инструментов, которые используется автором данной работы на протяжении нескольких лет: Google Style Code – проверка стиля кодирования [17], Valgrind – анализатор ошибок работы с памятью [18], CppCheck – инструмент статического анализа кода [19].

В ходе изменений также стабилизировалась архитектура самой тестовой системы проверки работ (рис. 3).



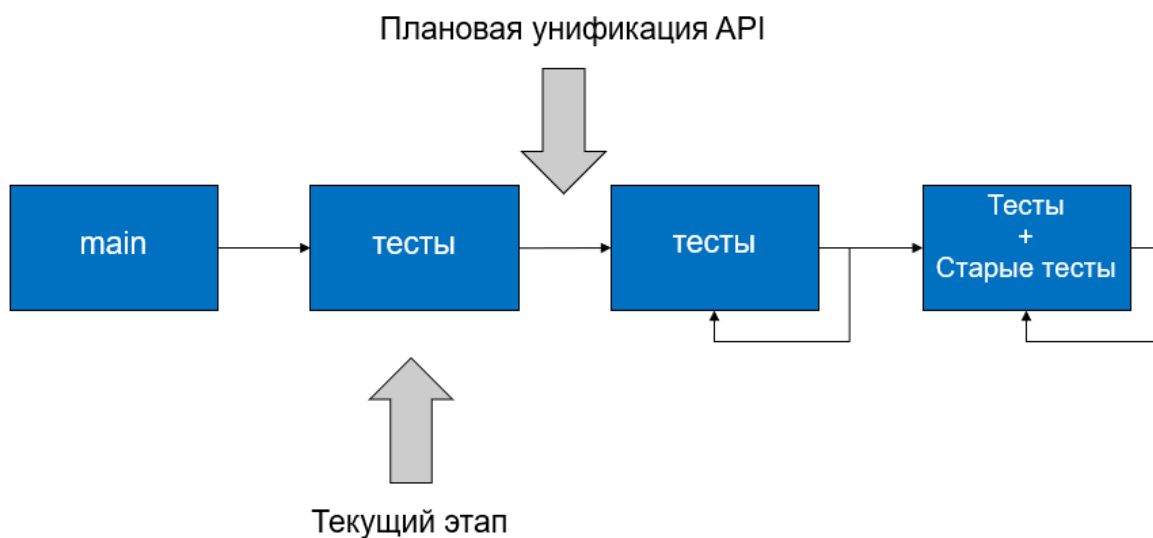
**Рис. 4.** Структура системы проверки работ

В исходных кодах [10] можно заметить следующие устойчивые архитектурные изменения:

1. В корне есть директория modules.
2. На втором уровне иерархии есть директории с задачами (task1, ..., taskN) и примерами реализаций (test) для простого порога вхождения.
3. На третьем уровне иерархии студент создает директорию с определенным названием (student1, ..., studentM) и помещает туда шаблонный скрипт CMakeLists.txt взятый из директории test, связанный с нужной технологией параллелизма. А также помещает свои реализации тестов и функций самой задачи в данную директорию.

### 3.3 Схема автоматизации

Идеи жизнеспособности нашей системы и возникшие, в ходе развития, проблемы позволили сформировать схему автоматизации (рис. 5) и спланировать дальнейшие модификации для минимизации влияния преподавателя на ход проверки работ.



**Рис. 5.** Схема автоматизации системы проверки работ

Схема, представленная выше, разделена на четыре больших этапа:

1. **Этап “main”.** Данный этап был описан ранее в работе. Его суть заключается в том, что студент должен был сам писать свои проверки работоспособности своего кода в main, что являлось дополнительной нагрузкой на преподавателя.
2. **Этап “тесты”.** Данный этап также был описан в работе. Его суть заключается в унификации тестирования. Происходит переход от написания собственных тестовых систем к написанию тестов в системе Google Test Framework.
3. **Этап “тестовое замыкание”.** На данный момент этот этап является планируемым после фазы унификации API – данная фаза является следствием текущей проблемы системы. Проблема заключается в отсутствие знаний о тестировании у студентов, что порождает ряд последствий в виде тестирования не всех случаев исполнения кода исходной задачи, избыточность данных и замедления процесса проверки, а также увеличения элементарных тестов, которые не приносят пользы для проверки задачи и системы в целом.

Унификация API включает в себя несколько частей:

- a. Разделить задачи, которые решает студент на согласованные темы.
- b. К каждой теме придумать свой набор названий и прототипов методов и функций.
- c. Ограничить студента в выборе прототипов функций и процедур для решения задач.

Данная методика позволит накопить тесты нужного формата и использовать их в дальнейших итерациях проверки.

4. **Этап “тесты + старые тесты”.** После проведения тестового замыкания накопится избыточная масса тестов, которую можно будет обновлять с помощью новых студентов и новых тестов. В ходе «тестовой эволюции», останутся только те тесты, которые улучшат качество проверки. При этом система не будет повторяться.

## 4. Результаты образовательных экспериментов

В ходе использования идей жизнеспособности текущей системы, возникших с помощью этих идей проблем, их решений, а также последствий модификации системы был выработан ряд принципов, которые мы реализовали на практике в курсе по параллельному программированию. Выработанные принципы разделились на две большие группы:

### 1. Принципы коллективной ответственности:

- a. Для всех студентов должна существовать единая очередь доступа к удаленным машинам в системе непрерывной интеграции. Единая очередь станет избыточной, если каждый студент захочет сдать задачу в один день перед крайним сроком сдачи, что в целом повлияет на распределение коммитов по времени.
- b. Если лабораторная работа не пройдет тестирование хотя бы один раз в течение семестра (а текущая работа повторяется каждый раз с каждым студентом, из-за того, что все работают в единой среде), то данная работа удаляется из системы и отдается на исправление в автоматическом режиме с оповещением самого студента. Частые выбросы работ из системы естественным образом нормализуют распределение коммитов во времени делают их равномерными.

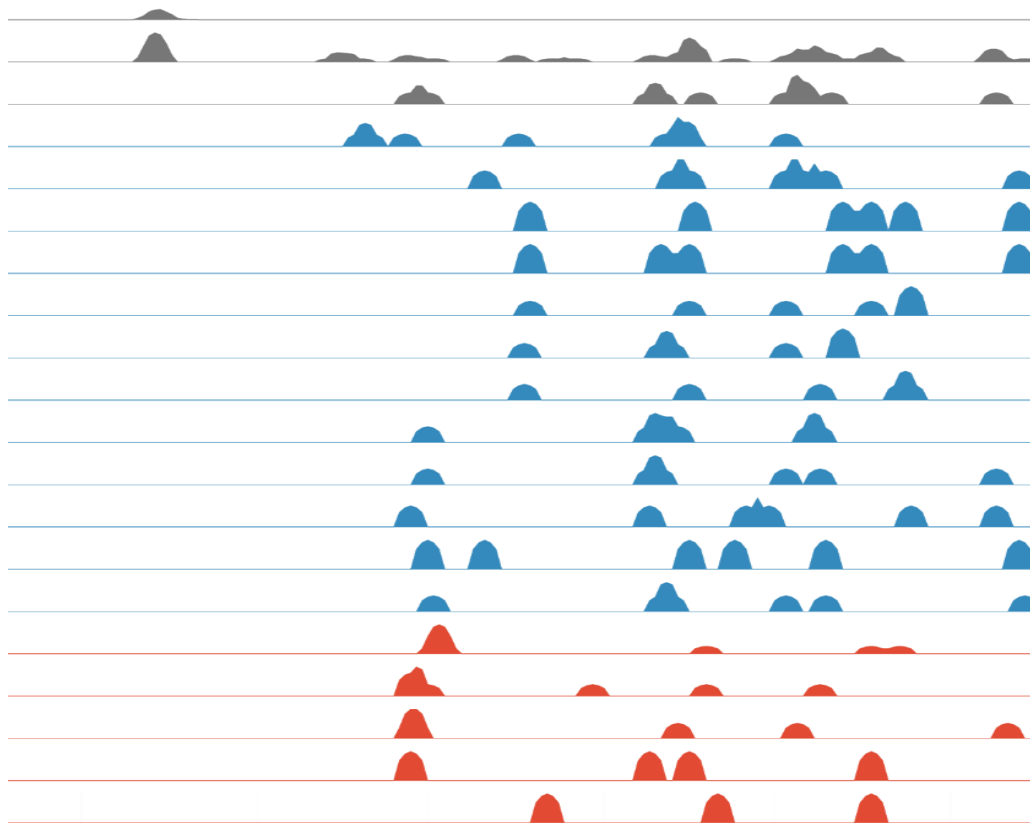
### 2. Принципы достижения максимального качества:

- a. Использование нескольких видов операционных систем и компиляторов, а также добавление Valgrind-анализатора для контроля памяти приводит к частым падениям некачественного кода. Вызывается принцип коллективной ответственности.
- b. Использование проверки стиля кодирования, добавление статического анализатора, а также использование анализа кода с помощью искусственного интеллекта на этапе предварительной проверки попадания в систему, помогает студенту проанализировать код повторно.

Также был проведен анализ коммитов нескольких студентов до использования приведенных принципов (рис. 6) и после их использования (рис. 7). Высота пика равна количеству коммитов в единицу времени. Левая граница рисунка обозначает начало семестра, а правая, соответственно, конец семестра. Семестры на двух представленных рисунках приближены друг к другу как по датам, так и по образовательным материалам. Анализируя полученные результаты, можно заметить большее рассеивание коммитов во времени, что и показывается в критерии непрерывности изменения кода. Данный критерий является одним из основных в текущей работе над системой, но не единственным. На текущий момент ведется еще один альтернативный эксперимент - исследование зависимости итоговых баллов работы студентов от вышеописанного критерия.

Данные принципы и сама система могут быть переиспользованы в других образовательных дисциплинах, основанных на программирование конечного числа задач в семестре с использованием языка C/C++.

Одним из основных результатов работы остается тот факт, что на начальном этапе работы преподавателем покрывались 1-2 группы в среднем по 15 человек в каждой, а на текущем этапе работы системы, покрытие занимает уже 7-8 групп в среднем по 20 человек.



**Рис. 6.** Коммиты нескольких студентов до использования указанных принципов

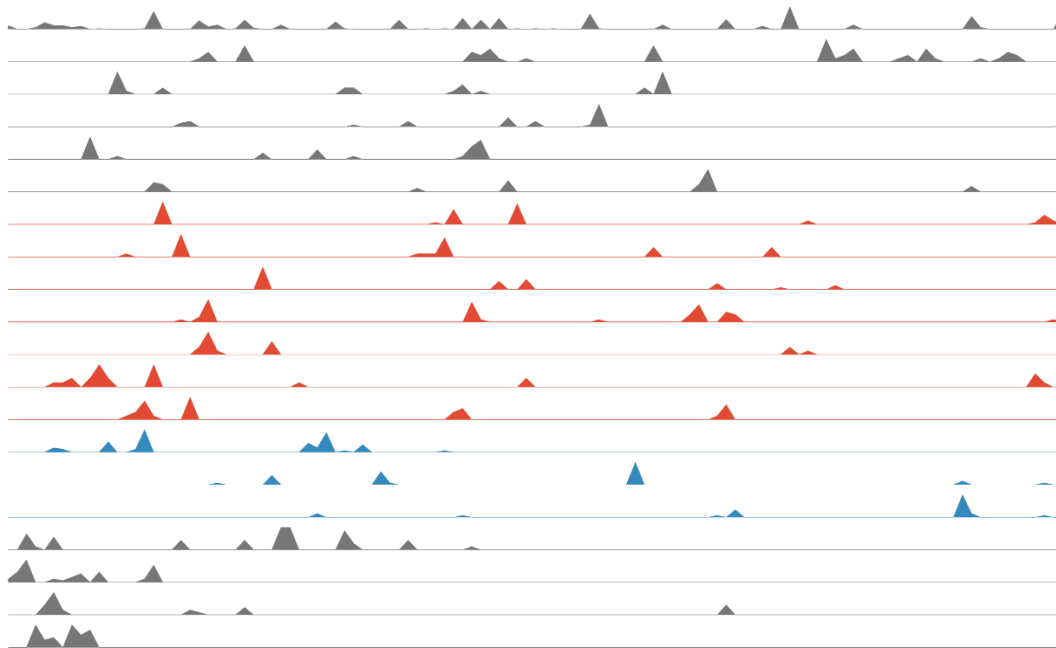


Рис. 7. Коммиты нескольких студентов после использования указанных принципов

## Заключение

В данной работе были предложены принципы и подходы к написанию тестовых систем проверки работ для курсов по программированию на основе экспериментальных данных курса по параллельному программированию. Также были введены идеи жизнеспособности систем для проверки работ, введен новый критерий оценивания, по которому можно изменять систему таким образом, чтобы повысить качество работы студента и качество проверки его работ. Была представлена система `parallel_programming_course` и ее особенности.

Далее предполагается провести унификацию API для перехода к следующему этапу автоматизации, а также добавить новые анализаторы качества исходных кодов студентов для сохранения и усиления представленных выше принципов. Также в будущем планируется реализовать качественную проверку ускорения задач и устраивать на основе данной проверки соревнования среди студентов.

## Литература

1. Гергель, В.П., Стронгин, Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. - Н.Новгород, ННГУ (2 изд., 2003), 2001.
2. Воеводин В. В., Воеводин В. В. Параллельные вычисления. СПб., БХВ. – 2002.
3. Корняков К. В., Кустикова В. Д., Мееров И. Б., Сиднев А. А., Сысоев А. В., Шишков А. В. Инструменты параллельного программирования в системах с общей памятью: учебник / под ред. проф. В. П. Гергеля. — М.: Издательство Московского университета, 2010. 272 с.
4. Гергель В.П. Теория и практика параллельных вычислений. – М.: Интернет-Университет Информационных технологий; Бином. Лаборатория Знаний, 2007.
5. Основная страница стандарта MPI. URL: <https://www.mcs.anl.gov/research/projects/mpi/> (дата обращения: 29.03.2021).
6. Основная страница стандарта OpenMP. URL: <https://www.openmp.org/> (дата обращения: 29.03.2021).



7. Voss M., Asenjo R., Reinders J. Pro TBB: C++ parallel programming with threading building blocks. – Apress, 2019. 820p.
8. Основная страница описания класса thread.  
URL: <https://en.cppreference.com/w/cpp/thread/thread> (дата обращения: 29.03.2021).
9. Гергель В.П., Мееров И.Б., Сысоев А.В. и др. Высокопроизводительные параллельные вычисления. 100 заданий для расширенного лабораторного практикума. – «Физматлит. МАИК-Наука», 2018.
10. Исходный код проекта parallel\_programming\_course. URL: [https://github.com/learning-process/parallel\\_programming\\_course](https://github.com/learning-process/parallel_programming_course) (дата обращения: 01.04.2021).
11. Основная страница технологии контроля версий кода git. URL: <https://git-scm.com/> (дата обращения: 05.04.2021).
12. Основная страница технологии кроссплатформенной сборки CMake. URL: <https://cmake.org/> (дата обращения: 10.04.2021).
13. Основная страница системы непрерывной интеграции Travis. URL: <https://travis-ci.com/> (дата обращения: 10.04.2021).
14. Основная страница системы непрерывной интеграции AppVeyor.  
URL: <https://www.appveyor.com/> (дата обращения: 10.04.2021).
15. Исходные коды тестовой системы Google Test Framework.  
URL: <https://github.com/google/googletest> (дата обращения: 11.04.2021).
16. Исходные коды системы проверки программного обеспечения на антиплагиат. URL: <https://github.com/jplag/jplag> (дата обращения: 11.04.2021).
17. Стиль кодирования Google Style Code C++.  
URL: <https://google.github.io/styleguide/cppguide.html> (дата обращения: 11.04.2021).
18. Основная страница инструмента анализа памяти Valgrind. URL: <https://valgrind.org/> (дата обращения: 11.04.2021).
19. Основная страница инструмента анализа памяти CppCheck.  
URL: <http://cppcheck.sourceforge.net/> (дата обращения: 11.04.2021).

## Валидация пакета программ «ЛОГОС» для решения задач судостроительной отрасли

А.С. Козелков<sup>1</sup>, В.В. Курулин<sup>1</sup>, А.Е. Таранов<sup>2</sup>, С.В. Лашкин<sup>1</sup>, К.С. Плыгунова<sup>1</sup>,  
О.Л. Крутякова<sup>1</sup>

<sup>1</sup>ФГУП «Российский Федеральный Ядерный Центр «Всероссийский научно-исследовательский институт экспериментальной физики»,  
<sup>2</sup>ФГУП «Крыловский государственный научный центр»

В работе рассматривается решение задач корабельной гидродинамики с помощью отечественного пакета программ ЛОГОС на основе численного решения трехмерных уравнений Навье-Стокса. Рассматривается решение задач двух классов: вращение гребных винтов и движение корпусов судов в потоке вязкой жидкости. По каждой задаче приводится описание постановки, имеющиеся опорные данные, а также результаты численного моделирования задач, полученные с помощью пакета программ ЛОГОС.

*Ключевые слова:* вычислительная гидродинамика, уравнения Навье-Стокса, свободная поверхность, volume of fluid, гребной винт, динамика судна, генерация волн, валидация

### 1. Введение

С развитием технологий возможности вычислительной гидродинамики достигли значительного прогресса, как следствие, она находит все большее применение в качестве альтернативы натурному эксперименту, который часто является сложным и дорогостоящим мероприятием. Это позволяет эффективно решать сложные прикладные задачи, в том числе задачи судостроения – обтекание тел потоком жидкости, моделирование вращения гребных винтов, плавание тел на водной поверхности с учетом ветро-волновых нагрузок, моделирование маневрирования судов с учетом вращения движительно-рулевых комплексов, моделирование процессов кавитации и т.д.

Проблемам корабельной гидродинамики посвящены международные симпозиумы, такие как SIMMAN [1], CFD in Ship Hydrodynamics [2], участники которых представляют решения известных тестов в различных расчетных кодах: ANSYS Fluent, STAR-CCM+, OpenFOAM и многих других. Одной из основных рассматриваемых задач является определение коэффициента сопротивления судна. Южно-Корейским научно-исследовательским институтом кораблестроения и океанической инженерии (MOERI, ранее KRISO) были разработаны модели контейнеровоза KCS (KRISO container ship) и танкера KVLCC2, для которых в опытовых бассейнах получены экспериментальные данные специально для верификации численных методов. Существует и много других корпусов с различной геометрией, для которых доступны данные по сопротивлению: модели DTMB 5415, Wigley, S175, HSVA и другие [3]. Вторая важная задача в судостроении – это определение кривых действия гребных винтов. Известной тестовой моделью является модель винта VP1304, для которой получены экспериментальные гидродинамические характеристики – Potsdam Propeller Test Case (PPTC) [4]. Еще одна модель гребного винта KP505 спроектирована институтом MOERI для контейнеровоза KCS, для нее также доступны экспериментальные кривые действия в открытой воде.

Решение подобных задач требует применения точных физико-математических моделей и численных алгоритмов, специализированных сеточных генераторов, а также современных средств распараллеливания и постобработки. Одним из пакетов программ, который удовлетворяет данным требованиям, является ЛОГОС [5-6] – российское CAE-решение, предназначенное для моделирования сопряженных трехмерных задач конвективного теплопереноса, аэродинамики, гидродинамики и прочности на высокопараллельных

ЭВМ. Пакет программ ЛОГОС успешно прошел верификацию и показал достаточно хорошие результаты на серии различных гидродинамических задач [7-8], промышленных задач [6, 9], включая задачи со свободной поверхностью [10-11]. В настоящей статье рассматривается применение пакета программ ЛОГОС для решения типичных задач судостроения – вращения гребных винтов и обтекания корпусов судов потоком вязкой жидкости. Решение задач данных классов основано на численном решении трехмерных уравнений Навье-Стокса, ядром численного алгоритма является итеративный SIMPLE-подобный метод [5, 12-14] с конечно-объемным методом дискретизации уравнений. Учет движения границ осуществляется путем использования технологии деформирующихся и перекрывающихся сеток [15-16], а также динамической сшивки сеток на скользящих интерфейсах [17]. В докладе представлены следующие задачи: вращение моделей корабельных винтов КР505 и 8029 в открытой воде [18-19], буксировка и качка на встречном волнении модели контейнеровоза KCS (KRISO container ship) [20]. Крыловским государственным научным центром ранее выполнялся расчет вышеперечисленных задач в коммерческом программном комплексе Star-CCM+ [19-20]. Результаты численного моделирования в пакете программ ЛОГОС сравниваются с результатами расчетов в пакете STAR-CCM+ и экспериментальными данными по каждой задаче.

## 2. Математическая модель

Для описания движения вязкой сжимаемой жидкости используются трехмерные уравнения Навье-Стокса, осредненные по Рейнольдсу. Предполагается, что течение изотермическое, а поля скорости и давления общие для всех фаз. Учитывая принятые допущения и применяя метод VOF для определения положения свободной поверхности, можно записать систему уравнений в декартовой системе координат, которая состоит из уравнения сохранения массы, уравнения сохранения импульса и уравнения переноса объемной доли:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0, \\ \frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\tau_{ij} + \tau'_{ij}) + \rho g_i \\ \frac{\partial \rho_\xi \alpha_\xi}{\partial t} + \frac{\partial}{\partial x_i}(u_i \rho_\xi \alpha_\xi) = 0 \end{cases} \quad (1)$$

где  $t$  – время,  $u_i = \{u_1, u_2, u_3\} = \{u, v, w\}$  – скорость,  $x_i$  – компонент пространственного вектора,  $\tau_{ij}$  – тензор вязких напряжений,  $g_i$  – вектор ускорения свободного падения,  $\xi$  – индекс, указывающий на принадлежность к отдельной фазе,  $\alpha_\xi$  – объемная доля  $\xi$ -й фазы,  $\rho$  – результирующая плотность, представляющая собой усредненное значение плотности по всем фазам:

$$\rho = \sum_{\xi=1}^N \rho_\xi \alpha_\xi, \quad (2)$$

где  $N$  – количество фаз.

Тензор вязких напряжений, который, согласно гипотезе Буссинеска, записывается в виде [12]:

$$\tau_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right), \quad (3)$$

где  $\mu$  – результирующая динамическая вязкость,  $\delta_{ij}$  – символ Кронекера.

Эмпирические соотношения для турбулентной вязкости  $\mu_t$  и тензора напряжений:

$$\tau'_{ij} = 2\mu_t \left( S_{ij} - \frac{1}{3} I_{ij} \nabla \cdot \vec{u} \right) + \frac{2}{3} k I_{ij}, \quad S_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad (4)$$

где  $k$  – кинетическая энергия турбулентности.

Система уравнений (1) замыкается моделью турбулентности  $k - \omega$  SST Ментера [21] совместно с моделью ламинарно-турбулентного перехода  $\gamma - \text{Re}_\theta$  (Gamma Re Theta) [22].

Для учета движения границ используется технологии деформирующихся и перекрывающихся сеток. Вычисление перемещения узлов осуществляется методом IDW [15]. Учёт движения сетки в системе (1) осуществляется путём перехода в уравнениях переноса объёмной доли фаз и импульса к подвижной системе координат по известному закону [23]:

$$\frac{d^* \varphi}{dt} = \frac{\partial \varphi}{\partial t} + v_i \frac{\partial \varphi}{\partial x_i} \quad (5)$$

где  $\frac{d^* \varphi}{dt}$  – субстанциональная производная переносимого скаляра  $\varphi$  относительно подвижной системы координат,  $v_i$  – вектор скорости перемещения сетки. С использованием (5) уравнение переноса объёмной доли можно записать следующим образом:

$$\frac{d^* \alpha_\xi}{dt} + (u_i - v_i) \frac{\partial \alpha_\xi}{\partial x_i} + \alpha_\xi \frac{\partial u_i}{\partial x_i} = - \frac{\alpha_\xi}{\rho_\xi} \left[ \frac{d^* \rho_\xi}{dt} + (u_i - v_i) \frac{\partial \rho_\xi}{\partial x_i} \right], \quad (6)$$

где  $\frac{d^* \alpha_\xi}{dt}$  – субстанциональная производная на движущейся сетке.

Уравнение сохранения импульса также формулируется относительно подвижной системы координат с учётом (5):

$$\rho \frac{d^* u_i}{dt} + \frac{\partial}{\partial x_j} (\rho u_i (u_j - v_j)) - u_i \frac{\partial}{\partial x_j} (\rho (u_j - v_j)) = - \frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i. \quad (7)$$

Уравнение сохранения массы записывается относительно скорости в подвижной системе координат в следующем виде:

$$\frac{\partial (u_i - v_i)}{\partial x_i} = - \sum_{\xi} \frac{\alpha_\xi}{\rho_\xi} \left[ \frac{d^* \rho_\xi}{dt} + (u_i - v_i) \frac{\partial \rho_\xi}{\partial x_i} \right]. \quad (8)$$

Таким образом, конечная система уравнений (1) принимает следующий вид:

$$\begin{cases} \frac{\partial (u_i - v_i)}{\partial x_i} = - \sum_{\xi} \frac{\alpha_\xi}{\rho_\xi} \left[ \frac{d^* \rho_\xi}{dt} + (u_i - v_i) \frac{\partial \rho_\xi}{\partial x_i} \right] \\ \rho \frac{d^* u_i}{dt} + \frac{\partial}{\partial x_j} (\rho u_i (u_j - v_j)) - u_i \frac{\partial}{\partial x_j} (\rho (u_j - v_j)) = - \frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i \\ \frac{d^* \alpha_\xi}{dt} + (u_i - v_i) \frac{\partial \alpha_\xi}{\partial x_i} + \alpha_\xi \frac{\partial u_i}{\partial x_i} = - \frac{\alpha_\xi}{\rho_\xi} \left[ \frac{d^* \rho_\xi}{dt} + (u_i - v_i) \frac{\partial \rho_\xi}{\partial x_i} \right]. \end{cases} \quad (9)$$

Дискретизация уравнений системы (9) осуществляется на основе метода конечных объёмов с учётом неструктурированности расчётной сетки. Решение системы уравнений (9) основано на SIMPLE-подобном алгоритме [5, 12-14], в котором поля скорости и давления находятся по схеме предиктор-корректор и решением систем линейных алгебраических уравнений (СЛАУ). Для учета силы тяжести используется алгоритм, основанный на поправке объёмных сил [8], обеспечивающий отсутствие осцилляций, связанных с неколокированным размещением неизвестных величин, на сетках любого типа. Ускорение вычислений для проведения расчетов на высокопроизводительных супер-ЭВМ осуществляется многосеточным методом [6].

### 3. Валидационные задачи

К первому классу рассматриваемых задач относится моделирование вращения гребных винтов. Проводятся нестационарные расчеты обтекания однородным потоком жидкости

моделей пятилопастных гребных винтов КР505 и 8029. Винты фиксируются на конце вала, диаметр модели КР505 составляет 0.25 м, модели 8029 – 0.228 м. Геометрия гребных винтов показана на рисунке 1.

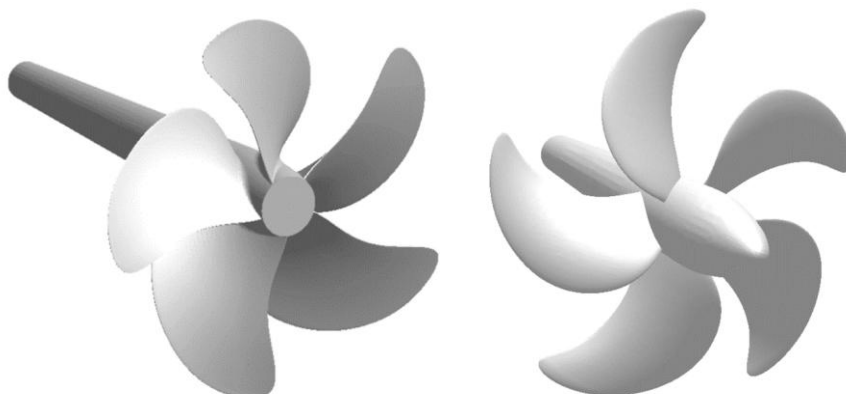


Рис. 1. Модели гребных винтов КР505 (слева) и 8029 (справа)

Для численного определения кривых действия гребных винтов проводится серия параметрических расчетов при частоте вращения  $n = 20$  об/с. Относительная поступь  $J$  в этих расчетах изменяется в интервале 0.1 – 0.9 с шагом 0.1, из соответствующей поступи вычисляется скорость набегающего потока по формуле:

$$V_m = J \cdot n \cdot D_m. \quad (10)$$

Соответственно, диапазон рассматриваемых скоростей набегающего потока составляет 1 ÷ 4.5 м/с для гребного винта КР505 и 0.91 ÷ 4.11 м/с для винта 8029.

Для численного моделирования используется модель турбулентности  $k - \omega$  SST и модель ламинарно-турбулентного перехода  $\gamma - Re_\theta$ . Вращение винта моделируется движением узлов расчетной сетки с использованием внутреннего интерфейса между вращающимся и статическим регионами расчетной области. Вращающийся регион представляет собой цилиндрическую область, окружающую винт. Расчетные сетки построены со сгущениями ячеек вблизи лопастей, по ходу движения закрученного потока, а также в пограничном слое (рисунки 2 и 3). Сетка для модели гребного винта КР505 содержит 15 млн ячеек, базовый размер ячейки равен 0.08 м, измельчение в контрольной области составляет 1.5% от базового размера. Расчетная сетка для винта 8029 состоит из 5.5 млн ячеек, базовый размер – 0.04 м, измельчение в контрольной области составляет 3% от базового размера.

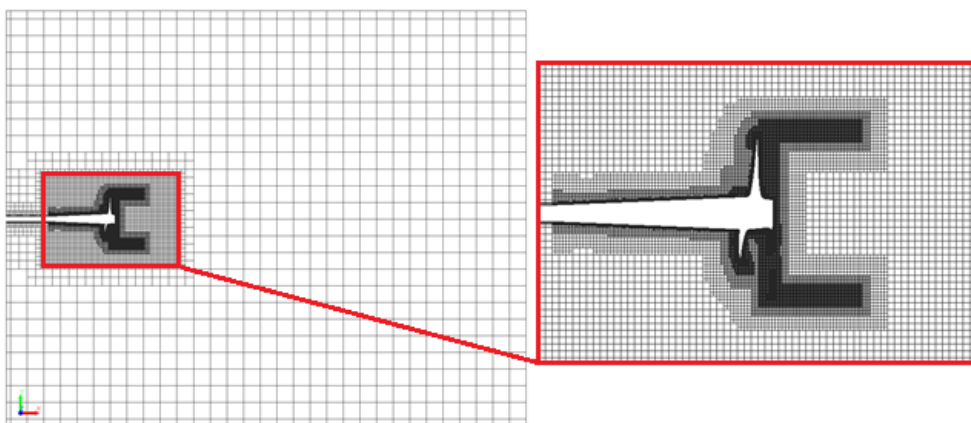


Рис. 2. Расчетная сетка для модели гребного винта КР505

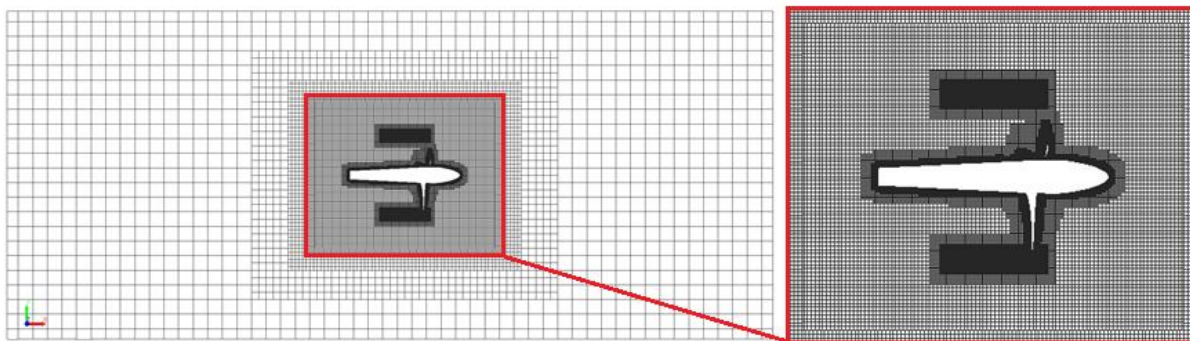


Рис. 3. Расчетная сетка для модели гребного винта 8029

На входной границе задана скорость набегающего потока. На выходной границе задано давление. На стенках винтов задана стенка без проскальзывания. На боковых границах внешней области задано условие симметрии.

Выполняется нестационарный расчет с шагом по времени  $\Delta t = 2 \cdot 10^{-4}$  с, для дискретизации конвективных слагаемых используется схема второго порядка LUD [13], по времени – трехслойная схема [24]. В расчетах были приняты следующие значения физических свойств воды: динамическая вязкость  $\mu = 0.00114$  Па·с, плотность  $\rho = 1000$  кг/м<sup>3</sup>.

Расчет производился на вычислительном кластере ФГУП «РФЯЦ-ВНИИЭФ», использовалось 100 расчетных ядер, физическое время расчета составляет от 12 до 24 часов в зависимости от рассматриваемого режима.

На рисунке 4 представлено мгновенное поле скорости на момент времени  $t = 1$  с (режим  $J = 0.4$ ).

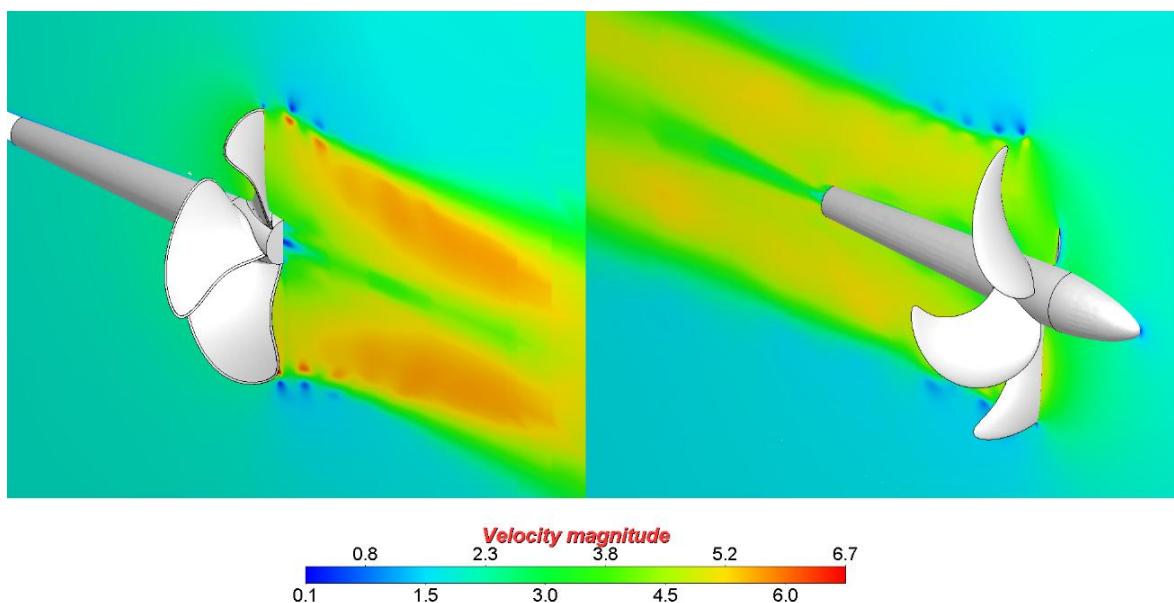


Рис. 4. Мгновенное поле скорости: гребной винт KP505 (слева) и гребной винт 8029 (справа)

На рисунке 5 приведено изображение изоповерхности, построенной по Q-критерию для гребных винтов KP505 и 8029 (режим  $J = 0.4$ ).



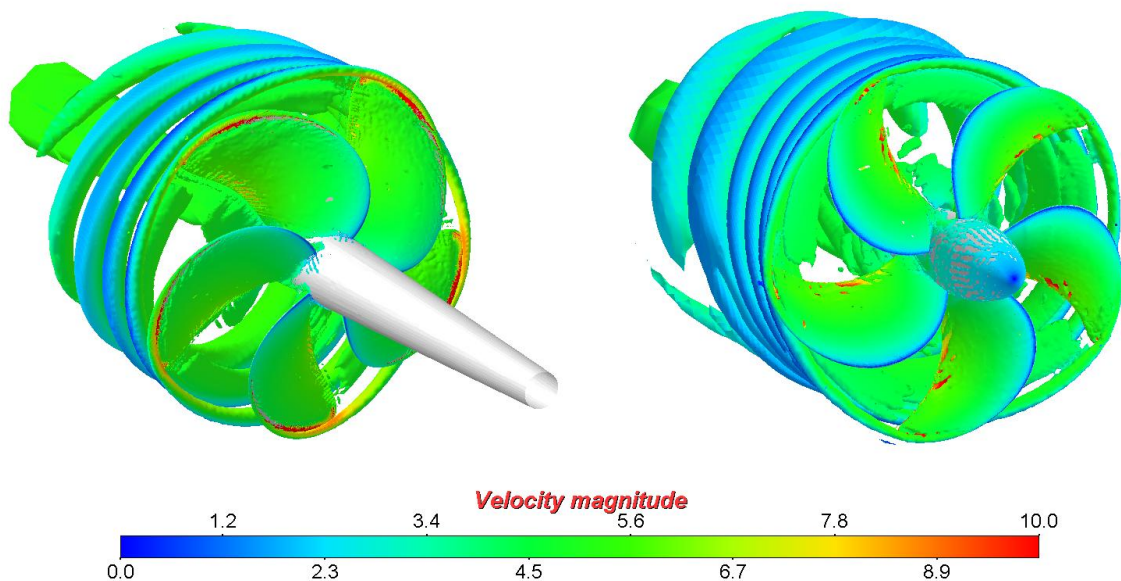


Рис. 5. Исоповерхность Q-критерия для модели винта KP505 (слева) и 8029 (справа)

Сравнение результатов численного моделирования проводится по характеристикам гребного винта: коэффициенту упора  $K_T = \frac{T}{\rho \cdot n^2 \cdot D^4}$ , коэффициенту момента  $K_Q = \frac{Q}{\rho \cdot n^2 \cdot D^5}$ , коэффициенту полезного действия  $\eta = \frac{J}{2\pi} \cdot \frac{K_T}{K_Q}$ , где  $T$  – сила тяги,  $Q$  – момент силы тяги,  $\rho$  – плотность среды,  $n$  – скорость вращения винта,  $D$  – диаметр винта. Данные характеристики в нестационарном расчете осредняются по времени после момента статистического установления.

На рисунке 6 показано сравнение кривых действия гребных винтов KP505 и 8029 с расчетом в STAR-CCM+ и экспериментальными данными [18, 19].

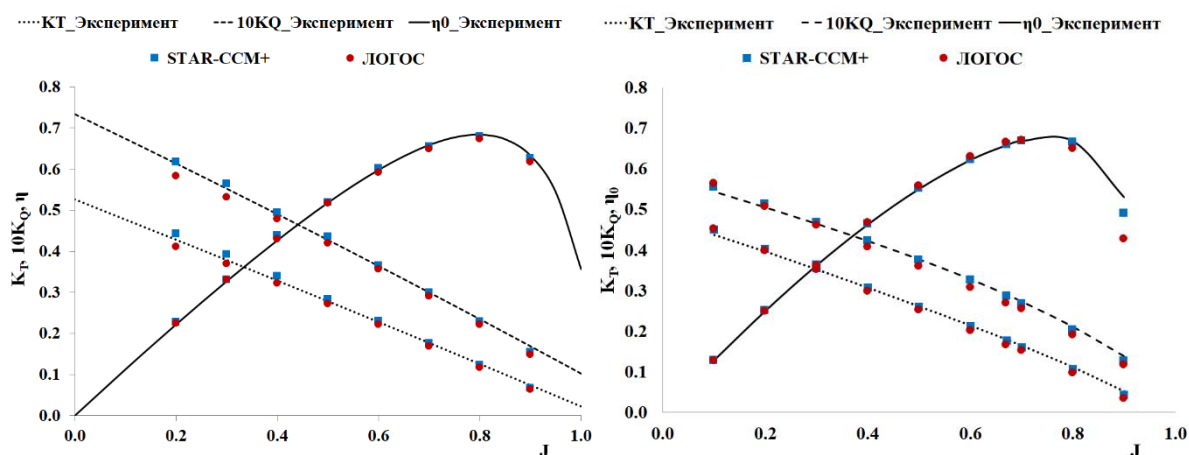


Рис. 6. Кривые действия гребных винтов KP505 (слева) и 8029(справа)

Как видно из рисунка 6 результаты численного моделирования в пакете программ ЛОГОС хорошо согласуются с экспериментальными данными и результатами расчета в STAR-CCM+. Погрешность в предсказании коэффициентов момента и упора принимает наибольшие значения на больших поступях, следовательно, возрастает погрешность при вычислении коэффициента полезного действия, что особенно заметно для модели винта 8029. Отклонения полученных значений характеристик гребных винтов при значениях поступи меньше 0.5

составляют менее 5%. Максимальное отклонение наблюдается в вычислении коэффициента упора при значении поступи нулевого упора и составляет порядка 15%.

Следующий класс задач – это обтекание корпусов судов потоком вязкой жидкости. Рассматривается динамика модели контейнеровоза KCS в условиях буксировки и на встречном волнении [20].

В задаче определения буксировочного сопротивления и посадки судна используется модель KCS в масштабе 1:31.6. Контейнеровоз имеет две степени свободы – вертикальное перемещение и дифферент. Расчетная сетка, содержащая около 3 млн ячеек, имеет призматический слой у поверхности судна, а также сгущение на разделе фаз и в областях носа и кормы (рисунок 7). Толщина первого призматического слоя у корпуса 0.0025 м. Длина расчетного бассейна 180 м, ширина – 100 м, высота – 48 м.

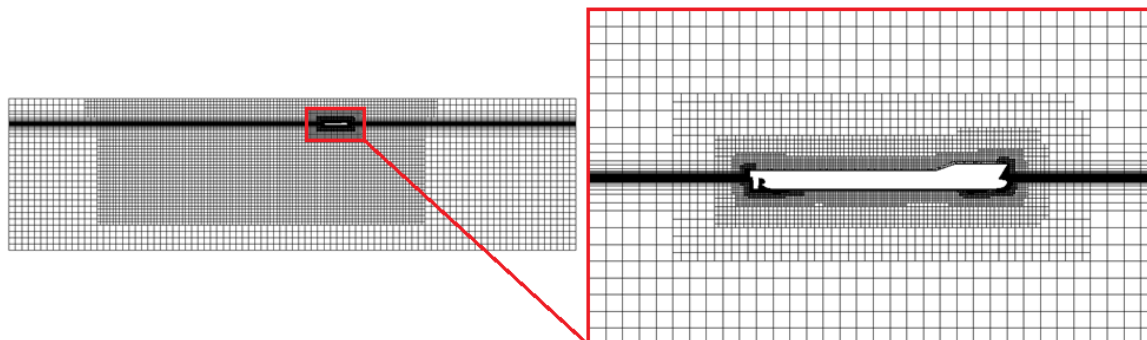


Рис. 7. Расчетная сетка для буксировки KCS

На входной границе задается скорость потока, на выходной и верхней границах фиксируется давление, на боковых и нижней границах задается условие непротекания без прилипания потока, а на корпусе судна – с прилипанием потока.

Масса судна составляет 1637 кг, диагональные моменты инерции (264; 5460; 5460) кг·м<sup>2</sup>. Координаты центра тяжести (3.532; 0; -0.2) м, осадка на нос и корму 0.3418 м.

Движение тела осуществляется с помощью деформации расчетной сетки. Задача решается в нестационарной постановке, со вторым порядком по времени. Временной шаг равен  $\Delta t = 0.002$  с. В уравнении объемной доли для дискретизации конвективных слагаемых используется схема HRIC, в уравнении движения – противопоточная схема второго порядка LUD.

Расчет проводится для 6 различных режимов течения с различными скоростями набегающего потока: 0.915 м/с, 1.281 м/с, 1.647 м/с, 1.922 м/с, 2.196 м/с, 2.379 м/с.

Приняты следующие значения физических свойств: динамическая вязкость воды  $\mu_e = 0.001269$  Па·с, плотность воды  $\rho_e = 999.5$  кг/м<sup>3</sup>, динамическая вязкость воздуха  $\mu_{возд} = 1.85508 \cdot 10^{-5}$  Па·с, плотность воздуха  $\rho_{возд} = 1.18415$  кг/м<sup>3</sup>.

Расчет 80 секунд для каждого режима выполняется на 160 процессорах, физическое время расчета составляет 62 часа.

На рисунке 8 показана картина распространения волн вокруг корпуса контейнеровоза для скорости набегающего потока 2.196 м/с.

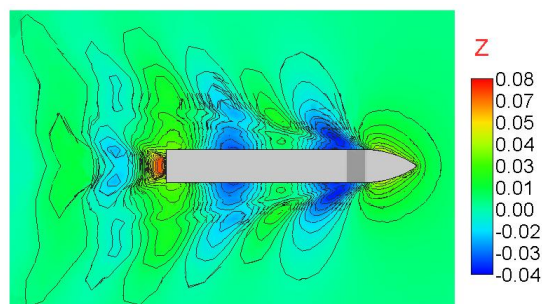


Рис. 8. Волновая картина при буксировке KCS



На рисунках 9-11 представлено сравнение зависимостей коэффициента буксировочного сопротивления  $C_T$ , всплытия судна  $z$  и угла дифферента  $\theta$  от числа Фруда, полученных в пакете программ ЛОГОС, с результатами численного моделирования в STAR-CCM+ и экспериментальными данными [0].

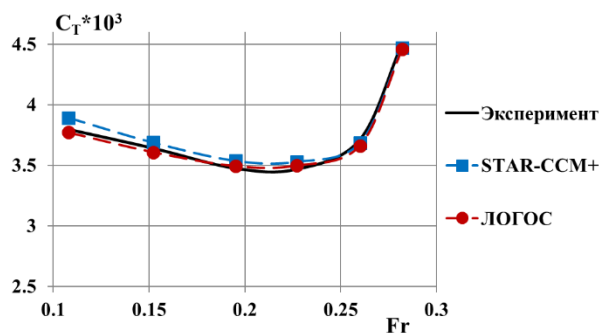


Рис. 9. Коэффициент сопротивления буксируемой модели контейнеровоза

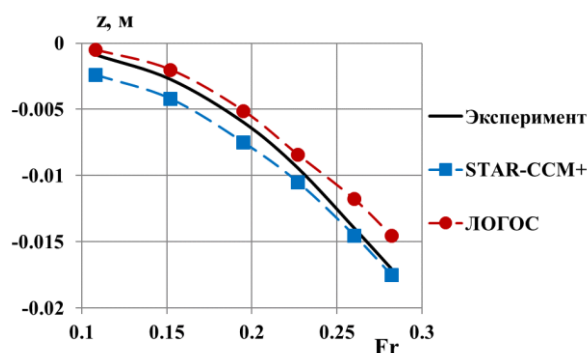


Рис. 10. Всплытие буксируемой модели контейнеровоза

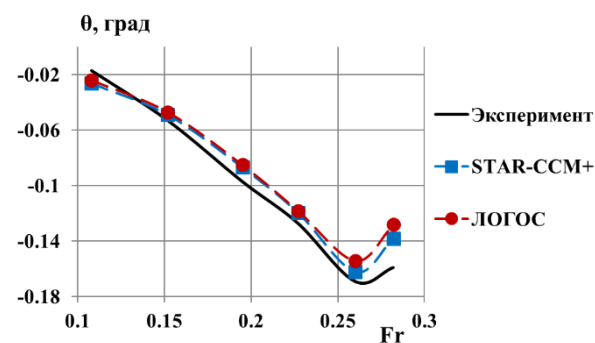
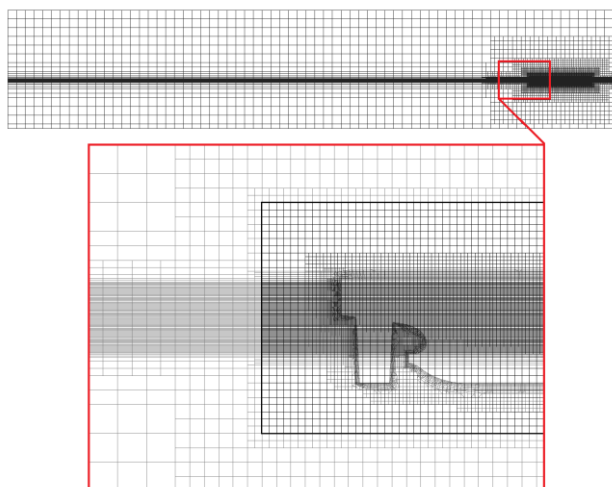


Рис. 11. Угол дифферента буксируемой модели контейнеровоза

Из рисунков 9-11 видно, что зависимости контрольных величин, полученные в двух пакетах программ, в целом, хорошо согласуются с экспериментом, для коэффициента сопротивления и угла дифферента результаты численного моделирования коррелируют между собой. Результаты расчета в пакете программ ЛОГОС для величины всплытия зеркально отличаются от результатов численного моделирования в STAR-CCM+ относительно экспериментальных данных. Отклонения значения коэффициента сопротивления от эксперимента не превышают 1%. Для величины всплытия и угла дифферента максимальное отклонение составляет около 16%.

Следующая рассматриваемая задача данного класса – качка контейнеровоза KCS на волнении. Модель судна выполнена в масштабе 1:37.9 и, как в предыдущей задаче, имеет две степени свободы – вертикальное перемещение и дифферент. Используются перекрывающиеся расчетные сетки, которые содержат ~13 млн ячеек, толщина ячейки у поверхности раздела

составляет 0.004 м (рисунок 12). Длина расчетного бассейна 70 м, ширина – 13.5 м, высота – 12 м. Судно расположено на расстоянии 3.7 м от входа.



**Рис. 12.** Фрагмент перекрывающихся сеток в области кормы

На входной и боковых границах задается встречное волнение, на верхней и выходной границах задается фиксированное давление, на нижнюю границу накладывается условие непротекания без прилипания потока, на корпус судна – условие непротекания с прилипанием потока. Скорость набегающего потока задана равной 2.017 м/с. Волнение поверхности задано с помощью генератора поверхностных волн, используется волна Стокса пятого порядка.

Масса судна составляет 955.78 кг, диагональные моменты инерции (110; 2222; 2222) кг·м<sup>2</sup>, Координаты центра тяжести (2.9453; 0; 0.093) м.

Задача решается в нестационарной постановке с первым порядком по времени. Временной шаг равен  $\Delta t = 0.004$  с. В уравнении объемной доли для дискретизации конвективных слагаемых используется схема HRIC, в уравнении движения – схема LUD.

Расчет проводится для 5 режимов течения с различными параметрами волны, которые приведены в таблице 1.

**Таблица 1.** Параметры режимов расчета

Режим	Длина волны $h_w$ , м	Высота волны $\lambda$ , м
C1	3.949	0.062
C2	5.164	0.078
C3	9.979	0.123
C4	8.321	0.149
C5	11.840	0.196

Расчет 60 секунд для каждого режима выполняется на 160 процессорах, физическое время расчета составляет 138 часов.

На рисунке 13 показана волновая картина вокруг корпуса KCS для режима C4. Из рисунка видно характерное распределение волн в следе за судном, а также заданное волнение водной поверхности.

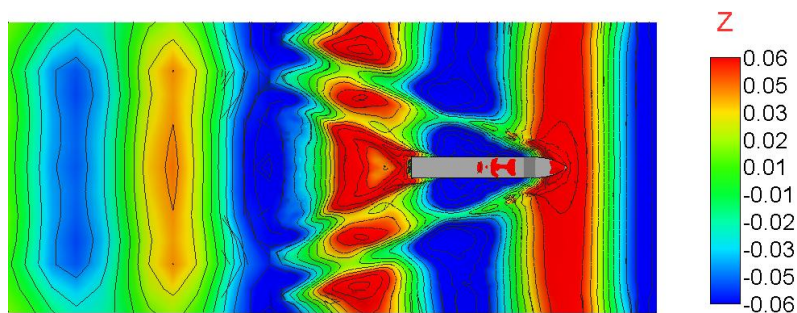


Рис. 13. Волновая картина (режим С4)

В качестве сравнительной характеристики рассматриваются коэффициент сопротивления и угол дифферента. Коэффициент сопротивления рассчитывается по формуле:

$$C_T = F_t / (1/2 \rho_0 U^2 S), \quad (11)$$

где  $F_t$  – сила, действующая на тело,  $U$  – скорость потока,  $S$  – площадь смоченной поверхности.

Величина всплытия модели судна обезразмерена на половину высоты волны  $\zeta = h_w / 2$ , величина угла дифферента обезразмерена на половину высоты волны и волновое число  $k = 2\pi / \lambda$ .

На рисунках 14 и 15 представлено сравнение с результатами численного моделирования в STAR-CCM+ и экспериментальными данными [20] полученных в пакете ЛОГОС зависимостей коэффициента сопротивления и обезразмеренного угла дифферента от относительной длины волны.

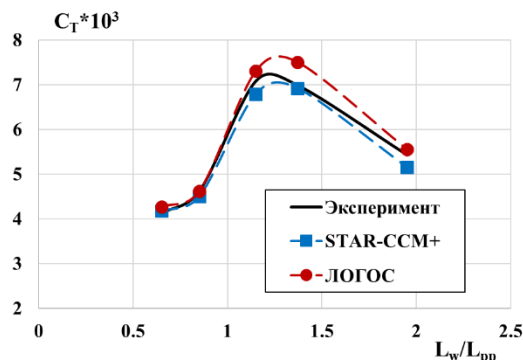


Рис. 14. Коэффициент сопротивления модели KCS на волнении

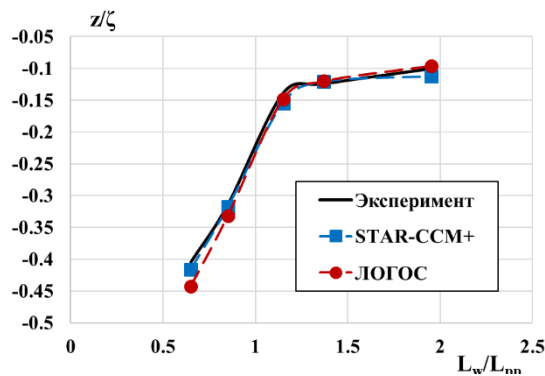


Рис. 15. Обезразмеренное всплытие модели KCS на волнении

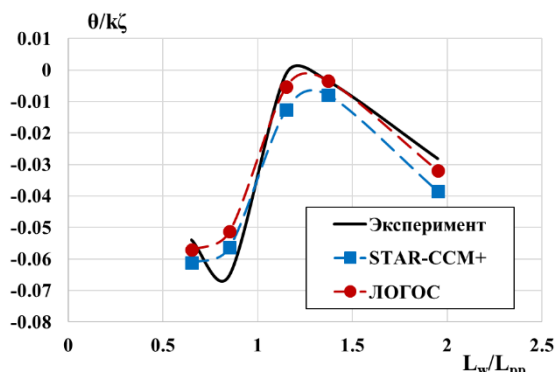


Рис. 16. Безразмерный угол дифферента модели KCS на волнении

Полученные результаты, в целом, хорошо согласуются с экспериментом. Безразмерное всплытие, рассчитанное в двух пакетах программ, достаточно близко к экспериментальной зависимости. Коэффициент сопротивления в промежуточных режимах точнее определяется в пакете STAR-CCM+, безразмерный угол дифферента для последних трех режимов – в пакете ЛОГОС. Максимальное отклонение коэффициента сопротивления наблюдается для режима С4 и составляет 6%, для остальных режимов отклонение не превышает 2.5%. Достаточно большое отклонение угла дифферента можно объяснить очень малыми значениями этой величины, полученными в эксперименте – 0.004-0.012°.

#### 4. Заключение

В настоящей работе представлены результаты валидации пакета программ ЛОГОС для решения задач судостроения, а именно двух классов задач – вращения гребных винтов, обтекания корпусов судов потоком вязкой жидкости. Представлены основные уравнения и подходы к их численному решению, используемыми в пакете программ ЛОГОС. Полученные результаты по моделированию гребных винтов показывают, что удается получить достаточно точный результат по КПД (отклонение до 5%), для коэффициентов момента и упора максимальное отклонение уже превышает 5%. Для задач движения судов в потоке вязкой жидкости максимальное отклонение расчетного коэффициента сопротивления составляет 6% в случае качки на волнении, а в случае буксировки модели KCS – не превышает 1%.

Сравнение результатов численного моделирования в пакете программ ЛОГОС с результатами расчета в STAR-CCM+ показывает, что полученные в этих двух пакетах зависимости контрольных величин коррелируют между собой, максимальные отклонения этих величин имеют один и тот же порядок.

Данные результаты показывают, что пакет программ ЛОГОС позволяет моделировать основные процессы корабельной гидродинамики с точностью, которая является удовлетворительной для задач судостроения. Постоянное развитие и усовершенствование численных методов помогут в дальнейшем улучшить полученные результаты и снизить погрешность математического моделирования.

#### Литература

1. SIMMAN Workshop 2014. URL: <https://simman2014.dk> (дата обращения: 01.06.2020).
2. Workshop on CFD in Ship Hydrodynamics 2015, Tokyo. URL: <https://t2015.nmri.go.jp> (дата обращения: 01.06.2020).
3. Specialist Committee on CFD in Marine Hydrodynamics: Final report and Recommendations to the 27th ITTC, Copenhagen. URL: <https://itc.info/media/6097/sc-cfd.pdf> (дата обращения: 05.12.2020).
4. Propeller performance Potsdam Propeller Test Case (PPTC). Open Water Test with the Model Propeller VP1304 // Report for the 2nd International Symposium on Marine Propulsors 2011 Workshop, May 2011, Potsdam. Parts 1-3.

5. Lashkin S.V., Kozelkov A.S., Yalozo A.V., Gerasimov V.Y., Zelensky D.K. Efficiency analysis of the parallel implementation of the SIMPLE algorithm on multiprocessor computers // Journal Of Applied Mechanics And Technical Physics. 2017. Vol. 58, No. 7, P. 1242-1259.
6. Kozelkov A.S., Kurulin V.V., Lashkin S.V., Shagaliev R.M., Yalozo A.V. Investigation of supercomputer capabilities for the scalable numerical simulation of computational fluid dynamics problems in industrial applications // Computational mathematics and mathematical physics. 2016. Vol. 56, No. 8. P. 1524–1535.
7. Kozelkov A.S., Krutyakova O.L., Kurulin V.V., Tyatyushkina E.S., Kurkin A.A. Zonal RANS–LES approach based on an algebraic Reynolds stress model // Fluid Dynamics. 2015. Vol. 50, No. 5. P. 24-33.
8. Efremov V. R., Kozelkov A. S., Kornev A.V., Kurkin A. A., Kurulin V. V., Strelets D. Yu., Tarasova N. V. Method for taking into account gravity in free-surface flow simulation // Computational Mathematics and Mathematical Physics. 2017. Vol. 57, No. 10. P. 1720-1733.
9. Betelin V.B., Shagaliev R.M., Aksenov S.V., Belyakov I.M., Deryugin Yu.N., Kozelkov A.S., Korchazhkin D.A., Nikitin V.F., Sarazov A.V., Zelenskiy D.K. Mathematical simulation of hydrogen–oxygen combustion in rocket engines using LOGOS code // Acta Astronautica. 2014. Vol. 96. P. 53–64.
10. Kozelkov A.S. Numerical technique for the landslide tsunami simulations based on Navier–Stokes equations // Fluid Dynamics. 2016. Vol. 9, No. 2. P. 218-236.
11. Kozelkov A.S., Kurkin A.A., Pelinovskii E.N., Kurulin V.V., Tyatyushkina E.S. Modeling the disturbances in the lake Chebarkul caused by the fall of the meteorite in 2013 // Fluid Dynamics. 2015. Vol. 50, No 6. No. 828-840.
12. Флетчер К. Вычислительные методы в динамике жидкостей в двух томах. Том 2. Москва: Издательство «Мир», 1991. 552 с.
13. Jasak H. Error Analysis and Estimation for the finite volume method with applications to fluid flows. Thesis submitted for the degree of doctor // Department of Mechanical Engineering. Imperial College of Science. 1996.
14. Ferziger J.H., Peric M. Computational methods for fluid dynamics. Berlin, Heidelberg: Springer, 2001.
15. Luke E., Collins E., Blades E. A fast mesh deformation method using explicit interpolation // Journal of Computational Physics. 2012. No. 231. P. 586–601.
16. Benek J. A., Buning P. G., Steger J. L. A 3-D Chimera Grid Embedding Technique // AIAA Paper. 1985. No. 85-1523.
17. Beaudoin M., Jasak H. Development of Generalized Grid Interface for Turbomachinery simulation with OpenFOAM // Open Source CFD International Conference: Berlin, 2008.
18. Paik K.J. Numerical study on the hydrodynamic characteristics of a propeller operating beneath a free surface // International Journal of Naval Architecture and Ocean Engineering. 2017. Vol. 6, No. 9. P. 655–667.
19. Панов Д.О., Смирнов Е.М., Таранов А.Е., Лобачев М.П. Моделирование ламинарно-турбулентного перехода в задаче численного определения кривых действия гребного винта // Труды ЦНИИ им. акад. А.Н. Крылова. 2013. С. 29–42.
20. Таранов А.Е. Определение локальных и интегральных гидродинамических характеристик контейнеровоза в цифровом бассейне // Труды Крыловского государственного научного центра. 2019. Т. 3, №389. С. 73-82.
21. Menter F. R., Kuntz M., Langtry R. Ten Years of Industrial Experience with the SST Turbulence Model // Turbulence, Heat and Mass Transfer. 2003. P. 625 – 632.
22. Menter F.R., Langtry R.B., Likki S.R., Suzen Y.B., Huang P.G., Völker S. A. Correlation-based Transition Model using Local Variables. Part 1: Model Formulation // ASME J. Turbomachinery, 2006. V.128, No. 3, pp. 413-422.
23. Лойцянский Л.Г. Механика жидкости и газа. Москва: Государственное издательство технико-теоретической литературы, 1950. 678 с.
24. Роуч П. Вычислительная гидродинамика. Москва: Мир, 1980. 618 с.

# Дополнительное распараллеливание MPI программ с помощью системы SAPFOR \*

Н.А. Катаев, А.С. Колганов

ИПМ им. М.В. Келдыша РАН

Системы SAPFOR и DVM были спроектированы и разработаны для упрощения разработки параллельных программ научно-технических расчетов. Главной целью системы SAPFOR является автоматизация процесса отображения последовательных программ на параллельные архитектуры в модели DVMH. В некоторых случаях пользователь системы SAPFOR может рассчитывать на полностью автоматическое распараллеливание, если программа была написана или приведена к потенциально параллельному виду. DVMH модель представляет собой расширение стандартных языков Си и Фортран спецификациями параллелизма, которые оформлены в виде директив и не видимы стандартным компиляторам. В статье будет рассмотрено автоматизированное дополнительное распараллеливание существующих MPI программ с помощью системы SAPFOR, где, в свою очередь, будут использованы новые возможности DVMH модели по распараллеливанию циклов в MPI программе внутри узла. Данный подход позволяет существенно снизить трудоемкость распараллеливания MPI программ на графические ускорители и многоядерные процессоры, сохранив при этом удобство сопровождения уже написанной программы. Данная возможность в системе SAPFOR была реализована для языков Фортран и Си. Эффективность данного подхода показана на примере некоторых приложений из пакета NAS Parallel Benchmarks.

*Ключевые слова:* SAPFOR, DVMH, MPI, автоматизация распараллеливания, дополнительное распараллеливание, ускорители, гетерогенные кластеры

## 1. Введение

Многопроцессорные вычислительные системы уже достаточно давно используются для проведения расчетов, накопился достаточно большой объем параллельных программ, способных задействовать несколько узлов вычислительного кластера. При этом, одним из основных инструментов разработки таких программ остается стандарт MPI. Но появившиеся сравнительно недавно гетерогенные вычислительные системы, узлы которых образованы многоядерными процессорами и ускорителями, требуют применения дополнительных средств разработки параллельных программ. Чтобы эффективно использовать все имеющиеся вычислительные ресурсы, в MPI-программы приходится добавлять дополнительные спецификации параллелизма. Низкоуровневые модели параллельного программирования (CUDA, OpenCL, POSIX Threads) дают полный контроль за процессом выполнения программы. Но в сочетании с тем, что, разработчику также приходится разбираться в структуре исходной MPI-программы, которая может быть существенно сложнее аналогичной последовательной программы, существенно увеличивают сложность разработки кода. Отладка таких MPI-программ, для которых было выполнено дополнительное распараллеливание, также является трудоемкой задачей.

Решением в данной ситуации может быть использование высокоуровневых моделей программирования (OpenMP, OpenACC), узкоспециализированных языков программирования (DSL) [1–3], а также параллельных библиотек общего назначения [4, 5]. Дополнением к этому может стать создание инструментальных средств автоматизирующих процесс дополнительного распараллеливания MPI-программ.

Стандарты OpenMP и OpenACC активно развиваются в указанном выше направлении,

---

\*Исследование выполнено при финансовой поддержке РФФИ в рамках научных проектов № 19-07-00889, 19-07-01101 и 20-01-00631.

но все еще не позволяют в полной мере задействовать все ресурсы вычислительного узла. В современных компиляторах данные стандарты реализованы не полностью [6], а для получения существенного выигрыша от их использования пользователю приходится разбираться в том, как высокоуровневые спецификации влияют на выполнение программы, погружаясь в детали их реализации в конкретном компиляторе и подбирая соответствующие оптимизационные опции. Более того в данный момент отсутствуют удобные высокоуровневые средства как для функциональной отладки, так и для отладки эффективности данных программ.

Если рассматривать автоматически распараллеливающие компиляторы как отдельные инструменты, то их возможности оказываются существенно ограничены и не позволяют выполнить приемлемое распараллеливание в узле кластера для реальных программ.

В данной ситуации, перспективным видится смешанный подход, объединяющий использование высокоуровневой модели параллельного программирования, в качестве целевого способа описания параллелизма в программе, и системы автоматизации, осуществляющей наиболее трудоемкие и приводящие к ошибкам этапы распараллеливания автоматически, но дающей пользователю контроль за ходом распараллеливания, достаточный для получения эффективных программ.

В качестве высокоуровневой модели может выступать модель DVMH [7,8], включающая в себя DVMH языки параллельного программирования, являющиеся директивными расширениями стандартных последовательных языков программирования Си и Фортран. Одним из основных преимуществ DVMH языков, наряду с высокоуровневой семантикой используемых конструкций, являются реализованные оптимизации времени выполнения программы. Это существенно снижает сложность разработки инструментов автоматизирующих процесс распараллеливания. Кроме того, DVM система включает в свой состав инструменты для функциональной отладки и отладки эффективности создаваемых программ.

Система SAPFOR (System FOR Automated Parallelization) [9, 10] может выступать в качестве инструмента автоматизации параллельного программирования. С одной стороны в состав системы входит автоматически распараллеливающий компилятор [11], а с другой SAPFOR предоставляет пользователю дополнительные возможности по контролю за процессом распараллеливания, как с использованием графического интерфейса пользователя [12], так и с помощью дополнительного указания свойств программы непосредственно в ее исходном коде. Чтобы расширить доступные возможности по анализу программ, в системе SAPFOR наряду со статическим анализом [13], также поддерживается динамический анализ [14]. Система предоставляет пользователю набор реализованных преобразований для языков Фортран и Си (подстановка функций, удаление мертвого кода, распространение выражений и др.), которые пользователь может выбрать для автоматического выполнения.

Изначально модель DVMH была разработана с целью скрыть от пользователя не только параллелизм внутри узла, но и обеспечить распараллеливание программ для систем с распределенной памятью. Но в последних реализациях компиляторов с DVMH языков был добавлен дополнительный режим, направленный на использование возможностей DVMH модели в MPI-программах. В данной статье рассматриваются новые возможности системы SAPFOR, которые предназначены для автоматизации дополнительного распараллеливания MPI программ и опираются на возможности, добавленные в систему DVM.

Статья состоит из введения, 4 разделов и заключения. В разделе 2 приведен краткий обзор работ, направленных на автоматизацию разработки параллельных программ для систем с общей памятью (многоядерных процессоров и графических ускорителей). В разделе 3 рассматриваются возможности, добавленные в DVMH модель для реализации дополнительного внутреузлового параллелизма в MPI программах. Раздел 4 посвящен системе SAPFOR и особенностям, связанным с автоматизацией дополнительного распараллеливания MPI программ. Результаты вычислительных экспериментов, охватывающие распараллеливание некоторых программ из набора NAS Parallel Benchmarks [15], приведены в разделе 5.

## 2. Обзор существующих работ

Можно выделить два основных подхода, применяемых при разработки инструментов автоматизирующих распараллеливание программ. Подход, наиболее понятный для пользователя и поэтому используемый в SAPFOR в силу ее направленности на поддержание процесса интерактивного распараллеливания, заключается в последовательном изменении кода программы, чтобы впоследствии привести ее к параллельному виду. Многие из выполняемых преобразований оказываются общими для разных программ и могут быть автоматизированы (подстановка функций, разделение и слияние циклов, разворот цикла и др. [16]). Часть из этих преобразований уже реализована в системе SAPFOR и при необходимости может быть запрошена пользователем. Альтернативой данному подходу является использование математической модели, описывающей поведение распараллеливаемой программы (модель многогранников, или полиэдральная модель). Данный подход позволяет сформировать оптимизационную задачу, решение которой фактически описывает переход от исходной программы к ее параллельной версии за один шаг преобразования.

Одним из основных недостатков данного подхода является большое количество ограничений, накладываемых на распараллеливаемую программу: в результате данный подход годится, в первую очередь, для локального применения к хорошо структурированным участкам кода (SCoP). Кроме того, степень изменений, вносимых в код программы, которые порождает решение оптимизационной задачи, оказывается такова, что пользователь фактически лишается возможности участвовать в процессе распараллеливания, даже если результирующая параллельная программа остается на языке высокого уровня. Таким образом, данный подход в первую очередь подходит для автоматически распараллеливающих компиляторов [17–21], а не инструментов автоматизации.

Компиляторы Pluto [17] и PPCG [18] направлены на преобразование только Си программ, при этом, Pluto выполняет распараллеливание только на многоядерные процессоры (с использованием OpenMP), в то время как PPCG опирается на CUDA или OpenCL для отображения программ на графический ускорители. Оба инструмента сохраняют высокий уровень исходного языка, но вносимые изменения негативно сказываются на возможности восприятия кода неподготовленным программистом. Существенным недостатком является необходимость явно задавать фрагменты исходного кода, которые должны быть оптимизированы. Для этого используется специальная директива **scop**. Так как оптимизация выполняется локально, то от задания данной директивы существенно зависит результат распараллеливания. Например, если соседние гнезда циклов поместить в разные области распараллеливания, то PPCG добавит в код программы вызовы CUDA, выполняющие обмены с графическим ускорителем в начале и в конце каждой области: глобальная оптимизация обменов не выполняется. Кроме того, Pluto и PPCG не обрабатывают участки кода, содержащие редуцирующие операции, и оставляют соответствующий код последовательным. В этом случае вычисления не могут быть полностью выполнены на ускорителях, и требуются дополнительные обмены данными с центральным процессором.

Другим примером применения модели многогранников является инструмент Polly [19] и его дальнейшее расширение для отображения программ на графические ускорители – Polly-ACC [20]. Данные инструменты используют LLVM [22] для анализа и преобразования программ и доступны в составе компилятора Clang. Использование низкоуровневого представления (LLVM IR) лишает пользователя возможности как-либо изменять код параллельной программы, но при этом это расширяет применимость данных инструментов на языки, для которых может быть построено LLVM IR. В отличие от Pluto и PPCG оптимизируемые участки кода определяются автоматически и также выполняется попытка оптимизировать обмены с графическим ускорителем. Отдельно была реализована возможность распараллеливания редуцирующих операций [23]. Хотя оптимизируемые участки кода определяются автоматически, выполнить распараллеливание удастся не всегда: инструменты опираются на статический анализ кода, возможности которого ограничены, особенно если в програм-



мах используются указатели и адресная арифметика. Чтобы понять, как привести программу в распараллеливаемую форму, пользователю может потребоваться изучить возникшие проблемы анализа, описываемые в терминах низкоуровневого представления LLVM IR.

Еще одним интересным инструментом является Apollo [21], выполняющий спекулятивное распараллеливание программы во время выполнения. Но в данном инструменте реализована возможность распараллеливания только для многоядерных процессоров.

### 3. Возможности модели DVMH для распараллеливания MPI программ

Обычно DVM подход [7] предполагает распараллеливание последовательной программы, в которую программист вставляет директивы распределения данных, а затем – директивы распределения вычислений. После этого, компилятор DVMH преобразует такую программу в программу в модели MPI+OpenMP+CUDA, которая затем с помощью библиотеки времени выполнения (runtime system, RTS) выполняется на гетерогенном кластере. RTS берет на себя распределение данных по узлам кластера согласно тому, как были составлены директивы распределения данных, а затем – отображает соответствующие этим данным вычисления в параллельных циклах. Таким образом, применение данного подхода в MPI программах потребует от программиста либо наличия исходной последовательной версии, либо перевода MPI программы обратно в последовательную версию. Зачастую, перевод MPI программы в исходную последовательную попросту невозможен, либо требует достаточно больших усилий. Кроме того, в этом случае достаточно велика вероятность совершения ошибки.

Для решения описанных проблем в DVM системе был введен специальный режим, в котором RTS не выполняет никаких межпроцессорных взаимодействий и не препятствует работе MPI функций, используемых в программе пользователя. Чтобы задействовать данный режим требуется специальная сборка DVM системы, в результате которой RTS работает локально в каждом MPI процессе вне зависимости от того, на какой процессорной решетке была запущена программа.

Как уже было отмечено выше, изначально DVMH модель предполагает отображение витков параллельных циклов на элементы распределенных массивов в терминах DVMH. В MPI программе пользователь сам берет на себя ответственность за распределение данных. Поэтому потребовалось ввести новую возможность в DVMH модель для того, чтобы отобразить параллельные циклы в такой программе на графические ускорители и многоядерные процессоры. Данная возможность позволяет локально в каждом MPI процессе распараллеливать витки параллельных циклов без привязки к элементам массивов. При этом, несмотря на отсутствие распределения данных в рамках модели DVMH, сохраняется возможность использования оптимизаций времени выполнения программ, доступных в DVM системе. К ним относятся: использование автоматической трансформации данных на графическом процессоре, а также использование интеллектуального механизма управления перемещением данных между памятью ЦПУ и ГПУ; использование оптимизирующих настроек RTS системы. Также сохраняется возможность использования удобных средств по отладке и анализу производительности MPI+DVMH программ.

В модели DVMH распределение данных выполняется при помощи выравнивания (директивы **align**), которое для каждого элемента массива **A** ставит в соответствие элемент или секцию массива **B**. При отображении на какой-либо процессор элемента массива **B**, на этот же процессор будет распределен и элемент массива **A** в соответствии с правилами выравнивания данных. Распределение вычислений в модели DVMH выполняется с учетом распределения данных. Правило отображения параллельного гнезда циклов задается при помощи спецификации **on**, которая позволяет связать циклы тесно вложенного гнезда с измерениями распределенных массивов.

Данная информация позволяет компилятору и RTS выполнять ряд оптимизаций, которые могут существенно повысить эффективность выполнения параллельной программы. Например, зная о том, как связаны массивы между собой, и зная правила отображения цикла, RTS может определить оптимальное для данного цикла представление массивов в памяти вычислительного устройства и выполнить динамическую трансформацию массивов только в тех случаях, когда это требуется. В результате, на графическом ускорителе все обращения к глобальной памяти будут выполняться CUDA-нитьями одного варпа наилучшим образом – соседние нити блока будут обращаться к соседним ячейкам памяти, в результате чего параллельный цикл может выполняться до 10 раз быстрее [24].

При распараллеливании MPI программ в модели DVMH описанные ранее директивы **distribute**, **align**, и **on** не используются. Программист сам реализует распределение данных и распределение вычислений по узлам кластера с помощью MPI. Для задания связи между витками параллельных циклов и используемыми в них массивами в DVMH модель была введена новая спецификация **tie** [25], а также понятие нераспределенного параллельного цикла. При помощи данной спецификации программист передает RTS информацию о том, с каким циклом в гнезде есть связь у конкретного измерения массива (а также направление этой связи – прямое или обратное), если связи нет, то указывается «\*». Измерение массива может быть отображено только на один из параллельных циклов в гнезде.

Для параллельного выполнения циклов с регулярными зависимостями по данным на графических ускорителях в DVM-системе реализован метод гиперплоскостей. Все элементы, лежащие на одной гиперплоскости, могут быть вычислены независимо друг от друга. При таком порядке выполнения витков цикла снова возникает проблема эффективного доступа к глобальной памяти в силу того, что параллельно обрабатываются не локально расположенные в памяти элементы массивов. Для эффективного выполнения таких циклов в RTS реализована диагональная трансформация, в результате которой соседние элементы массивов на диагоналях (в плоскости необходимых измерений) располагаются в соседних ячейках памяти, что позволяет применять технику выполнения цикла с зависимостями по гиперплоскостям без значительной потери производительности на операциях доступа к глобальной памяти графического ускорителя. Для обеспечения эффективного выполнения таких циклов в MPI+DVMH программе совместно применяются спецификаций **tie** и **across** для всех необходимых массивов.

На рис.1 и рис.2 показаны примеры нераспределенного параллельного цикла, который может выполняться на многоядерном процессоре или графическом процессоре.

**Листинг 1.** Нераспределенный параллельный цикл в языке CDVMH

```
#pragma dvm region targets(CUDA) out(A)
{
#pragma dvm parallel([i][j][k]) tie(A[i][j][k])
for (int i = Li; i <= Hi; i++)
  for (int j = Lj; j <= Hj; j++)
    for (int k = Lk; k <= Hk; k++)
      A[i][j][k] = ....
}
```

**Листинг 2.** Нераспределенный параллельный цикл в языке FDVMH

```
!DVM$ REGION TARGETS(HOST) OUT(A)
!DVM$ PARALLEL(I,J,K) TIE (A(K,J,I))
DO I = Li, Hi
  DO J = Lj, Hj
    DO K = Lk, Hk
      A(K,J,I) = ....
!DVM$ END REGION
```

Для того, чтобы параллельный цикл мог быть отображен на многоядерные ЦПУ или ГПУ, необходимо его окружить спецификацией **region**, которая обозначает начало и конец вычислительной области [8]. В вычислительную область выделяется часть программы

с одним входом и одним выходом для возможного ее выполнения на многоядерном процессоре или графическом ускорителе. Статически или динамически вложенные регионы не допускаются. Для выполнения вычислительных регионов на ГПУ требуется соблюдение некоторых ограничений на ввод/вывод и вызовы процедур из параллельных циклов. Регион может быть выполнен только на ЦПУ или только на ГПУ. Данная возможность обеспечивается указанием необязательной спецификации **target**.

Фрагменты кода, которые не входят в вычислительные регионы, всегда выполняются на центральном процессоре, что приводит к необходимости актуализации данных между памятью ЦПУ и памятью ГПУ. Для выполнения необходимых обменов, в программе должны быть размещены директивы **actual** и **get\_actual**.

Директива **get\_actual** делает все необходимые обновления для того, чтобы в памяти центрального процессора были актуальные (то есть самые новые) значения данных в указанных в списке массивах и скалярах. Директива является исполняемой и выполняется в той точке программы, где она была поставлена. Директива **actual** объявляет тот факт, что указанные в списке массивы и скаляры имеют самые новые значения в памяти центрального процессора. Значения указанных переменных и элементов массивов, находящиеся в памяти ускорителей, считаются устаревшими и перед использованием будут при необходимости обновлены. В данном случае, помечается только сам факт того, что данные были обновлены на ЦПУ, но физического копирования в точке постановки директивы не происходит.

Отложенная семантика директив актуализации позволяет RTS избегать избыточной передачи данных. Более того, нет никакой разницы, все ли регионы выполняются на ГПУ или какая-то их часть предназначена для выполнения на ЦПУ. Директивы актуализации влияют только на передачу данных между регионами и последовательными фрагментами кода. Система поддержки в любых случаях будет управлять необходимой передачей данных в автоматическом режиме. Реализованная автоматизация перемещения данных между ЦПУ и ГПУ в DVMH модели позволяет системе SAPFOR автоматически расставлять директивы актуализации данных оптимальным образом.

#### 4. Автоматизация дополнительного распараллеливания MPI программ с помощью системы SAPFOR

Для того, чтобы задействовать весь новый потенциал для распараллеливания MPI программ с помощью DVMH модели, необходимо добавить в код программы спецификацию **tie**, а также изменить подход к анализу программы в системе SAPFOR: больше нет необходимости выполнять распределение данных и в соответствии с ним распределение вычислений.

На первом этапе системой SAPFOR для распараллеливания отбираются самые внешние гнезда тесно-вложенных циклов. Каждое гнездо циклов проверяется на предмет того, может ли оно выполняться параллельно: из цикла не должно быть побочных выходов, не должно быть неразрешимых зависимостей между витками цикла, цикл должен быть в канонической форме, и др. [11]. Для каждого цикла также определяются входные и выходные данные, которые используются в нем, так как в компиляторе CDVMH не реализован междо-процедурный анализ и в основном все данные считаются как входными, так и выходными. Консервативная информация о входных и выходных данных в вычислительных областях может привести к дополнительным копированиям между ГПУ и ЦПУ.

На следующем этапе системе SAPFOR необходимо проанализировать все обращения к памяти, которые выполняются в теле рассматриваемого гнезда циклов. В каждом выражении в обращении к массиву ищутся индуктивные переменные циклов для того, чтобы выполнить сопоставление измерений массивов с витками циклов. Для этого выполняется попытка привести выражения в форму, зависящую от номера витка цикла за счет анализа внутреннего представления программы (expression propagation, scalar evolution). В силу того, что распределение данных строится пользователем с помощью MPI, нет необходимости

определять точное соответствие измерений массивов с витками циклов, которое в DVMH модели выражаются с помощью линейной связи. Тем самым, система SAPFOR может параллелизовать циклы и с косвенной адресацией, так как нам достаточно установить факт использования индуктивной переменной цикла в том или ином измерении массива.

После того, как был определен набор параллельных циклов, системе SAPFOR необходимо разделить программу на вычислительные регионы, в которые будут входить параллельные циклы, и на вне региональное пространство, которое будет выполняться на ЦПУ в последовательном режиме. Алгоритм расстановки регионов состоит из двух этапов. На первом этапе каждый параллельный цикл окружается своим собственным регионом. А на втором этапе происходит объединение соседних регионов в один для уменьшения накладных расходов на вход и выход из региона.

После того, как программа была поделена на регионы, необходимо обеспечить консистентность данных между ЦПУ и ГПУ. Чтобы уменьшить количество копирований, необходима расстановка директив актуализации данных оптимальным образом. DVM система обеспечивает ряд оптимизаций времени выполнения программы, что облегчает работу системе SAPFOR. Например, если пользователь вставит две одинаковые директивы **get\_actual** подряд, то копирование будет выполнено только один.

Основная цель на данном этапе работы системы SAPFOR – определить корректный набор данных, который необходимо обновить в памяти ГПУ или ЦПУ. Для данного анализа используются ранее собранные входные и выходные данные для каждого из циклов. Также используется анализ псевдонимов в случае косвенной адресации в программах на языке Си. Система SAPFOR выполняет актуализацию только тех переменных, которые используются на чтение или запись внутри вычислительных регионов. На основе полученных данных, система SAPFOR размещает директиву **actual** после каждого присваивания переменной, которое выполняется во вне региональном пространстве. Директива **get\_actual** ставится перед теми операторами, в которых присутствует чтение из переменных, которые были в свою очередь модифицированы в каком-либо вычислительном регионе.

Промышленные программы обычно состоят из большого количества процедур и функций. Система SAPFOR определяет три класса таких процедур: встроенные процедуры (например, математические функции), пользовательские функции и внешние процедуры. В случае наличия внешних процедур системе SAPFOR приходится принимать консервативное решение вставлять директивы актуализации всех используемых в вычислительных регионах переменных до и после вызова, так как тело процедуры не доступно для анализа. Для уменьшения накладных расходов на копирование данных, пользователь может подсказать системе SAPFOR, например, может указать, что процедура не содержит побочных эффектов. Так как функции MPI являются внешними и информация об использовании в параметрах этих функций переменных является стандартизированной, мы выделили их в системе SAPFOR в отдельный четвертый класс – класс MPI функций, что позволяет размещать директивы актуализации оптимально без подсказок пользователя.

Использование указателей в программах может негативно сказаться на результатах анализа в системе SAPFOR, так как это препятствует выполнению некоторых преобразований внутреннего представления программы. В частности, используемый в системе SAPFOR анализ редуцированных операций не способен обработать переменные, каким-либо образом участвующие в операциях адресной арифметики. Так как редуцированная операция – это коллективная операция выполняемая сразу всеми/несколькими процессорами, то для обмена значениями соответствующих переменных будут использованы функции библиотеки MPI. При этом для описания передаваемых данных будет использован указатель на буфер с данными (возможно состоящий из единственной переменной).

Чтобы расширить возможности системы SAPFOR и реализовать анализ редуцированных операций в MPI программах было реализовано дополнительное преобразование внутреннего представления программы в системе. Данное преобразование скрыто от пользователя

и не влияет на исходный код распараллеливаемой программы, но расширяет возможности для анализа программ. Для тех циклов, в которых была предварительно выявлена зависимость по некоторой переменной участвующей в операциях адресной арифметики, заводится локальный дубликат этой переменной и обращения к переменной в цикле заменяются на обращения к ее дубликату. Это позволяет воспользоваться уже имеющимся в SAPFOR анализом редуccionных переменных.

Возможности анализа предыдущих версий SAPFOR также были ограничены необходимостью во многих случаях выполнять предварительную подстановку функций на уровне исходного кода программы [11]. Нами была добавлена возможность определять функции, которые необходимо подставить, автоматически на основе предварительных результатов анализа программы. Более того, выполнение подстановки функций было перенесено на уровень внутреннего представления программы и, таким образом, исходный код программы остается неизменным, если этого не требуется для задания спецификаций параллелизма.

## 5. Вычислительные эксперименты

Рассмотрим возможности дополнительного распараллеливания MPI-программ, предоставляемые системой SAPFOR совместно с системой DVM, на примере распараллеливания вычислительных приложений BT, CG и EP из пакета NAS Parallel Benchmarks.

Эффективность выполнения исходных MPI программ, а также полученных с помощью системы SAPFOR MPI программ с DVM директивами, была исследована на суперкомпьютере K60 [26], состоящем из процессоров Intel Xeon Gold 6142v4 и графических ускорителей NVIDIA V100 GPUs с архитектурой Volta. Каждый узел содержит два 16-ти ядерных процессора (ЦПУ), связанных посредством общей памяти (архитектура NUMA), и четыре графических ускорителя (ГПУ). Пиковая производительность узла составляет примерно 60 TFLOPs в вычислениях одинарной точности и 30 TFLOPs в вычислениях двойной точности. Таким образом, для достижения высокой производительности параллельных программа, достаточно задействовать небольшое количество вычислительных узлов. Для вычислительных экспериментов были использованы максимально возможные ресурсы четырех узлов: для MPI программ использовалось до 128 ядер ЦПУ, а для MPI+DVM программ – до 128 ядер ЦПУ и 16 ГПУ. Времена выполнения различных версий MPI программ, написанных на языках Фортран и Си приведены в таблицах 1 и 2 соответственно.

**Таблица 1.** Время выполнения в секундах программ на языке Фортран, NPB 3.3 класс D.

	MPI программы			Преобр. MPI программы			MPI программы + <b>FDVMH</b>		
	BT	CG	EP	BT	CG	EP	BT	CG	EP
1 узел	665.1	397.5	93.68	785.29	376.8	83.34	63.3	80.99	0.62
2 узла	361.6	209.6	46.53	428.07	229.61	42.06	50.3	42.6	0.38
4 узла	196.8	91.3	23.3	232.36	96.67	25.16	45.5	34.3	0.17

Времена выполнения оригинальных версий, написанных разработчиками пакета NAS Parallel Benchmark, приведены в первой группе столбцов (MPI программы). Изначально, рассматриваемые программы были написаны только на Фортране, поэтому потребовалось выполнить перевод рассматриваемых программ на язык Си вручную.

Чтобы получить автоматически распараллеливаемые версии программ, были выполнены их предварительные преобразования (подстановка функций, объединение циклов, сужение размерности приватных массивов) [11]. Вторая группа столбцов (Преобразованные MPI

программы) показывает времена выполнения преобразованных версий. Большинство преобразований было выполнено системой SAPFOR, другая часть преобразований, не реализованных на данный момент в системе или специфичных для конкретной программы и трудно формализуемых в виде отдельного преобразования, была выполнено вручную.

Времена выполнения программ, полученных после автоматического распараллеливания преобразованных версий приведены в группе столбцов: MPI программы + DVMH.

**Таблица 2.** Время выполнения в секундах программ на языке Си, NPV 3.3 класс D.

	MPI программы			Преобр. MPI программы			MPI программы + <b>CDVMH</b>		
	BT	CG	EP	BT	CG	EP	BT	CG	EP
1узел	694.6	326.16	98.41	768.5	328.8	99.37	97.7	186.12	0.67
2узла	386	218.9	49.29	421.3	214.3	50.05	75.7	96.75	0.38
узла	215	89.2	24.71	229.06	92.46	26.5	67.33	71.79	0.17

Для выполнения исходных и преобразованных MPI программ использовалось максимально возможное количество ядер ЦПУ одного или двух узлов. А для выполнения MPI+DVHM программ использовалось максимально возможное количество ГПУ в узле. Исходя из приведенных времен можно сделать вывод, что преобразования практически не замедляют выполнение программы, позволяя использовать систему SAPFOR для автоматического распараллеливания циклов на ГПУ и многоядерные ЦПУ в рамках одного узла.

На данный момент компиляторы CDVMH и FDVMH реализуют различный набор оптимизаций при выполнении конвертации кода. В компиляторе CDVMH отсутствуют некоторые оптимизации, доступные в компиляторе FDVMH, что является основной причиной в различии ускорений MPI+DVHM программ. В некоторых случаях, компилятор FDVMH способен дополнительно распараллелить не тесно вложенные циклы. Это позволяет значительно повысить эффективность выполнения программы CG на графическом процессоре.

Максимально возможное ускорение приложения BT с использованием четырех графических процессоров составляет около 10.5 раз по сравнению с 32 MPI процессами. Ускорение выполнения программ CG и EP на графических процессорах при использовании двух узлов постоянно и не зависит от конфигурации. В данном случае оно составляет примерно 5 раз и 151 раз соответственно. При использовании четырех узлов для программы CG не хватает данных для вычисления, поэтому ускорение падает до 2.6 раз.

Такая разница в полученном ускорении в разных приложениях обусловлена их разной алгоритмической сложностью. Программа EP производит большое количество независимых вычислений, причем количество обращений к памяти практически нет. Тем самым, в данном приложении можно видеть наибольшее полученное ускорение. В то время как в программе CG основная доля времени приходится на умножение разреженной матрицы на вектор, что приводит к косвенной индексации и снижению общей производительности в целом.

И, наконец, программа BT состоит из большого количества вычислений, большого количества обращений к памяти, а также циклов с регулярными зависимостями по данным. В результате чего, в программе появляется большое количество межпроцессорных обменов, которые сильно сказываются на масштабируемости программы при переходе с одного узла на два и более. Рис. 1 показывает соотношение времен, затрачиваемых программой на вычисления и на накладные расходы (коммуникации, пересылки ЦПУ-ГПУ). Из графиков видно, что количество коммуникаций и связанные с ними копирования данных с ЦПУ на ГПУ и обратно при увеличении количества процессов в двое не сокращается и не зависит от количества процессов. В то время как время, затрачиваемое на выполнение параллель-

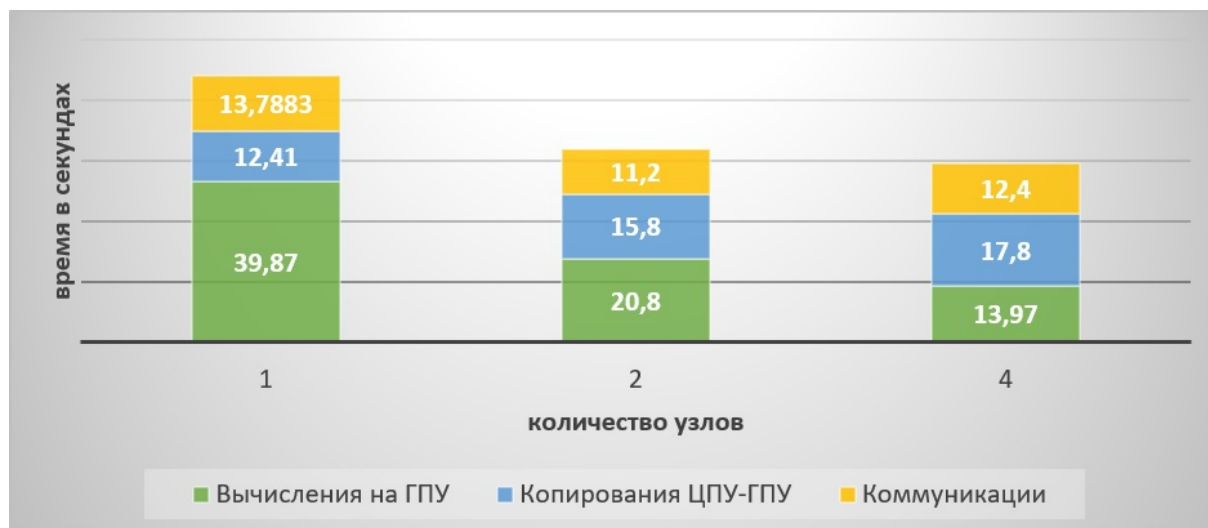


Рис. 1. Соотношение времени вычислений и коммуникаций на примере VT

ных циклов на графическом процессоре, пропорционально уменьшается в соответствии с количеством использованных устройств.

## 6. Заключение

В статье были рассмотрены новые возможности модели DVMH и их применения в системе SAPFOR для автоматизированного дополнительного распараллеливания существующих MPI программ на примере некоторых программ из пакета NAS Parallel Benchmarks.

Предлагаемый нами подход к разработке параллельных программ сочетает модель программирования на основе директив (DVMH) и инструменты автоматизации и взаимодействия с пользователем (SAPFOR). Разрабатываемые системы взаимно дополняют друг друга, что, в свою очередь, позволяет получить существенное преимущество по сравнению с другими моделями параллельного программирования, такими как MPI+OpenMP или MPI+OpenACC при разработке параллельных программ для гетерогенных кластеров.

Прежде всего, в систему SAPFOR входит автоматически распараллеливающий компилятор, который успешно справляется с потенциально параллельными программами без вмешательства со стороны пользователя. Если же реализованный в системе SAPFOR анализ кода не справляется, то пользователь может помочь системе с помощью соответствующих директив, указав недостающие свойства программы, либо выполнить с помощью системы SAPFOR необходимые преобразования, которые не приводят к снижению производительности, при этом повышают уровень ее потенциальной параллельности. Можно отметить, что преобразования производятся на уровне исходного кода и не требуют детального изучения параллельных архитектур и языков параллельного программирования.

Благодаря заложенным в модели DVMH автоматизации и оптимизации перемещения данных между памятью ЦПУ и памятью GPU SAPFOR может эффективно автоматизированно отображать (а в некоторых случаях полностью автоматически) существующие MPI программы на графические процессоры. Кроме того, система SAPFOR опирается на оптимизации, выполняемые DVM системой во время выполнения программы: трансформацию массивов во время выполнения программы для обеспечения наилучшего доступа к памяти GPU, динамическую компиляцию и оптимизацию CUDA-обработчиков во время выполнения программы, эффективное выполнение параллельных циклов с регулярными зависимостями по данным на GPU и др. Это позволяет достичь высокой эффективности получаемых MPI+DVMH программ. В тоже время, данные оптимизации скрыты от пользователя и до-

ступни через высокоуровневые спецификации параллелизма DVMH модели, таким образом упрощая поддержку разрабатываемых параллельных кодов.

В DVM систему также входят средства анализа эффективности выполнения программ. Выдаваемые характеристики выполнения программ описывают проблемы потери эффективности понятными пользователю терминами. Все получаемые характеристики выполнения привязаны к DVMH директивам и исходному коду. В дальнейшем, система SAPFOR сможет использовать полученные результаты анализа эффективности для оптимизаций MPI программы.

Таким образом, системы SAPFOR и DVM могут значительно сократить усилия, необходимые для дополнительного распараллеливания MPI программ, позволяющего задействовать все доступные внутри узла ресурсы (многоядерные процессоры и графические ускорители), а также существенно повысить эффективность выполнения параллельной MPI программы. Мы надеемся, что данный подход должен помочь эффективной разработке и оптимизации масштабируемых программ для суперкомпьютеров.

## Литература

1. Ragan-Kelley, J., Barnes, C., Adams, A., Paris, S., Durand, F., Amarasinghe, S.P. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines // Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, 2013. P. 519–530. DOI: 10.1145/2499370.2462176
2. Beaugnon, U., Kravets, A., Sven van Haastregt, Baghdadi, R., Tweed, D., Absar, J., Lokhmotov, A. Vobla: A vehicle for optimized basic linear algebra // Proc. of the 2014 SIGPLAN/SIGBED Conf. on Languages, Compilers and Tools for Embedded Systems, LCTES '14, New York, NY, USA, 2014. P. 115–124. DOI: 0.1145/2666357.2597818
3. Zhang, Y., Yang, M., Baghdadi, R., Kamil, S., Shun, J., Amarasinghe, S. Graphit: A high-performance graph dsl // Proc. ACM Program. Lang., 2(OOPSLA), 2018. P. 121:1–121:30. DOI: 10.1145/3276491
4. An, P., Jula, A., Rus, S., Saunders, S., Smith, T., Tanase, G., Thomas, N., Amato, N., Rauchwerger, L. STAPL: An adaptive, generic parallel C++ library // Dietz H.G. (eds) Languages and Compilers for Parallel Computing. LCPC 2001. Lecture Notes in Computer Science, vol. 2624, Springer Berlin Heidelberg, 2003. P. 193–208. DOI: 10.1007/3-540-35767-X\_13
5. Bell, N., Hoberock, J.: Thrust: A Productivity-oriented library for CUDA // GPU Computing Gems, Jade Edition, Edited by Wen-mei W. Hwu, 2012. P. 359–371. DOI: 10.1016/B978-0-12-385963-1.00026-5
6. Компиляторы и инструменты OpenMP.  
URL: <https://www.openmp.org/resources/openmp-compilers-tools/>. (Дата обращения: 8 апреля 2021).
7. Konovalov, N.A., Krukov, V.A, Mikhajlov, S.N., Pogrebtsov, A.A. Fortan DVM: a Language for Portable Parallel Program Development // Programming and Computer Software. Vol. 21, No. 1. 1995. P. 35–38.
8. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Пригула М.Н., Сазанов Ю.Л. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2012. № 18(277). С. 82-92.



9. Клинов М.С., Крюков В.А. Автоматическое распараллеливание Фортран-программ. Отображение на кластер // Вестник ННГУ, 2009. №2. С. 128–134.
10. Бахтин В.А., Бородич И.Г., Катаев Н.А., Клинов М.С., Ковалева Н.В., Крюков В.А., Поддерюгина Н.В. Диалог с программистом в системе автоматизации распараллеливания САПФОР // Вестник ННГУ, 2012. № 5(2). С 252-245.
11. Kataev, N. LLVM Based Parallelization of C Programs for GPU // Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2020. Communications in Computer and Information Science, vol 1331. Springer, Cham, 2020. P. 436–448. DOI: 10.1007/978-3-030-64616-5\_38
12. Kataev, N. Interactive Parallelization of C Programs in SAPFOR // Scientific Services & Internet 2020. CEUR Workshop Proceedings, Vol. 2784, 2020. P. 139–148.
13. Kataev, N. Application of the LLVM Compiler Infrastructure to the Program Analysis in SAPFOR // Voevodin V., Sobolev S. (eds) Supercomputing. RuSCDays 2018. Communications in Computer and Information Science, Vol. 965,. Springer, Cham, 2018. P. 487–499. DOI: 10.1007/978-3-030-05807-4\_41
14. Kataev, N., Smirnov, A., Zhukov A. Dynamic data-dependence analysis in SAPFOR // CEUR Workshop Proceedings, Vol. 2543, 2020, P. 199–208. DOI: 10.20948/abrau-2019-62
15. NAS Parallel Benchmarks. URL: <https://www.nas.nasa.gov/publications/npb.html> (Дата обращения: 8 апреля 2021).
16. Wolfe, M. High Performance Compilers for Parallel Computing // Addison-Wesley, 1995.
17. Bondhugula, U., Hartono, A., Ramanujam, J., Sadayappan, P. A practical automatic polyhedral parallelizer and locality optimizer // SIGPLAN Notices, 43(6), 2008. P. 101–113. DIO: 10.1145/1375581.1375595
18. Verdoolaege, S., Juega, J. C., Cohen, A., Gomez, J. I., Tenllado, C., Catthoor, F. Polyhedral parallel code generation for CUDA // ACM Trans. Archit. Code Optim., Vol. 9, No. 4, 2013. DOI: 10.1145/2400682.2400713
19. Grosser, T., Groesslinger, A., Lengauer. C. Polly — performing polyhedral optimizations on a low-level intermediate representation // Parallel Processing Letters, Vol 22(04), 2012. DOI: 10.1142/S0129626412500107
20. Grosser, T., Hoefler, T. Polly-ACC Transparent compilation to heterogeneous hardware // ICS '16: Proceedings of the 2016 International Conference on Supercomputing June 2016. P. 1–13. DOI: 10.1145/2925426.2926286
21. Caamano, J.M.M., Sukumaran-Rajam, A., Baloian, A., Selva, M., Clauss, P. APOLLO: Automatic speculative POLyhedral Loop Optimizer // 7th International Workshop on Polyhedral Compilation Techniques (IMPACT), January 2017, Stockholm, Sweden, 2017.
22. Lattner, C., Adve, V.: LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation // Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO'04). Palo Alto, California, 2004. DOI: 10.1109/CGO.2004.1281665
23. Doerfert, J., Streit, K., Hack, S., Benaissa, Z.: Polly's Polyhedral Scheduling in the Presence of Reductions // 5th International Workshop on Polyhedral Compilation Techniques (IMPACT), 2015.

24. Bakhtin, V.A., Kolganov, A.S., Krukov, V.A., Podderugina, N.V., Pritula, M.N. Methods of dynamic tuning of DVMH programs on clusters with accelerators // Russian Supercomputing Days: Proceedings of International conference (28–29 september 2015, Moscow), Moscow: Moscow University Press, 2015. P. 257–268
25. Aleksakhin, V.F., Bakhtin, V.A., Zhukova O.F., Zakharov D.A., Krukov V.A., Podderugina N.V., Savitskaya O.A. New Features of DVM System // Proceedings of the 21st Conference on Scientific Services & Internet, SSI '19, 2019. P. 13–25.
26. Гетерогенный кластер K60. URL: <https://www.kiam.ru/MVS/resources/k60.html>

# Исследование фазовых переходов в модели Поттса методом Ванг-Ландау \*

М.А. Фадеева

Национальный исследовательский университет Высшая школа экономики, Москва

Алгоритм Ванг-Ландау используется для вычисления плотности состояний энергии в задачах статистической механики. В статье излагается модификация алгоритма Ванг-Ландау, основанная на ранее введенном критерии точности. Матрица переходов между уровнями энергии стремится к стохастической при приближении плотности состояний к искомому значению. В качестве критерия точности было предложено использовать отклонение старшего собственного значения матрицы переходов между уровнями энергии от единицы. Модифицированный алгоритм Ванг-Ландау применен для прямого вычисления плотности состояний энергии при исследовании окрестности фазового перехода первого рода в модели Поттса на квадратной решетке с периодическими граничными условиями и числом компонент  $q=5$  и  $q=20$ .

*Ключевые слова:* плотность состояний, метод Монте-Карло, метод Ванг-Ландау, модель Поттса, фазовый переход первого рода.

## 1. Введение

Повышение вычислительной мощности компьютерных систем и разработка эффективных алгоритмов дают возможность проведения численных экспериментов с целью уточнения применимости гипотезы универсальности теории фазовых переходов [1], а также для выявления тонких закономерностей в физике фазовых переходов и критических явлений. Например, применение компьютерного моделирования позволило установить, что наличие анизотропии взаимодействий приводит к явной зависимости кумулянта Биндера от степени анизотропии [2, 3], что уточнило понимание универсальности значений этой величины. Этот численный результат получил подтверждение в аналитическом исследовании [4].

В то же время, ряд опубликованных утверждений основан на применении численных методов, имеющих ограниченную точность оценки значений термодинамических величин, что приводит к неконтролируемым результатам. В настоящей заметке мы обсуждаем модификации метода оценки плотности состояний – алгоритма Ванг-Ландау [7]. Этот метод прост в реализации и активно используется многими исследователями. Однако, на примере точно решаемых моделей выяснено, что этот метод приводит к конечной ошибке при оценке плотности состояний, которая может составлять несколько процентов [8]. Повышение точности вычислений было эвристически решено в работах [12, 13], в которых был предложен так называемый метод  $1/t$  (смотри раздел 3), позволяющий повышать точность вычислений, однако не дающий возможности оценить их точность. Эта проблема была решена в работе [15], в которой был предложен метод оценки точности за счет анализа старшего собственного значения вспомогательной матрицы переходов по уровням энергии (смотри раздел 4). В разделе 5 мы приводим обзор методов распараллеливания алгоритма Ванг-Ландау, который применим и для  $1/t$  модификации, и предлагаем модификацию метода разбиения на окна с использованием матрицы переходов по энергии.

Таким образом, мы имеем возможность проведения контролируемого по точности вычисления плотности состояний спиновых систем с дискретным спектром энергии. Мы применили разработанный метод для анализа оценки температуры перехода двумерных моде-

---

\*Вычисления проводились на высокопроизводительном вычислительном кластере "сHARISMa" (Computer of HSE for Artificial Intelligence and Supercomputer Modelling)  
Работа выполнена при поддержке гранта РФФИ «аспиранты» 20-37-90084.

лей Поттса с пятью и двадцатью компонентами, проявляющих фазовый переход первого рода. Мы проверили утверждение, сделанное авторами статьи [5] о том, что в точке перехода экстремумы плотности энергии пропорциональны энергии числу состояний  $q$  в модели Поттса, но не равно единице [6], как считалось на протяжении длительного времени. Таким образом, использование замечания, сделанного в статье [5] в совокупности с использованием изложенного в статье метода позволяет улучшить точность определения температуры фазового перехода первого рода.

## 2. Алгоритм вычисления плотности состояний

Алгоритм Ванг-Ландау [7, 8] относится к семейству методов Монте-Карло и напрямую вычисляет плотность состояний энергии  $g(E)$ . Алгоритм применим для любой системы, для которой статистическая сумма может быть представлена в следующем виде

$$Z = \sum_j e^{-E_j/k_B T} = \sum_{E_n} g(E_n) e^{-E_n/k_B T}, \quad (1)$$

где  $g(E_n)$  - количество состояний системы с энергией  $E_n$ ,  $T$  - температура,  $k_B$  - постоянная Больцмана. Вычисленная плотность состояний  $g(E)$  используется для вычисления термодинамических величин системы (теплоемкость, внутренняя энергия, восприимчивость), как функций температуры.

При некоторой конфигурации системы мы вычисляем энергию  $E_k$ . Далее случайным образом изменяем конфигурацию, считаем ее энергию  $E_m$  и оставляем с вероятностью

$$P_{WL} = \min \left[ 1, \frac{\tilde{g}(E_k)}{\tilde{g}(E_m)} \right], \quad (2)$$

где  $\tilde{g}(E_k)$  и  $\tilde{g}(E_m)$  текущие приближения плотности состояний энергетических уровней  $E_k$  и  $E_m$ . Эти приближения обновляются каждый раз, когда мы попадаем на конфигурацию системы с соответствующим энергетическим уровнем  $\tilde{g}(E_k) := f \tilde{g}(E_k)$ . Значения множителя  $f$  меняются каждый раз, когда гистограмма  $H(E)$  становится до некоторой степени “плоской”. При этом значения  $f$  меняются по правилу  $f_{i+1} = \sqrt{f_i}$ . Для повышения точности вычислений используются логарифмы величин  $S(E) = \ln(g(E))$  и  $F = \ln f$ , благодаря чему от произведения можно перейти к более простой операции сложения:  $\tilde{S}(E) := \tilde{S}(E) + F$ .

Несмотря на широкое применение, алгоритм обладает рядом ограничений.

Во-первых, гистограмма  $H(E_i)$  хранит в себе информацию о том, сколько раз был посещен каждый уровень энергии  $E_i$  и играет роль некоторого регулятора точности алгоритма, но явная зависимость между ровностью гистограммы и точностью вычислений не была выявлена [7, 8].

Во-вторых, это систематическая ошибка алгоритма. Итеративный характер алгоритма предполагает уточнение результата с увеличением количества шагов. Однако, начиная с некоторого момента, точность оценки плотности состояний выходит на константу, несмотря на рост числа шагов вычислений [12, 15].

## 3. Алгоритм 1/t

Была предложена модификация алгоритма [12], которая получила название 1/t-Wang-Landau. Ее теоретическое обоснование дано в работе [14]. Было предложено разделить выполнение алгоритма на два этапа. На первом этапе все происходит в точности, как в оригинальном алгоритме [7, 8], кроме замены условия ровности гистограммы на условие  $H(E_i) \neq 0$ , т.е. каждый уровень энергии  $E_i$  должен быть посещен хотя бы один раз. После выполнения условия  $F \leq N_E/t$ , алгоритм завершает первый этап. На втором этапе гистограмма больше не проверяется, а параметр  $F$  изменяется по другому закону  $F := N_E/t$ .

Здесь  $t$  - это время работы алгоритма, измеряемое в количестве попыток изменить конфигурацию системы, а  $N_E$  - некоторая константа. Позднее было показано, что ее можно принять за единицу  $N_E = 1$  [15]. На точно решаемой модели было проверено, что разница между вычисляемой плотностью состояний и известным аналитическим решением уменьшается обратно пропорционально времени вычислений  $\sim 1/t$ .

#### 4. Критерий точности в алгоритме Ванг-Ландау

Критерий точности необходим для исследования задач, в которых аналитическое решение отсутствует. В работах [15–17] было предложено вместо гистограммы ровности использовать матрицу переходов  $T$ . Алгоритм следующий. Матрица инициализируется нулями  $\tilde{T}$ , далее совершаются случайные блуждания Ванг-Ландау и, если совершается переход с уровня  $E_m$  на  $E_k$ , то элемент матрицы  $\tilde{T}(E_m, E_k)$  увеличивается на единицу  $\tilde{T}(E_m, E_k) + 1$ . Нормировка каждого элемента на сумму в столбце или строке  $T(E_m, E_k) = \tilde{T}(E_m, E_k)/N$  приводит к матрице, элементы которой  $T(E_m, E_k)$  будут давать оценку вероятности перехода с энергетического уровня  $E_m$  на уровень  $E_k$ . Причем, для недиагональных элементов эта вероятность может быть представлена в виде

$$T(E_m, E_k) = \min\left[1, \frac{\tilde{g}(E_k)}{\tilde{g}(E_m)}\right]P(E_m, E_k), \quad (3)$$

где первое слагаемое - это вероятность Ванг-Ландау принять новое состояние  $P_{WL}$  (выражение (2)), а второе  $P(E_m, E_k)$  - это вероятность из любой конфигурации системы с энергией  $E_m$  попасть в конфигурацию системы с энергией  $E_k$ . Можно показать [15], что матрица переходов является симметричной  $T(E_m, E_k) = T(E_k, E_m)$  и дважды стохастической, т.е. сумма ее элементов и по столбцам, и по строкам равна единице. При правильной организации вычислений с увеличением числа шагов матрица переходов должна приближаться к дважды стохастической, а старшее собственное значение  $\lambda_1$  такой матрицы равно 1. Отсюда вытекает, что отличие собственного значения матрицы переходов от единицы

$$\delta = |1 - \lambda_1|, \quad (4)$$

может быть использовано, как критерий сходимости алгоритма при  $\delta \rightarrow 0$ .

#### 5. Параллельный алгоритм Ванг-Ландау

На данный момент существует несколько подходов к решению задачи о модификации алгоритма для его эффективного распараллеливания. Условно эти подходы можно разделить на две группы: 1) независимые друг от друга блуждания происходят по отрезку спектра и 2) в каждой параллельной реплике происходит блуждание по всему спектру энергии.

В статьях [25–27] предлагается подход, в рамках которого весь спектр энергии делится на  $h$  частей (назовем их окнами), так, что доля их пересечений составляет  $x\%$ . Причем параметр  $x$  подбирается эмпирическим путем индивидуально для каждой задачи, и авторы предлагают отталкиваться от значения 75%. На каждом окне происходит  $m$  независимых Ванг-Ландау случайных блужданий: в рамках каждого блуждания существует своя гистограмма ровности  $H^{m,h}(E)$  и текущее приближение плотности состояний  $g^{m,h}(E)$ . После определенного числа шагов между двумя случайными блужданиями  $i$  и  $j$  происходит обмен текущими конфигурациями следующим образом: для  $i$ -того блуждания случайным образом выбирается случайное блуждание  $j$  среди соседних окон. Обмен конфигурациями происходит с вероятностью

$$P_{acc} = \min\left[1, \frac{g_i(E(X))}{g_i(E(Y))} \frac{g_j(E(Y))}{g_j(E(X))}\right] \quad (5)$$

где  $X$  конфигурация  $i$ -того случайного блуждания, а  $Y$  -  $j$ -того блуждания.  $E(X), E(Y)$  - соответствующие им уровни энергии.

После того, как внутри каждого случайного блуждания выполняется условие "ровнотности" гистограммы  $H(E)$ , значение  $g(E)$  на этом окне усредняется, чтобы избежать накопления ошибки, и происходит обновление параметра  $f$ . Алгоритм останавливается, когда на каждом окне случайное блуждание достигнет финального значения модификационного параметра  $f = f_{final}$ .

Итоговое значения плотности состояний на всем спектре формируется "из кусочков": точкой объединения любых двух интервалов является энергетический уровень  $E$ , где обратная микроканоническая температура  $\beta = d \log(g(E))/dE$  наилучшим образом совпадает.

В другом подходе [28] напротив, предлагается не делить спектр на части, а проводить  $n$  случайных блуждания по всему спектру так, что  $g(E)$ ,  $H(E)$  и значение  $f$  хранятся в общей памяти. В каждом ядре создается копия системы, происходят случайные блуждания, у которых есть право на чтение и запись  $g(E)$  и  $H(E)$ , но проверка "ровнотности" гистограммы и обновление параметра  $f$  происходит в рамках одного процесса. Так как значения гистограммы и плотности состояний являются общими для всех процессов, то возникает риск преждевременной перезаписи значений. Авторы решают эту проблему с помощью директив CRITICAL и ATOMIC из стандарта OpenMP.

Мы модифицировали метод окон [25] путем исключения вероятности обмена конфигурациями (выражение (5)) и введения шивки вычисленных в окнах плотностей энергии следующим образом. Пусть наш спектр энергий разбит на  $S$  пересекающихся интервалов. В каждом интервале  $i$  мы вычисляем плотность состояний по случайному блужданию с вероятностью Ванг-Ландау  $g_j$ . После определенного числа шагов вычисляется плотность состояний для всего спектра  $G_i$ , как взвешенная сумма интервальных оценок  $g_j$

$$G_i = \sum_{j=1}^N \frac{g_j}{|\lambda_j - 1|}, \quad (6)$$

где  $\lambda_j$  собственное значение блока матрицы переходов, соответствующего интервалу энергии  $j$ , при этом значения функции  $g_j$  в пересекающихся точках учитываются с коэффициентом  $1/2$ . Заметим, что после вычисления полной функции плотности состояний требуется проведение ее нормировки, что является обычной процедурой для метода оценки плотности состояний. Стохастичность блока матрицы гарантирована вблизи искомого решения для систем с дискретным спектром в случае, если интервал включает в себя число уровней, большее, чем ширина ленточной матрицы переходов. Эта величина легко оценивается для конкретной статистической модели с дискретным спектром. Такая модификация интервального метода увеличивает скорость сходимости плотности состояний к искомой [30].

## 6. Моделирование алгоритмом Wang-Landau

### 6.1. Модель Поттса

Гамильтониан в модели Поттса имеет вид [29]

$$H = - \sum \delta_{\sigma_i, \sigma_j}, \quad (7)$$

где спины  $\sigma_i$  могут принимать  $q$  различных значений и  $\delta$  - символ Кронекера. Мы исследовали модель Поттса на квадратной решетке с взаимодействием только ближайших соседей. Такая модель имеет фазовый переход первого рода при  $q \geq 5$ . Значение критической температуры известно из точного решения [24]

$$T_c = 1 / \ln(1 + \sqrt{q}). \quad (8)$$

Если обозначить за  $e_0 = E_0(\beta_c)/V$  значение внутренней энергии  $E(\beta) = -\frac{\delta Z(\beta)}{\delta \beta}$  как функцию обратной температуры  $\beta_c = 1/T$  при движении от высоких температур  $\beta < \beta_c \rightarrow \beta_c$ , а за  $e_d = E_d(\beta_c)/V$  при движении наоборот от низких температур  $\beta > \beta_c \rightarrow \beta_c$ , то известно [24], что их среднее определяется по формуле:

$$(e_0 + e_d)/2 = -(1 + 1/\sqrt{q}) \quad (9)$$

и их разница (скрытая теплота - latent heat) также известна точно [24]

$$\Delta e = e_d - e_0 = 2(1 + 1/\sqrt{q}) \tanh^2(\Theta/2) \prod_{n=1}^{\infty} \tanh^2(n\Theta) \quad (10)$$

где  $2 \cosh(\Theta) = \sqrt{q}$ .

## 6.2. Проведение моделирования

Моделирование проводилось по схеме  $1/t$ -алгоритма [12] Ванг-Ландау [7] с критерием точности [15]. Параметры моделирования:  $L$  - линейный размер квадратной решетки,  $q$  - количество компонент модели Поттса,  $steps_{min}$  - количество шагов  $t$  второго этапа алгоритма, это число шагов с изменением параметра  $f$  по алгоритму  $1/t$ . Типичное значение числа шагов  $steps_{min} = 10^6 \times L^2$ . Итоговая плотность состояний  $g(E)$  усреднялась по  $k = 10$  вычислениям со случайным начальным распределением спинов на решетке.

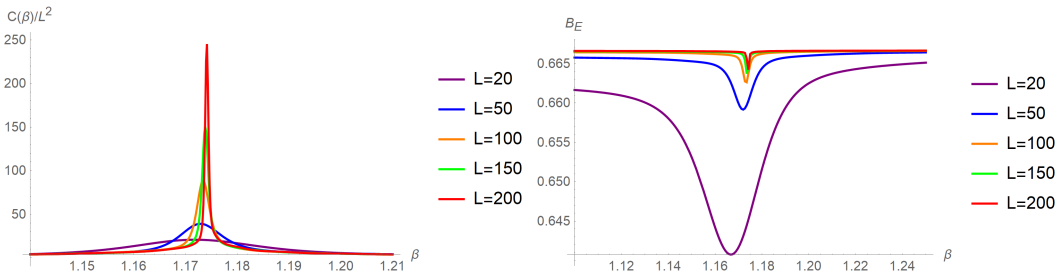
Численно исследовались теплоемкость

$$C(\beta, N) = \beta^2 N (\langle e^2 \rangle - \langle e \rangle^2) \quad (11)$$

и кумулянт Биндера [9]

$$B_E(\beta, N) = 1 - \frac{\langle e^4 \rangle}{3\langle e^2 \rangle^2} \quad (12)$$

Положение максимума теплоемкости с ростом размера системы  $L$  приближается к критическому значению  $\beta_c$  и кривая теплоемкости приобретает ярко выраженный пик при увеличении размера системы, что видно на рис. 1 слева. Положение экстремума кумулянта Биндера также стремится к критическому значению при увеличении размера системы. Известно, что величина сдвига экстремума от критического значения  $\Delta\beta$  обратно пропорциональна объему системы [6] и в нашем случае  $\Delta\beta \propto L^{-2}$ .

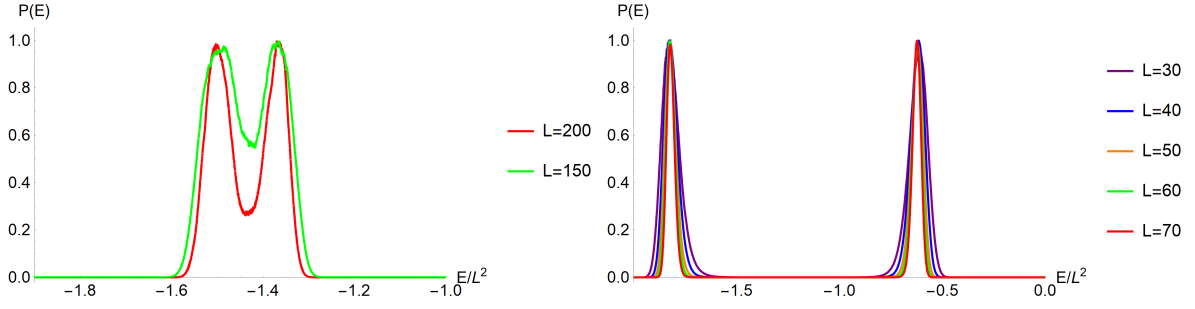


**Рис. 1.** График теплоемкости  $C(\beta, L)$  (слева) и график кумулянта Биндера  $B_E(\beta, L)$  (справа) для квадратной модели Поттса с числом компонент  $q = 5$  с периодическими граничными условиями на квадратной решетке с линейным размером  $L$ .

Фазовому переходу первого рода свойственно присутствие двух выраженных пиков на графике функции распределения энергии

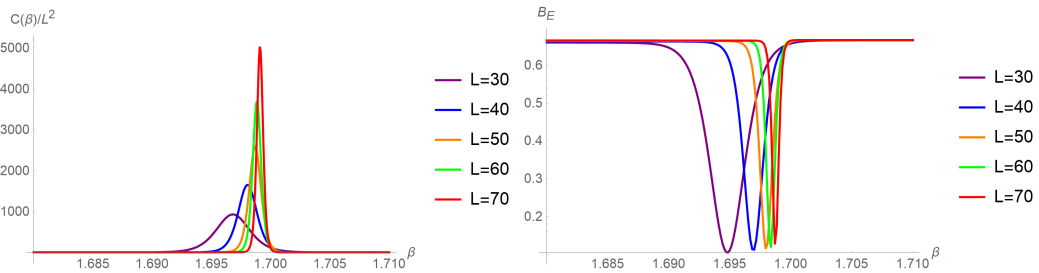
$$P_{\beta, L}(E) = g(E)e^{-E/k_B T} \quad (13)$$

при  $\beta = \beta_c$ , причем положения пиков соответствуют значениям энергии  $e_0$  и  $e_d$ , а расстояние между ними равно значению скрытой теплоты.



**Рис. 2.** Функция распределения энергии  $P_{\beta,L}(E)$  в точке  $\beta_0$ , при которой наблюдается два равных пика для квадратной модели Поттса с числом компонент  $q = 20$  с периодическими граничными условиями.

Результаты моделирования 5-компонентной модели Поттса для разных размеров решетки представлены на рис. 1. Для данной системы значение обратной критической температуры равно  $\beta_c = \ln(1 + \sqrt{5}) \approx 1,174$ . На графике видно, что с ростом размера решетки пик функции теплоемкости стремится к критической точке. Аналогичная тенденция наблюдается и на графике с зависимостью кумулянта Биндера от температуры для разных размеров: точки минимума также стремится к критической точке  $\beta_c \approx 1,174$ .



**Рис. 3.** График теплоемкости  $C(\beta, L)$  (слева) и график кумулянта Биндера  $B_E(\beta, L)$  (справа) для квадратной модели Поттса с числом компонент  $q = 20$  с периодическими граничными условиями

Таковую же тенденцию можно наблюдать и для 20-компонентной модели Поттса рис. 3: экстремум функции с ростом системы также сдвигается к критической температуре  $\beta_c = \ln(1 + \sqrt{20}) \approx 1,6997$ .

По графику теплоемкости можно заметить тенденцию, что с ростом системы пик кривой не только сдвигается к критической точке, но становится более ярко выраженным, и растет значение экстремума функции. В работе [6] выводится соотношение между значением максимума теплоемкости  $C_{max}$  и размером системы  $V = L^2$

$$C_{max} = V\beta_c^2(\Delta e/2)^2 \quad (14)$$

и определяется положение пика [6]:

$$\beta_{C_{max}} = \beta_c - \ln q/V\Delta e + \dots \quad (15)$$

Аналогичные выражения можно привести и для минимального значения кумулянта Биндера

$$B_{min} = 1 - (e_0/e_d + e_d/e_0)^2/12 \quad (16)$$

и

$$\beta_{B_{min}} = \beta_c - \ln(qe_0^2/e_d^2)/V\Delta e + \dots \quad (17)$$



Определим значение скрытой теплоты  $\Delta\tilde{\epsilon}$  из результатов моделирования следующим образом. Сначала найдем такое  $\beta_0$ , при котором наблюдаются равные значения пиков в графике функции распределения энергии  $P_{\beta_0, L}(E)$  (рис. 2 b) и определим значения оценки значений  $e_0$  и  $e_d$ .

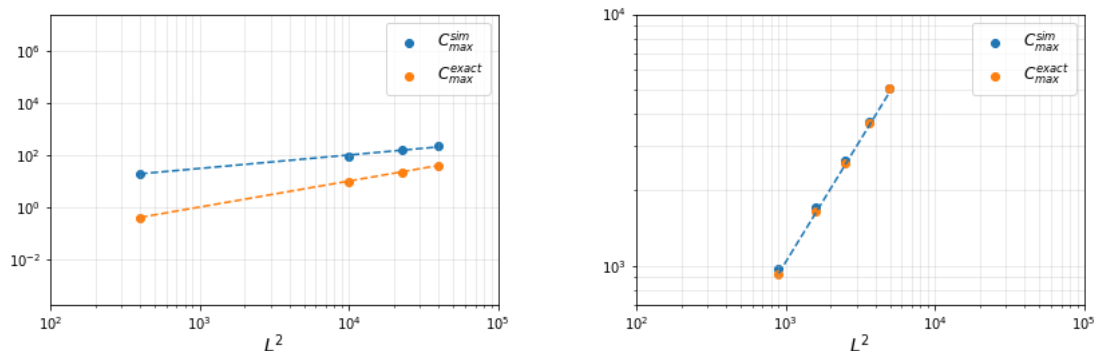


Рис. 4. График зависимости значения максимума теплоемкости от размера системы для модели Поттса с числом компонент  $q = 5$ (слева) и  $q = 20$  (справа)

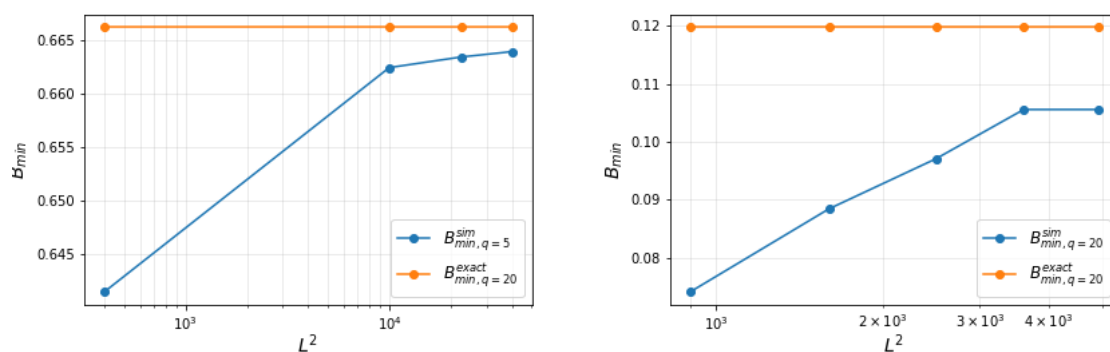


Рис. 5. График зависимости значения минимума кумулянта Биндера от размера системы для модели Поттса с числом компонент  $q = 5$ (слева) и  $q = 20$  (справа)

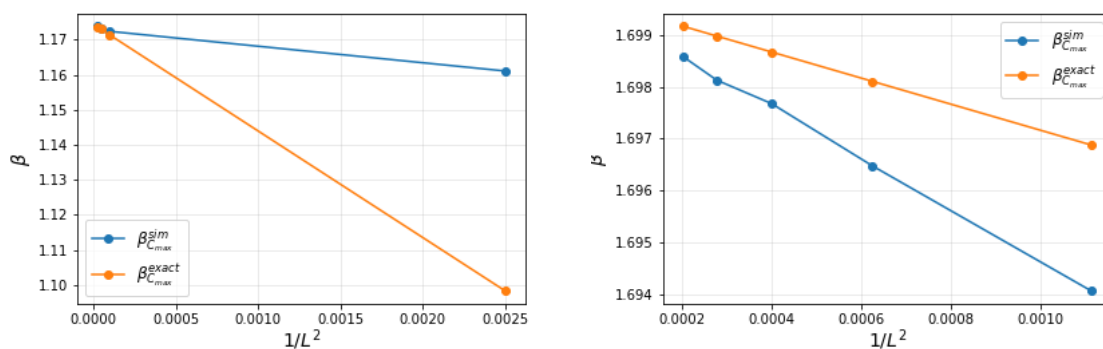
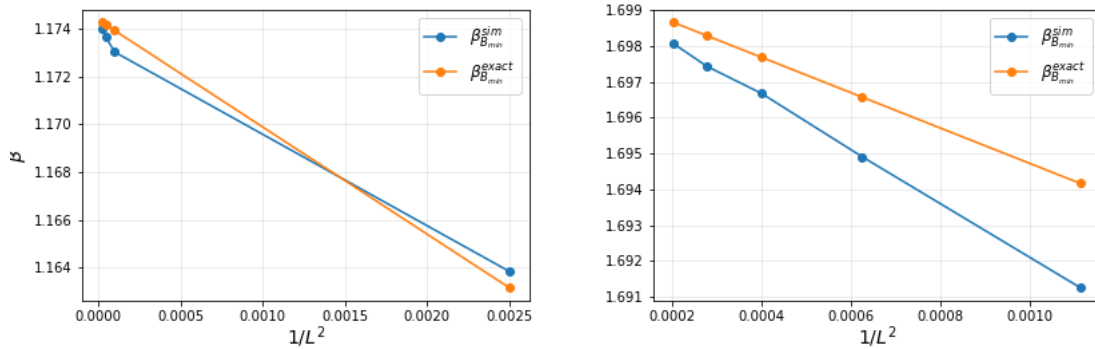


Рис. 6. График зависимости значения  $\beta_{C_{max}}$  в котором достигается максимум теплоемкости от размера системы для модели Поттса с числом компонент  $q = 5$ (слева) и  $q = 20$  (справа)

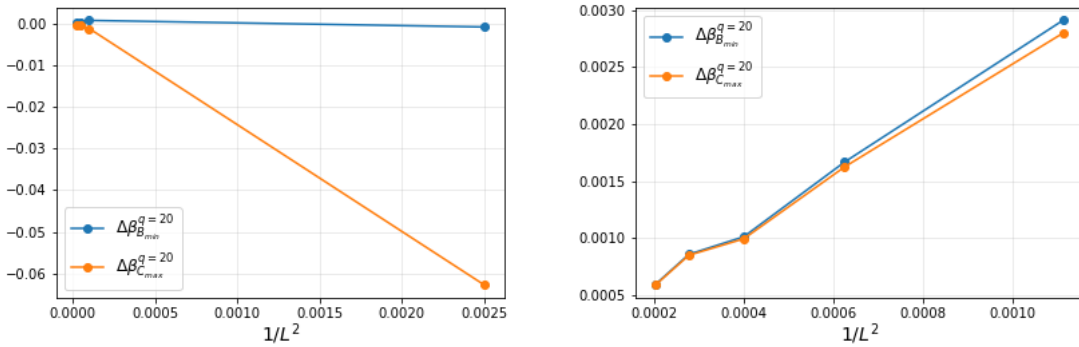
Затем подставим полученные значения  $\Delta\tilde{\epsilon}$  и  $\beta_0$  в уравнение 14. Результат отображен на рисунке 4. Для 5-компонентной модели наблюдается зависимость  $C_{max} \sim V^{0.5}$ , для 20-компонентной модели -  $C_{max} \sim V^{0.976}$ , что та близко к аналитическому расчету.

Значение минимума кумулянта Биндера по результатам моделирования также можно вычислить, подставив полученные значения  $\beta_0$ ,  $e_0$  и  $e_d$  в уравнение 16. Результаты представлены на рис. 5. Значение минимума кумулянта Биндера для бесконечно большой решетки является константой. По графику видно, что для обеих систем  $q = 5$  и  $q = 20$  с ростом размера решетки значение приближается к искомой константе.

Повторим аналогичное упражнение для значения температуры, при которой достигается максимум теплоемкости  $\beta_{C_{max}}$  (уравнение (15)) и минимум кумулянта Биндера  $\beta_{B_{min}}$  (уравнение(17)). Результаты отображены на рис. 6 и рис.7. Как видно из графиков, значение температуры, при которой наблюдается максимум теплоемкости и минимум кумулянта Биндера во всех случаях стремятся в истинному значению при увеличении размера решетки. Это также видно на графике отклонения от точного значения, выражение (8).



**Рис. 7.** График зависимости значения  $\beta_{B_{min}}$  при котором наблюдается минимум кумулянта Биндера от размера системы для модели Поттса с числом компонент  $q = 5$ (слева) и  $q = 20$  (справа)



**Рис. 8.** График отклонения обратной температуры  $\beta_{C_{max}}^{sim}$  и  $\beta_{B_{min}}^{sim}$  от точного значения  $\Delta\beta = \beta_{exact} - \beta_{sim}$  в зависимости от размера системы для модели Поттса с числом компонент  $q = 5$ (слева) и  $q = 20$  (справа)

Проверим соотношение :

$$r_c = \frac{\sum_{e < e_c} P_{\beta,L}(e)}{\sum_{e \geq e_c} P_{\beta,L}(e)} \quad (18)$$

Пиковое соотношение связано с разностью свободных энергий упорядоченных и неупорядоченных фаз. Поскольку существует  $q$  разных упорядоченных фаз, то точное значение пикового соотношения в пределе термодинамики равно  $q$  при температуре перехода  $\beta_c$  [31]. За  $e_c$  была принята средняя энергия, вычисленная по формулам (9) и (10), за  $\beta = \beta_c$ . Результаты моделирования представлены в таблиц, рис. 9

Автор благодарит Л.Н. Щура за научное руководство и помощь в подготовке рукописи.

L	$\epsilon_c = \frac{q_0 + \epsilon_d}{2} = -\left(1 + \frac{1}{\sqrt{q}}\right)$	$\beta_c$	$r_c$
20	-1,447213595	1,174359006	1.31013
50	-1,447213595	1,174359006	1.595096
100	-1,447213595	1,174359006	3.338584
150	-1,447213595	1,174359006	4.141748
200	-1,447213595	1,174359006	3.467259

L	$\epsilon_c = \frac{q_0 + \epsilon_d}{2} = -\left(1 + \frac{1}{\sqrt{q}}\right)$	$\beta_c$	$r_c$
30	-1,22361	1,69967	21.7514463248
40	-1,22361	1,69967	22.4148142337
50	-1,22361	1,69967	19.0111971469
60	-1,22361	1,69967	36.493148975
70	-1,22361	1,69967	29.017965312

**Рис. 9.** Соотношения величин пиков в модели Поттса с числом компонент  $q = 5$  (слева) и  $q = 20$  (справа)

## Литература

1. Phase transitions and critical phenomena, Edited by C. Domb and J.L. Lebowitz, Elsevier (2001).
2. W. Selke, L.N. Shchur, Critical Binder cumulant in a two-dimensional anisotropic Ising model with competing interactions, Phys. Rev. E 80, 042104 (2009)
3. W. Selke, L.N. Shchur, Critical Binder cumulant in two-dimensional anisotropic Ising models, J. Phys. A 38(44), L739-L744 (2005).
4. V. Dohm, Diversity of critical behavior within a universality class, Phys. Rev. E 77 061128 (2008).
5. N. Rose and J. Machta, Equilibrium microcanonical annealing for first-order phase transitions, Phys. Rev. E 100 063304 (2019).
6. Janke W. First-order phase transitions //Computer Simulations of Surfaces and Interfaces. – Springer, Dordrecht, 2003. – С. 111-135.
7. Wang F., Landau D. P. Efficient, multiple-range random walk algorithm to calculate the density of states //Physical review letters. – 2001. – Т. 86. – №. 10. – С. 2050.
8. Wang F., Landau D. P. Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram //Physical Review E. – 2001. – Т. 64. – №. 5. – С. 056101.
9. Taylor M. P., Paul W., Binder K. Phase transitions of a single polymer chain: A Wang–Landau simulation study //The Journal of chemical physics. – 2009. – Т. 131. – №. 11. – С. 114907.
10. Gervais C. et al. Application of the Wang–Landau algorithm to the dimerization of glycoporphin A //The Journal of chemical physics. – 2009. – Т. 130. – №. 21. – С. 06B609.
11. Zhou C., Bhatt R. N. Understanding and improving the Wang-Landau algorithm //Physical Review E. – 2005. – Т. 72. – №. 2. – С. 025701.
12. Belardinelli R. E., Pereyra V. D. Fast algorithm to calculate density of states //Physical Review E. – 2007. – Т. 75. – №. 4. – С. 046701.
13. Belardinelli R. E., Pereyra V. D. Wang-Landau algorithm: A theoretical analysis of the saturation of the error //The Journal of chemical physics. – 2007. – Т. 127. – №. 18. – С. 184105.
14. Liang F., Liu C., Carroll R. J. Stochastic approximation in Monte Carlo computation
15. L. Yu. Barash, M. A. Fadeeva, and L. N. Shchur, Control of accuracy in the Wang–Landau algorithm, Phys. Rev E 96, 043307 (2017).

16. M. Fadeeva and L. Shchur, On the mixing time in the Wang–Landau algorithm, *J. Phys.: Conf. Ser.*, 955, 012028 (2018).
17. L.N. Shchur, On properties of the Wang–Landau algorithm, *J. Phys.: Conf. Ser.*, 1252, 012010 (2019).
18. P. Dayal, et al, Performance limitations of flat-histogram method, *Phys. Rev. Lett.* 92, 097201 (2004).
19. S. Boyd, P. Diaconis, L. Xiao, Fastest mixing Markov chain on a graph, *SIAM Review*
20. Xu J., Ma H. R. Density of states of a two-dimensional X Y model from the Wang-Landau algorithm //Physical Review E. – 2007. – T. 75. – №. 4. – C. 041115.
21. Wolff U. Collective Monte Carlo updating for spin systems //Physical Review Letters. – 1989. – T. 62. – №. 4. – C. 361.
22. Kasteleyn P. W., Fortuin C. M. Phase transitions in lattice systems with random local properties //PSJJS. – 1969. – T. 26. – C. 11.
23. Wu F. Y. The potts model //Reviews of modern physics. – 1982. – T. 54. – №. 1. – C. 235.
24. Baxter R. J. Potts model at the critical temperature //Journal of Physics C: Solid State Physics. – 1973. – T. 6. – №. 23. – C. L445.
25. Vogel T. et al. Generic, hierarchical framework for massively parallel Wang-Landau sampling //Physical review letters. – 2013. – T. 110. – №. 21. – C. 210603.
26. Vogel T. et al. Exploring new frontiers in statistical physics with a new, parallel Wang-Landau framework //Journal of Physics: Conference Series. – IOP Publishing, 2014. – T. 487. – №. 1. – C. 012001.
27. Li Y. W. et al. A new paradigm for petascale Monte Carlo simulation: Replica exchange Wang-Landau sampling //Journal of Physics: Conference Series. – IOP Publishing, 2014. – T. 510. – №. 1. – C. 012012.
28. Zhan L. A parallel implementation of the Wang–Landau algorithm //Computer Physics Communications. – 2008. – T. 179. – №. 5. – C. 339-344.
29. Potts R. B. Some generalized order-disorder transformations //Mathematical proceedings of the cambridge philosophical society. – Cambridge University Press, 1952. – T. 48. – №. 1. – C. 106-109.
30. M. Fadeeva and L. Shchur, in preparation.
31. Rose N., Machta J. Equilibrium microcanonical annealing for first-order phase transitions //Physical Review E. – 2019. – T. 100. – №. 6. – C. 063304.

## Компетенции и подготовка кадров в области суперкомпьютерных технологий

Ю.Я. Болдырев, В.В. Глухов, О.А. Картавенко

Санкт-Петербургский политехнический университет Петра Великого

В работе сделана попытка наметить направления исследований роли и места компетенций в решении проблемы подготовки кадров высшей технической школой в области суперкомпьютерных технологий. При этом рассматривается существо подходов к этой проблеме для формирующегося в России технологического сектора на базе суперкомпьютерных технологий. Главное внимание уделяется подготовке кадров для инженерного корпуса, формирование и рост квалификации которого решающим образом влияет на развитие страны. Следуя традициям Санкт-Петербургского политехнического университета, в своем рассмотрении авторы опираются на развитие фундаментальных основ естественнонаучного знания, которые всегда служили базой развития инженерного анализа и проектирования, при этом акцент делается на подготовку разработчиков новой техники, - инженеров разработчиков и элиту инженерного корпуса, - инженеров исследователей. В центре нашего внимания находятся важнейшие технологии исследования, как явлений природы, так и проведения инженерных разработок, - методы естественнонаучного анализа, которые со второй половины прошедшего века, на основе вычислительных машин, приобрели ярко выраженные черты технологий математического моделирования. Мы пытаемся показать, что суперкомпьютерные технологии, являясь ядром передовых разработок и исследований, должны входить в инструментарий авангарда инженерного сообщества, а значит приобретение компетенций в сфере, как технологий математического моделирования, так и в суперкомпьютерных технологиях, становится все более актуальным.

*Ключевые слова:* инженерные кадры, компетенции, естественнонаучное знание, математическое моделирование, суперкомпьютерные технологии.

К теме подготовки кадров в области суперкомпьютерных технологий мы обращались на нашей конференции в 2019 г. в контексте взаимосвязи между суперкомпьютерными технологиями и цифровой экономикой [1]. Здесь же мы попытаемся посмотреть на проблему подготовки кадров для суперкомпьютерных технологий с некоторых системных позиций, а именно с точки зрения взгляда на эту проблему через призму компетенций, которые хотя и медленно, но входят в обиход высшей школы.

Сначала несколько слов о самом понятии компетенции, введенным в широкий научный обиход в 80-е годы прошлого столетия американским ученым В. Макелвиллом и развитым французским социологом Г. Каннаком показавшим, что для успеха организации необходимо повышать компетенцию каждого работника. Понятие компетенция была исходно определено как круг проблем, сфера деятельности, в которой данный человек обладает знанием и опытом; совокупность полномочий, прав и обязанностей должностного лица, общественной организации.

Заметим, что компетенции ранее, еще в 70-е годы, рассматривали как способность эффективно выполнять определенные действия и увязывали с качеством успешных профессионалов. Так в глоссарии терминов Европейского фонда образования компетенция определяется как:

- способность делать что-либо хорошо или эффективно;
- соответствие требованиям, предъявляемым при устройстве на работу;
- способность выполнять особые трудовые функции.

Таким образом компетенции увязывались не столько со знанием и опытом, что очевидно подразумевалось, а со способностью качественно выполнять работу. Здесь отметим, что в ряде работ указывается, что «Организации все чаще обнаруживают, что успех зависит от наличия

компетентных сотрудников. Оплата по уровню компетентности означает в этом случае, что организации ориентируется на будущее, а не на прошлое» [2]

Такое определение и понимание компетенции приводит нас к тому, что объективно приобретение их совокупности будущим выпускником есть центральная задача высшей школы. Но тут же встает вопрос о том круге – наборе компетенций, которые необходимы выпускнику, заканчивающему вуз по тому или иному направлению. При этом, важно заметить, что при создании учебных планов по направлениям подготовки на основе построения совокупности компетенций, мы вообще говоря, можем придать некоторый системный, количественный характер содержанию самих этих учебных планов, если в свою очередь, найдем способ придать компетенциям определенный количественный характер.

Укажем, что компетенциям и построенных на них подходах в учебном процессе, достаточно давно и активно уделяется внимание отечественными исследователями, в том числе и в части оценки некоторых величин, описывающих качественные, а в ряде случаев и количественные характеристики реализации выпускниками сформированных при учебе в вузе знаний, умений и навыков [3,4].

Круг компетенций, приобретаемых в вузах, крайне широк и более того заявленные нами к рассмотрению суперкомпьютерные технологии являются и с каждым днем становятся все более «тотальными». Под этим мы понимаем то, что они носят, во-первых, междисциплинарный (мультидисциплинарный) характер и являются универсальным вспомогательным инструментом для изучения и преобразования Природы, к каковым относится и сама деятельность инженерного сообщества. И, во-вторых, суперкомпьютерные технологии к настоящему времени охватили практически все отрасли индустрии и все стороны деятельности человеческого сообщества. Поэтому в нашей краткой работе мы ограничимся высшей технической школой и попытаемся сформировать некоторые позиции в части компетенций, которые важны для освоения студентами суперкомпьютерных технологий. Подчеркнем, что мы находимся только в начале пути в изучении данной темы.

И на этом пути мы сразу же сталкиваемся с первой серьезной трудностью, которая связана с принципами реализации математической подготовки инженерных кадров. Мы говорим о математической подготовке потому, что без таковой вообще бессмысленно говорить о суперкомпьютерных технологиях. Среди упомянутых принципов выделим два, ключевых для дальнейшего рассмотрения. Во-первых, это *принцип учета основных типов деятельности инженерных кадров, определяемых характером их будущей деятельности*. И, во-вторых, *принцип нацеленности математической подготовки инженеров на внедрение и развитие на всех этапах производства передовых производственных технологий*.

Остановимся на этих двух ключевых принципах подробнее с позиций нашего компетентного подхода. Что касается *основных типов деятельности* инженеров, то здесь мы можем указать три главных направления инженерной подготовки, принятой в нашем Политехническом университете с момента его основания. Это направления, ориентированные на подготовку:

- «инженеров по эксплуатации», характер работы которых эксплуатация машин, разнообразных систем и технологических линий;
- «инженеров массовых технических специальностей» или «инженеров-разработчиков», ориентированных, как на проектирование и конструирование новой техники и технологий, так и на вопросы обеспечения их эффективной эксплуатации и функционирования;
- «инженеров – исследователей», ориентированных на разработку, становящихся все более высокотехнологичными передовых машин, систем и технологий.

Оценка количества выпускников в каждой из названных групп составляет примерно 15%, 70 и 15% от их общего количества соответственно.

Переходя к *принципу нацеленности* математической подготовки инженеров на внедрение и развитие на всех этапах производства передовых производственных технологий, следует указать, что этот принцип является центральным с позиций внедрения суперкомпьютерных технологий в передовую индустрию. Действительно, фундаментальной основой всей передовой инженерной деятельности является естественнонаучное знание, а суперкомпьютерные (компьютерные) технологии имеют математическое моделирование [5,6] своей базой, что, увы еще не является признанной нормой. И здесь следует указать, например, на то обстоятельство, что получившее сего-

дня широкое хождение понятие *цифрового двойника* есть не что иное, как математическая модель изделия или его элемента, некоторой системы или технологии, реализованная в каком-либо программном комплексе или их в совокупности на той или иной вычислительной системе.

Теперь кратко остановимся на описании концептуальных основ математического моделирования и приведем основные составляющие-блоки этой концепции [6], с учетом её основополагающей значимости, как для рассматриваемой нами темы, так и для образовательного процесса современной высшей школы.

При этом сразу же укажем на то обстоятельство, что для построения как вертикальной иерархии компетенций, так и, если так можно выразиться их «слоев» на некоторых уровнях, математическое моделирование критически важно.

Приведем в сжатой форме основные позиции концепции математического моделирования, следуя подходу А.А. Самарского.

I блок. Составление математической модели явления, процесса, задачи и т.д. Это предметная область естественных и других наук, где составляются количественные соотношения для описываемых явлений и процессов. И это труднейшая область научной деятельности, являющаяся основой, тех же естественных наук. Выделяя основное, - этот блок составляет существо некоторой «замены» реального физического процесса или явления его математической моделью, то есть описание всех его свойств-характеристик с помощью математических соотношений.

II блок. Анализ математической корректности построенной математической модели, описывающей нашу задачу. Весь спектр рассматриваемых здесь проблем носит сугубо математический характер и в подавляющем большинстве обходится исследователями и инженерами.

Вообще, решение проблемы математической корректности той или иной задачи включает в себя решение следующих ключевых подзадач:

- существования решения (решений) в том или ином классе математических объектов (векторов, функций, целых или вещ. чисел и т.д.);
- единственность решения в этом классе;
- устойчивость решения по отношению к возмущению параметров задачи.

III блок. Переход от непрерывной математической модели к модели дискретной. Отметим, предварительно, что в ряде задач этот блок может отсутствовать, то есть, он, как правило, характерен для таких моделей - явлений, которые описываются дифференциальными и интегральными соотношениями. Таким образом, мы, по сути дела, переформулируем нашу задачу и получаем некоторую новую, где присутствует один или несколько параметров, характеризующих новую дискретную задачу и которых не было в исходной задаче.

IV блок. Анализ математической корректности вновь полученной дискретной задачи (на основе исходной). По сути дела, это и есть предмет вычислительной математики. Здесь рассматриваются те же проблемы, что и в блоке II, к которым добавляется еще одна задача, - весьма важная. Решение этой задачи должно ответить на вопрос, - будет ли построенная в блоке III дискретная задача, стремиться к исходной непрерывной задаче?

V блок. Написание алгоритма для дискретной задачи, то есть последовательности вычислительных шагов и его перенос на компьютер, - программирование. Этот блок носит совершенно особый характер и его роль, со времени внедрения вычислительных машин в научную и инженерную практику, непрерывно нарастала.

VI блок. Отладка программы, то есть её тестирование, получение результатов и их анализ. Назначение этого блока вполне очевидно и поэтому оставим его без комментариев.

Изложенная схема, конечно, является очень общей и её применение в полном объеме отдельным исследователем при решении какой-либо задачи, в общем, не характерно и является, как правило, функцией научной группы или коллектива исследователей. Но с позиций тех вопросов, которые мы здесь рассматриваем, концепция математического моделирования принципиально важна. При этом в том виде, в котором она здесь приведена, она не только не изучается, но и не упоминается в высшей школе, хотя именно эта концепция служит стрессоустойчивой основой, для вводимых ниже, как универсальных, так и профессиональных компетенций.

Теперь кратко остановимся на текущем состоянии в применении компетенций, и основанном на них компетентностном подходе в деятельности современной высшей школы. В основополагающем документе, в методических рекомендациях «Проектирование основных образователь-

ных программ, реализующих федеральные государственные образовательные стандарты высшего образования» (2010 г.) в структуру программ введены экспериментальные программные документы:

- паспорта и программы формирования профессиональных компетенций;
- таблица содержательно-логических связей учебных курсов, предметов, дисциплин, модулей, практик;
- компетентно - ориентированная часть учебного плана;
- сквозная программа комплексных испытаний на соответствие подготовки ожидаемым результатам образования;
- программы итоговых комплексных испытаний.

Теперь о принятой на сегодня системе классификации компетенций. Их разделяют на три большие группы:

- общие или общекультурные,
- универсальные,
- профессиональные.

*Универсальные компетенции* непосредственно связаны с умением применять знания в профессиональной деятельности, тогда как *профессиональные компетенции* выражают способность выпускника осуществлять те или иные виды профессиональной деятельности.

Ключевые компетенции современного инженера относятся к сфере фундаментальных основ инженерного знания и относятся к классам универсальных и профессиональных. Памятуя о том, что предназначение труда и, труда инженера, в частности, - преобразование Природы в интересах человека и всего человеческого сообщества, то безусловно краеугольным камнем здесь является естественнонаучное знание: математика, физика, механика и химия, знание которых мы безусловно должны отнести к универсальным компетенциям. При этом глубокое знание самих этих ключевых дисциплин, как и знание тех или иных их разделов очевидно должны быть отнесены к профессиональным компетенциям.

Нам же нужно попытаться связать естественнонаучное знание, с его поистине бесчисленным спектром направлений, с суперкомпьютерными технологиями, которые, как указывалось выше, имеют своей базой, математическое моделирование. И, здесь отметим следующее важнейшее обстоятельство, - с учетом того, что технологии математического моделирования приобрели характер универсального инструментария, представляется, что эти технологии необходимо рассматривать как ключевую универсальную компетенцию в инженерном образовании. Естественно, что в первую очередь это касается «инженеров-разработчиков» и «инженеров – исследователей».

Но как концептуально связать приобретение знаний в естественных науках и в методологиях математического моделирования (частью которых и являются суперкомпьютерные технологии) с компетентным подходом в подготовке инженеров?

И здесь нам необходимо посмотреть, а как реализуется компетентный подход в вузах страны, причем не только в подготовке инженеров. При этом, так как тематика компьютерных технологий, а тем более суперкомпьютерных технологий системно никак не отражена в направлениях подготовки инженеров в России, мы должны посмотреть на наиболее близкие здесь направления. Таким, очевидно, является направление подготовки магистров «Прикладная математика и информатика» (направление 01.04.02), однако, к сожалению, рассматриваемая здесь тема компетенций не представлена достаточно широко такими ведущими вузами в части суперкомпьютерных технологий, как МГУ им. М.В. Ломоносова и ННГУ им. Н.И. Лобачевского. Вместе с тем весьма интересно и содержательно структура компетенций выпускника по магистерской программе «Компьютерные технологии инжиниринга» направления «Информатика и вычислительная техника» представлена в Санкт-Петербургский государственном электротехническом университете (ЛЭТИ им. В.И. Ульянова (Ленина) [7].

Теперь обратимся к нашему опыту, - опыту СПбПУ, где нам безусловно, наиболее интересен блок *профессиональных компетенций* (ПК), при подготовке магистров по направлению «Прикладная математика и информатика» (направление 01.04.02), содержащий четырнадцать позиций ПК-1, ПК-2, ..., ПК-14. Разработчики разделили их на восемь направлений – приложений: *научная и научно-исследовательская деятельность, проектная и производственно-технологическая*



*деятельность и др.* Выделение именно этих направлений обусловлено тем, что они в наибольшей степени находятся на острие передовых позиций научно – технического развития. И здесь, уместно, еще более сузить постановку нашей задачи, а именно рассмотреть формирование системы компетенций для той группы инженеров, которую ранее мы отнесли к инженерам – исследователям. При этом очевидно, что именно для этой группы изучение суперкомпьютерных технологий наиболее актуально.

А теперь попробуем конкретизировать систему компетенций как некоторую совокупность целевых функций для освоения их студентами. И здесь мы должны обратиться к такой последовательности: инжиниринг- компьютерные технологии- суперкомпьютерные технологии, где под инжинирингом понимается весь спектр инженерной деятельности, в той части, которую мы отнесли к деятельности инженеров – исследователей. При этом мы должны попытаться понять, во – первых, а чем же принципиально отличаются суперкомпьютерные технологии от компьютерных, применительно к исследовательской деятельности? Или обращаясь к компетенциям, - какие новые компетенции привносят суперкомпьютерные технологии в учебный процесс относительно компьютерных?

Эти вопросы оказываются не такими сложными, поскольку универсальное отличие рассматриваемых технологий известно специалистам в области высокопроизводительных вычислений с самого момента их появления. Это отличие реализуется посредством параллельных вычислительных технологий [8,9], которые физически реализуются в конструкциях суперкомпьютеров в виде множества объединенных процессоров, а также, в последние, примерно 15 лет, и с использованием многоядерности процессоров. Но эта особенность суперкомпьютеров является чисто конструктивной и, вообще говоря, никак затрагивает проблемы подготовки инженерного корпуса, исключая сравнительно узкую прослойку инженеров - разработчиков суперкомпьютеров, а также системного и прикладного ПО к ним.

И здесь возникает следующий вопрос о том, а где инженеры-исследователи могут столкнуться в своей деятельности с параллельными вычислениями? Это, во-первых, использование программных комплексов для инженерных расчетов и, во – вторых, разработка ПО для решения собственных, то есть поставленных ими задач, с использованием суперкомпьютеров. Очевидно, что глубокие знания в области параллельных вычислений нужны в первую очередь последней группе инженеров, - разработчиков прикладного ПО, тогда как для массового корпуса инженеров-исследователей они не находятся на первом плане. Таким образом к разряду *профессиональных компетенций* в части содержащей направления – приложения по *научной и научно-исследовательской деятельности* мы должны отнести «...*компетенции мирового уровня в области передовых параллельных вычислительных технологий...*».

Эта компетенция актуальна и важна и, безусловно, должна стать одной из ключевых при подготовке передового отряда инженеров – исследователей.

Вместе с тем непрерывно растущий уровень в постановках задач реальной инженерной практики, связанный, как правило, с потребностями в описании сложных процессов в Природе, междисциплинарных (мультидисциплинарных) и многомасштабных<sup>1</sup> процессов в изделиях, системах и технологиях придает совершенно новую значимость суперкомпьютерным технологиям. Здесь имеется в виду то, что только они, как уже указывалось, являясь универсальным и уникальным инструментарием, позволяют решать такого рода задачи. Но отсюда мы можем сделать вывод о том, что освоение самих подходов к постановке и решению междисциплинарных задач, прямо связанных с суперкомпьютерными технологиями, является назревшей проблемой в учебном процессе и должно быть отнесено к категории *профессиональных компетенций*. И здесь следует продумать формирование направления подготовки, которое естественно назвать «Суперкомпьютерный инжиниринг» [10], при этом возникает крайне важная и интересная задача баланса специальных дисциплин области применения и профильных суперкомпьютерных дисциплин. Думается, что такое направление подготовки будет служить продолжением и развитием направления «Компьютерный инжиниринг» [11], формирование которого необходимо было реализовать уже достаточно давно.

---

<sup>1</sup> Под многомасштабными мы понимаем задачи, в которых геометрические масштабы процессов могут различаться на многие порядки.

Вместе с тем названная только что тема междисциплинарных проблем, и развитие в этом направлении проблематики суперкомпьютерных технологий заставляет совершенно по-новому посмотреть на сами концептуальные основы решения подобного рода задач, которыми изобилуют природные явления и каждая глубокая научно-техническая разработка. Более того, в наше время, мы, по-видимому, находимся в «точке роста и развития» совершенно иного взгляда не только на развитие производства в человеческом сообществе, но и на начальном этапе его реализации. И все это, безусловно потребует серьезной, если не кардинальной смены самого существа учебного процесса в высшей школе, а стало быть и всей совокупности компетенций в этом процессе. Попробуем посмотреть на эту проблему с позиции формирования среды «Суперкомпьютерного инжиниринга», памятуя о том, что само понятие *инжиниринга* есть некоторая область человеческой деятельности и связанных с ней процессов, существо которых разработка и проектирование изделий и их элементов, создание технических и иных систем или технологий, разработка систем по поддержанию их жизненного цикла вплоть до переработки или утилизации.

Неустанно подчеркивая, что фундаментом этой деятельности является современное математическое моделирование на основе суперкомпьютеров, укажем, что направления Суперкомпьютерного инжиниринга уже сегодня для своего перечисления требуют привести не один десяток наименований, от авиакосмического и механического инжиниринга до био-инжиниринга и инжиниринга в области продуктов питания.

Но инженерные задачи, возникающие здесь, являются как правило междисциплинарными в своей постановке и требуют очень больших вычислительных ресурсов, поскольку в их основе лежит постановка совокупности связанных начально краевых или краевых задач математической физики. Например, задачи горения, – это связка задач аэрогидродинамики, тепло и массообмена, излучения и, наконец, физико – химической кинетики.

В целом же в самое ближайшее время в индустриальной среде должны появиться такие среды Суперкомпьютерного инжиниринга [10], которые будут решать весь спектр задач производственного цикла с учетом междисциплинарности, многомасштабности и множества технологических процессов производства. Эти подходы в свою очередь будут способствовать переходу на новый качественный уровень самих процессов инженерного анализа и проектирования на основе оптимизации конструкций и узлов, включая многокритериальную оптимизацию и оптимизацию технологических процессов. Естественно, что в такого рода классах задач без суперкомпьютеров не обойтись. На этих путях уже происходит создание сред проектирования и оптимизации с расширением до производства продукции и переходом к виртуальной разработке изделий с использованием суперкомпьютеров. Последние технологии ведут к разработке «цифровых прототипов» – виртуальной цифровой пространственной (реальной) модели изделия и всех его компонентов, позволяя исключить из процесса разработки создание дорогостоящих натуральных моделей-прототипов, и предоставляя разработчикам точно оценивать и моделировать любые характеристики объекта в любых условиях эксплуатации. А это уже путь к «цифровому производству» – как основному компоненту передовых заводов и фабрик.

С позиций рассматриваемого здесь компетентного подхода есть еще одно обстоятельство, изучение которого с позиций учебного процесса в вузах представляет значительный интерес. Речь идет о том, что в последние десятилетия стала все более и более размываться граница между естественнонаучным и инженерным знанием. Трудность инженерных задач, рассматриваемых, например, в аэрогидродинамике, такова, что непосредственно связана с теоретическими проблемами разработки моделей турбулентности. И с позиций характера компетенций необходимо продумать как соотносятся здесь универсальные и профессиональные компетенции.

## Литература

1. Болдырев Ю.Я., Боровков А.И., Глухов В.В., Картавенко О.А. Некоторые вопросы подготовки кадров в области суперкомпьютерных технологий для цифровой индустрии В сборнике: Суперкомпьютерные дни в России труды международной конференции. Москва, 2019. С. 165-174.
2. Чупланова О.Л. Применение компетентного подхода при разработке системы оплаты труда. Интернет-журнал. Науковедение. № 6. 2014.

3. Байденко В.И. Компетентный подход к проектированию государственных образовательных стандартов: методическое пособие /В.И. Байденко. - М., 2005.
4. Дорошенко В.А. Концепции современного естествознания: учебно-практическое пособие для студентов экономико-управленческих специальностей всех форм обучения/В.А.Дорошенко, М.Р. Москаленко. - Екатеринбург, 2009.
5. Самарский А. А., Михайлов А. П. Математическое моделирование: Идеи. Методы. Примеры. — 2-е изд., испр. - М.: Физматлит, 2001 - 320 с. - ISBN 5-9221-0120-X.
6. Болдырев Ю.Я. История и методология науки (на примере прикладной математики и механики). Учебное пособие. Санкт-Петербург. Издательство политехнического университета. 2019, 216 с.
7. <https://etu.ru/>
8. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. Санкт - Петербург, «ВНУ», 2002, 599 с.
9. Гергель В.П., В.А. Фурсов. Лекции по параллельным вычислениям. Самара. Издательство СГАУ, 2009, 163 с.
10. Боровков А.И., Болдырев Ю.Я., Заборовский В.С. Суперкомпьютерный инжиниринг. Труды Международной суперкомпьютерной конференции Научный сервис в сети Интернет: все грани параллелизма. Направление 1: Параллельные вычисления и их приложения. Г. Новороссийск, 23-28 сентября 2013 г. Изд-во МГУ, 2013. Электронное издание, стр. 436-437.
11. Болдырев Ю.Я., Боровков А.И., Заборовский В.С. Компьютерный инжиниринг - платформа модернизации отечественной промышленности. Труды Международной суперкомпьютерной конференции Научный сервис в сети Интернет: многообразие суперкомпьютерных миров. Направление: Параллельные вычисления и их приложения. Г. Новороссийск, 22-27 сентября 2014 г. Изд-во МГУ, 2014. Электронное издание, стр. 152-153.

# Математика и компьютеринг: опыт покорения космоса и перспективы для «Будущего Земли». К 110-летию со дня рождения М.В.Келдыша и 60-летию полета Ю.А.Гагарина в Год науки и технологий.

Т.А. Сушкевич, С.А. Стрелков, С.В. Максакова

Институт прикладной математики им. М.В. Келдыша РАН

В двадцать первый год двадцать первого века – год славного 60-летнего юбилея легендарного первого полета в космос первого землянина и это был гражданин Советского Союза Социалистических Республик Юрий Алексеевич Гагарин – статья приурочена к 110-летию со дня рождения Главного Теоретика космонавтики и Президента Академии наук СССР незаменного М.В.Келдыша, с которым связаны покорение космоса и «золотой век» отечественной фундаментальной и прикладной науки. «Атомный проект» явился главной движущей силой развития не только физики, математики и других наук, но и детонатором создания больших математических счетных машин, названных позже ЭВМ (электронно-вычислительные машины). Однако мощными драйверами развития вычислительной техники, информационных технологий, искусственного интеллекта и «цифровизации» стали «Космический проект» и проект «Ракетно-ядерный щит». Планета Земля – естественный пример динамической системы с нелинейными процессами, находящейся в непрерывных изменениях. Уже очевидна угроза глобальных изменений на планете. Программа «Будущее Земли», инициатором которой является Международный научный совет, – естественный всеобъемлющий этап развития науки, технологий и общества, направленный на устойчивое развитие и обеспечение жизни на планете. Требуется консолидация всего мирового научного сообщества и мировых ресурсов суперкомпьютеров и информационных баз big data. Повышается роль математиков, «computer sciences» и космоса для реализации Программы, поскольку невозможны натурные эксперименты для исследования эволюции природной среды и климата планеты. По инициативе Т.А.Сушкевич радиационное поле Земли признали как «нематериальную» компоненту климата. В приоритете сопряженные прямые и обратные задачи – компьютерное моделирование радиационных процессов, прогностические расчеты радиационных характеристик и обработка огромных массивов данных глобального мониторинга и дистанционного зондирования Земли из космоса. Огромный теоретический и прикладной научный потенциал, созданный отечественными учеными на заре космической эры, позволяет сохранять ведущие позиции в мире при реализации Программы «Будущее Земли».

*Ключевые слова:* Космический проект, М.В.Келдыш, математика, ЭВМ, математическое моделирование, компьютеринг, Будущее Земли

## 1. Введение

В Год науки и технологий **9 апреля 2021 года** – двадцать первый год двадцатого столетия накануне знаменательного юбилея – 60-летия первого полета в космос жителя планеты Земля и это был гражданин Союза Советских Социалистических Республик (СССР) коммунист Юрий Алексеевич Гагарин – Россия в прямом эфире показала всему миру запуск «Космического корабля Ю.А.Гагарин» по уникальной сверхкороткой орбите и его стыковку с Международной космической станцией (МКС), тем самым всему человечеству напомнила об отечественных пионерских приоритетных достижениях и продемонстрировала свое лидерство в космосе. В соответствии с программой Международной космической станции в 10 часов 42 минуты со стартовой площадки № 31 космодрома Байконур осуществлен пуск ракеты-носителя «Союз-2.1а» с пилотируемым кораблем «Ю.А.Гагарин» (Союз МС-18) и международным экипажем из трех космонавтов (двое из России, один из США) длительной экспедиции МКС-65.

М.В.Келдыш [1-3] был идеологом и организатором космических исследований, а с 1946 года отвечал за прикладную математику, расчеты и ЭВМ. Ключевым организационным решением по консолидации научных организаций и специалистов в области космических исследований послужило Постановление ЦК КПСС и Совета Министров СССР от **10 декабря 1959 г. №1388-618** о создании Межведомственного научно-технического совета по космическим исследованиям (МНТС по КИ) при АН СССР (открытое название – Совет № 1), **Председателем** которого в статусе **Министра СССР** был назначен **академик М.В.Келдыш** (1959-1978), академик-секретарь Отделения математики и директор Института прикладной математики АН СССР.

А **60 лет назад 5-12 апреля 1961 года на космодроме «Байконур»** вершилась история человечества: шла мощнейшая интеллектуальная работа – «мозговой штурм русских гениев» – на последнем этапе реализации мечты человека преодолеть земное притяжение и полететь в космос. Главными были академик М.В.Келдыш, академик С.П.Королев и генерал-лейтенант авиации Н.П.Каманин. Главный Конструктор С.П.Королев был ответственным за техническое обеспечение, Главный Теоретик космонавтики и Председатель МНТС по КИ министр и академик М.В.Келдыш – за программу исследовательской работы космонавта на борту космического корабля (первый взгляд на Землю из космоса – «Красота»!) и баллистические расчеты, которые проводились на ЭВМ «Стрела» в Отделении прикладной математики Математического института им. В.А.Стеклова АН СССР (ОПМ МИАН СССР – секретное), созданного в 1953 году, а Н.П.Каманин – за подготовку космонавтов (с апреля 1958 года заместитель начальника Главного штаба ВВС, с января 1961 года заместитель начальника боевой подготовки ВВС по космосу).

**Из дневника первого руководителя подготовки космонавтов Н.П.Каманин:**

*«6 апреля прилетел председатель Государственной комиссии Константин Николаевич Руднев. Основным событием дня было техническое совещание под руководством С.П.Королева. ... Присутствовали: председатель Госкомиссии, академик Мстислав Всеволодович Келдыш, главные конструкторы двигателей, систем связи, оборудования, управления... Итог совещания: окончательно разработано задание космонавту на одновитковый полет. Подписать этот документ выпала честь С.П.Королеву, М. В. Келдышу и мне. Первое задание летчику-космонавту на первый полет в космос.*

*8 апреля состоялось заседание Государственной комиссии по пуску космического корабля «Восток» с человеком на борту... Рассмотрели и утвердили задание на космический полет... Мне были даны полномочия от имени командования Военно-Воздушных Сил назвать кандидатом Гагарина Юрия Алексеевича, а его дублером – Титова Германа Степановича...*

*10 апреля вечером состоялось торжественное заседание Государственной комиссии по пуску корабля «Восток». В небольшом зале второго этажа монтажного корпуса собралось все руководство: Государственная комиссия, испытатели, главные конструкторы, медики, другие ученые (прим. около 70 человек). Столы поставлены буквой «Т». В середине Константин Николаевич Руднев – председатель Государственной комиссии. Рядом Сергей Павлович Королев, Мстислав Всеволодович Келдыш.*

*И вот 12 апреля. Чуть брезжит рассвет. В 5.30 разбудили космонавтов. В 6.00 состоялось заседание комиссии. Оно было удивительно простым и коротким. Доклады сводились к нескольким фразам: «Замечаний нет, все готово». «Вопросов нет». «Можно производить пуск». В 8.20 на старт прибыли члены Государственной комиссии. Договариваемся о порядке посадки Гагарина в корабль. Автобус с космонавтами должен прибыть на стартовую площадку в 8.50. От автобуса до лифта – 50 метров. Гагарина должны провожать председатель Государственной комиссии, С. П. Королев, другие члены Госкомиссии ...»*

Первую пресс-конференцию первого космонавта Ю.А.Гагарина проводил Президент АН СССР А.Н.Несмеянов **15 апреля** в Доме ученых. Ю.А.Гагарину вручили первую Золотую медаль им. К.Э.Циолковского «За выдающиеся работы в области межпланетных сообщений», учрежденную АН СССР в 1954 году. М.В.Келдыш получил такую медаль в 1972 году. А 15 апреля М.В.Келдыш встречал Ю.А.Гагарина и маршала авиации К.А.Вершинина (см. Рис. 1), но из-за чрезмерных мер секретности в президиуме не появлялся и не выступал. **19 мая 1961 года М.В.Келдыша избирают Президентом АН СССР** и в тот же день он выступил на Общем собрании с сообщением о полете Ю.А.Гагарина. Все последующие пресс-конференции по ключевым событиям освоения космоса М.В.Келдыш проводил сам как Президент АН СССР в Актовом зале МГУ и был известен на весь мир, поскольку никто так профессионально и эмоцио-

нально не мог рассказывать о космических успехах СССР. Во время похорон С.П.Королева на Красной площади **11 января 1966 года**, находясь на Мавзолее, с речью о С.П.Королеве выступил М.В.Келдыш – все поняли, кто же был Главный Теоретик космонавтики. В соответствии с Постановлением Президиума Академии наук СССР от **8 июля 1966 г.** № 465-010 секретное ОПМ МИАН СССР было преобразовано в Институт прикладной математики Академии наук СССР (ИПМ АН СССР) и на здании впервые появилась табличка с названием института. С 1953 года это был первый в мире институт прикладной математики!



**Рис. 1.** М.В.Келдыш встречает Ю.А.Гагарина и маршала авиации К.А.Вершинина перед первой пресс-конференцией в Доме ученых 15 апреля 1961 года. (АРАН. Ф.1546. Оп.1. Д.75. Л.170.)

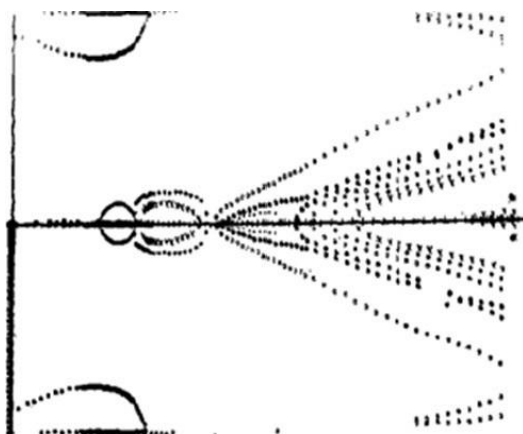
На космодром «Байконур» **5 апреля** прилетели шесть первых кандидатов в космонавты, которые с декабря 1960 года проходили подготовку к первому полету в космос: Юрий Гагарин, Герман Титов, Григорий Нелюбов, Андриян Николаев, Павел Попович, Валерий Быковский. М.В.Келдыш отдавал предпочтение самому молодому с хорошо развитым интеллектом и увлекающемуся поэзией Г.С.Титову (ему было 25 лет), с которым сложились дружественные доверительные отношения. Герман Титов был вторым космонавтом и совершил ответственный и самый опасный суточный полет **6-7 августа 1961 года**, получив большую дозу облучения, но мужественно выполнил задание и впервые в истории человечества сделал съемки Земли с космического корабля. Г.С.Титов снимал репортерской кинокамерой «Конвас» отечественного производства как на обычную пленку, так и на цветную. Космонавт выполнил киносъемку земной поверхности с высоты 244 км и стал первым космическим кинооператором. Это **Г.С.Титов первый показал всем Землю как она видится из космоса!** Материалы съемок были уникальные и имели огромное научное значение. Для разработки и верификации глобальной модели Земли мы использовали результаты съемок Г.С.Титова (см. ниже Рис. 3), с которым лично была знакома Т.А.Сушкевич. Не случайно в новом здании Института Келдыша и Института космических исследований АН СССР (основан по инициативе М.В.Келдыша в 1965 году) на Профсоюзной ул., д. 88 первую секцию предоставили для Управления МО СССР, которое до начала 90-х годов возглавлял Г.С.Титов. В этой секции кабинет Г.С.Титова сохранили как музей.

Уровень мобилизации профессиональных кадров, прежде всего ученых, инженеров, конструкторов, беспрецедентный уровень организации работы многочисленных коллективов и нескольких ведомств, мощнейший интеллект образованного советского народа и гениальность творцов до сих пор поражают специалистов. Это был трудовой подвиг талантливого народа, который только что победил фашизм и прославил свою Родину навека, сделав первый шаг в бесконечность! СССР безоговорочно выиграл космическую гонку у своего главного конкурента – США и более десяти лет оставался Лидером, осуществив полеты АМС на Луну, Венеру, Марс, а в приземном космосе космонавты впервые совершали выход в открытое космическое пространство, стыковку космических кораблей, длительные экспедиции на Долгосрочных орбитальных станциях. Вершиной сотрудничества СССР и США стал проект «Союз-Аполлон».

Создание в 1953 году первого в мире Института прикладной математики как ОПМ МИАН – это был компромисс ради сохранения единого коллектива математиков. Международное признание научных школ по разным направлениям математики было продемонстрировано в 1966 году: **55 лет назад с 16 по 26 августа** в МГУ имени М.В.Ломоносова проходил Международный конгресс математиков. О его широком размахе свидетельствует число участников – более 5 тысяч, из них 2400 математиков из СССР, 463 докладчика из США! Открывал конгресс президент АН СССР академик М.В.Келдыш, президентом конгресса был его друг – ректор МГУ академик И.Г.Петровский. Это было начало расцвета «Эпохи Келдыша»!

## 2. Опыт решения прикладных работ

Подтверждается стратегический выбор ответа на вопрос «**Зачем нужен космос?**», сделанный в **1955 г.** М.В.Келдышем – Главным Теоретиком Космонавтики и государственным деятелем: «**разведка и наблюдение Земли**», актуальный и в XX-м и в XXI-м веке. В **1955 году** было основано Министерство общего машиностроения (Роскосмос его преемник), начали строить космодром «Байконур», а в Институте Келдыша профессор Е.С.Кузнецов (наш советский Чандрасекар) основал уникальный и единственный в мире отдел «Кинетические уравнения», который составлял основу московской научной школы по уравнениям переноса излучения, нейтронов, заряженных частиц в разных средах. В этот отдел **60 лет назад в 1961 году** с кафедры «математика» А.Н.Тихонова на физическом факультете МГУ им. М.В.Ломоносова была направлена физик-теоретик **Т.А.Сушкевич** – последняя ученица и преемница научного наследия **Е.С.Кузнецова** (13.03.1901-17.02.1966), которому в 2021 г. было бы **120 лет со дня рождения**. С основания ОПМ МИАН М.В.Келдыш сделал его полигоном для развития вычислительной и прикладной математики и освоения новых ЭВМ для решения прикладных задач. Первой ЭВМ второго поколения была «Весна» [4], для математической сдачи которой Т.А.Сушкевич была написана первая большая программа (в кодах команд) для моделирования прохождения спутников и ракет через только что открытые с помощью спутников ионосферу и радиационные пояса Земли [5]. Фундаментальная научная проблема имела важные приложения: нужно было оценить влияние на космический аппарат и влияние на космическую связь. Для расчета одного варианта требовалось 28 часов процессорного времени. На ЭВМ «Весна» впервые был реализован мультипрограммный режим, а также построен первый в СССР компьютерный график и создан анимационный фильм, которые показали впервые за рубежом. Расчеты и формирование матриц с рисунками обеспечивала Т.А.Сушкевич. Ю.М.Баяковский создал технические средства для характрона, фотосъемки с его экрана, размножения и монтажа изображений.



**Рис. 2.** Первый в СССР компьютерный график, ЭВМ «Весна», август 1964 г.



**Рис. 3.** Один из первых снимков Земли. Г.С.Титов на «Восток-2», 6 августа 1961 г.

В 1966 году в ИПМ установили первый экземпляр самой успешной и лучшей в Европе советской БЭСМ-6. Начался длительный период освоения космического пространства с помощью БЭСМ-6, АС-6, ЭВМ ЕС-ряда и других, который длился до 1992 года. **55 лет назад в 1966 году** была разработана первая в мире и никем не превзойденная до сих пор глобальная сферическая модель радиационного поля Земли, которая использовалась для теоретико-расчетных исследований, пионерских научных космических экспериментов и выполнения стратегических государственных заданий до 1992 года [6-16]. Алгоритмы были настолько сложные и с огромным числом условных переходов, что не хватало ресурсов ни одного транслятора с автоматических языков программирования. Программа была написана на Автокоде (разработка В.С.Штаркмана и его сотрудников), занимала около 25 тысяч перфокарт. Для расчета одного варианта требовалось порядка 300 часов процессорного времени, 14 МЛ и 6 МД. Впервые были использованы приемы распараллеливания вычислений с помощью эффективной организации оперативной памяти и обменов с внешними носителями, что позволяло проводить расчеты без потерь.

### 3. Математические модели и методы

Непреодолимая сложность проблемы состоит в том, что для исследований планеты не допустимы натуральные эксперименты и возможны только мониторинг и наблюдения разными средствами, с одной стороны, а с другой стороны на момент измерений радиации невозможно восстановить весь набор оптико-геофизических и оптико-метеорологических параметров системы «атмосфера-суша-океан», от которых зависит радиация, и не возможно повторить условия наблюдений, так как среда непрерывно изменяется и никогда не повторяется. И только математическое моделирование позволяет провести теоретико-расчетные исследования столь сложных проблем и получить качественные и количественные оценки для анализа и прогнозов.

Планета Земля - естественный пример динамической системы с нелинейными процессами, находящейся в непрерывных изменениях. Древние астрономы использовали свет для наблюдений за другими планетами и звездами. И не случайно наблюдения и исследования планеты Земля проводятся с помощью "световых технологий", поскольку скорость света такова, что исследуемый объект можно считать "стационарным" и в теории переноса излучения практически решаются стационарные кинетические уравнения без временной зависимости.

#### 3.1. Приближение плоского слоя

Рассматривается задача переноса излучения в системе «многослойная атмосфера-подстилающая поверхность на уровне  $z = h$ » (САП) в приближении плоского слоя, в которой атмосфера принимается гетерогенной, т.е. вертикально-неоднородной и, возможно, с разными радиационными режимами в подслоях, но горизонтально-однородной, а подстилающая поверхность может быть любого типа: горизонтально-однородной или горизонтально-неоднородной с ламбертовым или анизотропным отражением. САП считается немультимплицирующей (без размножения).

Формулируется общий подход для задач с 1D-, 2D-, 3D-размерности по пространству, которая зависит от пространственных характеристик отражающей подстилающей поверхности: если подстилающая поверхность горизонтально-однородная, то и перенос излучения в такой САП одномерный; если подстилающая поверхность горизонтально-неоднородная по обеим координатам  $x, y$  (например, мозаичная, как в натуральных условиях), то приходится иметь дело с проблемами трехмерного переноса излучения в САП; двумерная задача получается при наличии осевой симметрии или неоднородности по одной координате.

Такую САП для гетерогенного рассеивающего, поглощающего и излучающего слоя, не ограниченного в горизонтальном направлении ( $-\infty < x, y < \infty, r_{\perp} = (x, y)$ ) и конечного по высоте ( $0 \leq z \leq h$ ), описываем в общем случае для трехмерного евклидова пространства: радиус-вектор  $r = (x, y, z)$ . Множество всех направлений распространения излучения  $s = (\mu, \varphi)$ , где  $\mu = \cos \vartheta$ ,  $\vartheta \in [0, \pi]$  — зенитный угол, отсчитываемый от направления внутренней нормали к верхней границе слоя  $z = 0$ , которая совпадает с осью  $z$ , и  $\varphi \in [0, 2\pi]$  — азимут, от-



считываемый от положительного направления оси  $x$ , образует единичную сферу  $\Omega = \Omega^+ \cup \Omega^-$ ;  $\Omega^+$  и  $\Omega^-$  — полушеры для направлений распространения нисходящего, пропущенного излучения с  $\mu \in [0,1]$  и восходящего, отраженного излучения с  $\mu \in [-1,0]$  соответственно. Для удобства записи краевых условий на внешних границах САП вводим множества:

$$t = \{z, r_{\perp}, s : z = 0, s \in \Omega^+\}, \quad b = \{z, r_{\perp}, s : z = h, s \in \Omega^-\}.$$

Интенсивность (энергетическая яркость) монохроматического излучения  $\Phi_{\lambda}(r, s)$  в САП (длина волны  $\lambda$  фиксирована и ниже опущена) находится как решение общей краевой задачи (ОКЗ при  $R_b \neq 0$  и  $\varepsilon \neq 0$ ) для кинетического уравнения в форме линеаризованного уравнения Больцмана в приближении бинарных «столкновений» фотонов с компонентами среды, когда электромагнитная волна, в силу принципа дуализма «волна-частица», рассматривается как частица-фотон:

$$K\Phi = F^{in}, \quad \Phi|_t = F_t, \quad \Phi|_b = \varepsilon R_b \Phi + F_b; \quad K \equiv D - S. \quad (1)$$

ОКЗ (1) определена для линейных операторов: оператор переноса

$$D \equiv (s, \text{grad}) + \sigma(z) = D_z + \left( s_{\perp}, \frac{\partial}{\partial r_{\perp}} \right); \quad D_z \equiv \mu \frac{\partial}{\partial z} + \sigma(z)$$

описывает перенос в одномерной 1D-задаче и  $K_z \equiv D_z - S$ ; в случае трехмерной 3D-задачи

$$D\Phi \equiv \mu \frac{\partial}{\partial z} \Phi + \sin \vartheta \cos \varphi \frac{\partial}{\partial x} \Phi + \sin \vartheta \sin \varphi \frac{\partial}{\partial y} \Phi + \sigma(z)\Phi;$$

интеграл столкновений (для линеаризованного уравнения Больцмана, учитывающего только бинарные взаимодействия без изменения длины волны)

$$S\Phi \equiv \sigma_s(z) \int_{\Omega} \Phi(z, r_{\perp}, s') \gamma(z, s, s') ds', \quad ds' = d\mu' d\varphi', \quad S(1) \leq 1;$$

оператор отражения, описывающий взаимодействие излучения с подстилающей поверхностью без изменения длины волны,

$$[R_b \Phi](h, r_{\perp}, s) \equiv \int_{\Omega^+} \Phi(h, r_{\perp}, s^+) q(r_{\perp}, s, s^+) ds^+, \quad R_b(1) = q^*(r_{\perp}, s) \leq 1, \quad (2)$$

$\gamma(z, s, s')$  — индикатриса рассеяния, зависящая в том числе от высоты;  $\sigma(z)$  и  $\sigma_s(z)$  — вертикальные профили коэффициентов ослабления (экстинкции) и рассеяния;  $q(r_{\perp}, s, s^+)$  — ядро оператора отражения; параметр  $0 \leq \varepsilon \leq 1$  фиксирует акт взаимодействия излучения с подложкой;  $F^{in}(z, s)$ ,  $F_t(r_{\perp}, s^+)$ ,  $F_b(r_{\perp}, s^-)$  — источники инсоляции (внешний солнечный поток, собственное излучение среды).

Для гетерогенной атмосферы на тех высотах, где существенно меняются оптические свойства атмосферы, вводятся внутренние границы  $z = h_m$ ,  $m = 2, 3, \dots, M$ , на которых ставятся дополнительные краевые условия:

$$\Phi|_{d \uparrow, m} = \varepsilon (R_m^{\uparrow} \Phi + T_m^{\uparrow} \Phi) + F_{m-1}^{\uparrow}, \quad \Phi|_{d \downarrow, m} = \varepsilon (R_m^{\downarrow} \Phi + T_m^{\downarrow} \Phi) + F_m^{\downarrow},$$

$$d \downarrow, m = \{z, r_{\perp}, s : z = 0, s \in \Omega^+\}, \quad d \uparrow, m = \{z, r_{\perp}, s : z = h, s \in \Omega^-\},$$

описывающие обмен излучением между подслоями с помощью операторов отражения  $R_m^{\downarrow}$ ,  $R_m^{\uparrow}$  и пропускания  $T_m^{\downarrow}$ ,  $T_m^{\uparrow}$ .

Источники на границах  $h_m$ ,  $m = 2 \div M$ , учитывают обмен излучением между подслоями:

$$F_m^{\downarrow} = T_m^{\downarrow} \Phi_{m-1}^{\downarrow} + T_m^{\downarrow} \Phi_{m-1}^{\uparrow} + R_m^{\downarrow} \Phi_m^{\downarrow} + R_m^{\downarrow} \Phi_m^{\uparrow};$$

$$F_m^{\uparrow} = R_{m+1}^{\uparrow} \Phi_m^{\downarrow} + R_{m+1}^{\uparrow} \Phi_m^{\uparrow} + T_{m+1}^{\uparrow} \Phi_{m+1}^{\downarrow} + T_{m+1}^{\uparrow} \Phi_{m+1}^{\uparrow};$$

$$F_1^{\downarrow} = F_t^{\downarrow}; \quad F_{M+1}^{\uparrow} = F_b^{\uparrow}; \quad d \downarrow, 1 = t \downarrow; \quad d \uparrow, M+1 = b \uparrow.$$

Решение краевой задачи (1) можно искать в виде суперпозиции  $\Phi = \Phi_a + \Phi_q$ . Фоновое излучение атмосферы  $\Phi_a$  определяется как решение первой краевой задачи теории переноса (ПКЗ) с «вакуумными» граничными условиями ( $R = 0$ )

$$K\Phi_a = F^{in}, \quad \Phi_a|_t = F_t, \quad \Phi_a|_b = F_b. \quad (3)$$

Задача для нахождения подсветки  $\Phi_q$  от подстилающей поверхности, — это ОКЗ

$$K\Phi_q = 0, \quad \Phi_q|_t = 0, \quad \Phi_q|_b = \varepsilon R_b \Phi_q + E_b; \quad E_b(r_\perp, s) \equiv R_b \Phi_a \quad (4)$$

— функция источника излучения — это яркость (освещенность, облученность) подложки, создаваемая фоновым излучением.

Вводим алгебраические векторы с размерностью  $2M$  :

полное решение  $\Phi = \{\Phi_1^\downarrow, \Phi_1^\uparrow, \Phi_2^\downarrow, \Phi_2^\uparrow, \dots, \Phi_m^\downarrow, \Phi_m^\uparrow, \dots, \Phi_M^\downarrow, \Phi_M^\uparrow\}$ ;

функции влияния слоев  $\Theta = \{\Theta_1^\downarrow, \Theta_1^\uparrow, \Theta_2^\downarrow, \Theta_2^\uparrow, \dots, \Theta_m^\downarrow, \Theta_m^\uparrow, \dots, \Theta_M^\downarrow, \Theta_M^\uparrow\}$ ;

начальное приближение источников  $\mathbf{E} = \{E_1^\downarrow, E_1^\uparrow, E_2^\downarrow, E_2^\uparrow, \dots, E_m^\downarrow, E_m^\uparrow, \dots, E_M^\downarrow, E_M^\uparrow\}$ ;

«сценарии» на границах  $\mathbf{Z} = \{Z_1^\downarrow, Z_1^\uparrow, Z_2^\downarrow, Z_2^\uparrow, \dots, Z_m^\downarrow, Z_m^\uparrow, \dots, Z_M^\downarrow, Z_M^\uparrow\}$ .

Функции влияния отдельных подслоев находятся как решения первых краевых задач для подслоев с номерами  $m = 1, 2, \dots, M$  :

$$K\Theta_m^\downarrow = 0, \quad \Theta_m^\downarrow|_{d\downarrow, m} = f_{\delta, m}^\downarrow, \quad \Theta_m^\downarrow|_{d\uparrow, m+1} = 0; \quad f_{\delta, m}^\downarrow = \delta(s - s_m^\downarrow);$$

$$K\Theta_m^\uparrow = 0, \quad \Theta_m^\uparrow|_{d\downarrow, m} = 0, \quad \Theta_m^\uparrow|_{d\uparrow, m+1} = f_{\delta, m}^\uparrow; \quad f_{\delta, m}^\uparrow = \delta(s - s_m^\uparrow).$$

Вводится матричный оператор, описывающий взаимодействие между подслоями через функции влияния ( $P$  - матрица, содержащая операторы  $R_m^\downarrow, R_m^\uparrow$  и  $T_m^\downarrow, T_m^\uparrow$ ):

$$\mathbf{GF} = P(\Theta, \mathbf{F}) = \begin{pmatrix} 0 \\ R_2^\uparrow(\Theta_1^\downarrow, F_1^\downarrow) + R_2^\uparrow(\Theta_1^\uparrow, F_1^\uparrow) + T_2^\uparrow(\Theta_2^\downarrow, F_2^\downarrow) \\ \dots \\ T_m^\downarrow(\Theta_{m-1}^\downarrow, F_{m-1}^\downarrow) + T_m^\downarrow(\Theta_{m-1}^\uparrow, F_{m-1}^\uparrow) + R_m^\downarrow(\Theta_m^\downarrow, F_m^\downarrow) + R_m^\downarrow(\Theta_m^\uparrow, F_m^\uparrow) \\ R_{m+1}^\uparrow(\Theta_m^\downarrow, F_m^\downarrow) + R_{m+1}^\uparrow(\Theta_m^\uparrow, F_m^\uparrow) + T_{m+1}^\uparrow(\Theta_{m+1}^\downarrow, F_{m+1}^\downarrow) + T_{m+1}^\uparrow(\Theta_{m+1}^\uparrow, F_{m+1}^\uparrow) \\ \dots \\ T_M^\downarrow(\Theta_{M-1}^\downarrow, F_{M-1}^\downarrow) + T_M^\downarrow(\Theta_{M-1}^\uparrow, F_{M-1}^\uparrow) + R_M^\downarrow(\Theta_M^\downarrow, F_M^\downarrow) + R_M^\downarrow(\Theta_M^\uparrow, F_M^\uparrow) \\ R_b^\uparrow(\Theta_M^\downarrow, F_M^\downarrow) + R_b^\uparrow(\Theta_M^\uparrow, F_M^\uparrow) \end{pmatrix}.$$

Асимптотически точное решение задачи (4) получается в форме линейного векторного функционала, ядром которого является вектор функций влияния  $\Theta$  :

$$\Phi_q = (\Theta, \mathbf{Z}); \quad \mathbf{Z} \equiv \sum_{n=0}^{\infty} G^n \mathbf{E}.$$

Этот функционал есть оптический передаточный оператор гетерогенной системы переноса излучения, в котором все эффекты многократного рассеяния и поглощения в атмосфере с учетом взаимодействия между подслоями, а также отражения от подстилающей поверхности содержатся в векторе «сценариев» на всех границах  $\mathbf{Z}$ . Оптическое изображение «сценария» или яркость подстилающей поверхности удовлетворяет уравнению Фредгольма второго рода

$$Y = R_b(\Theta, Y) + E_b,$$

которое называют «приземной фотографией». Полное излучение САП и «космическая фотография» (изображение, получаемое при наблюдении из космоса) есть «суперпозиция»

$$\Phi = \Phi_a + (\Theta, Y).$$

### 3.2. Сферическая модель

Нас интересует проблема расчета радиационного поля Земли в масштабах всей планеты (одновременно для всех пространственных точек по всей единичной сфере при всех условиях освещения, горизонт, сумерки, область сумерек и тени, полярные регионы и т.д.).

Рассматривается общая краевая задача (ОКЗ) для кинетического уравнения переноса излучения в сферической системе атмосфера-Земля, освещаемой внешним параллельным солнечным потоком. Полная интенсивность монохроматического (при фиксированной длине волны  $\lambda$ ) или квазимонохроматического (при фиксированной  $\lambda$  и интервале разрешения  $\Delta\lambda$ ) стационарного излучения  $\Phi_\lambda(\mathbf{r}, \mathbf{s})$  (индекс  $\lambda$  ниже опускаем) в любой точке  $A(\mathbf{r})$  с радиус-вектором  $\mathbf{r} = (r, \psi, \eta)$  в любом направлении  $\mathbf{s} = (\vartheta, \varphi)$  находится как решение общей краевой задачи переноса излучения (ОКЗ)

$$K\Phi(\mathbf{r}, \mathbf{s}) = F^{in}, \quad \Phi|_t = F^t, \quad \Phi|_b = \varepsilon R\Phi + F^b \quad (5)$$

в фазовой области аргументов  $(\mathbf{r}, \mathbf{s})$  с линейным интегро-дифференциальным оператором  $K \equiv D - S$ , где оператор переноса

$$D \equiv (\mathbf{s}, \text{grad}) + \sigma_{tot}(\mathbf{r}), \quad (6)$$

для задачи со сферической геометрией 3D-размерности

$$(\mathbf{s}, \nabla\Phi) = \cos\vartheta \frac{\partial\Phi}{\partial r} + \frac{\sin\vartheta \cos\varphi}{r} \frac{\partial\Phi}{\partial\psi} - \frac{\sin\vartheta}{r} \frac{\partial\Phi}{\partial\vartheta} + \frac{\sin\vartheta \sin\varphi}{r \sin\psi} \frac{\partial\Phi}{\partial\eta} - \frac{\sin\vartheta \sin\varphi \text{ctg}\psi}{r} \frac{\partial\Phi}{\partial\varphi}; \quad (7)$$

интеграл столкновений или функция источника есть интеграл по единичной сфере направлений  $\Omega := \{\mathbf{s} = (\vartheta, \varphi)\}$

$$B(\mathbf{r}, \mathbf{s}) \equiv S\Phi = \sigma_{sc}(\mathbf{r}) \int_{\Omega} \gamma(\mathbf{r}, \mathbf{s}, \mathbf{s}') \Phi(\mathbf{r}, \mathbf{s}') ds', \quad ds' = \sin\vartheta' d\vartheta' d\varphi'; \quad (8)$$

оператор отражения на подстилающей поверхности в общем случае есть интеграл

$$[R\Phi](\mathbf{r}_b, \mathbf{s}) = \int_{\Omega^-} q(\mathbf{r}_b, \mathbf{s}, \mathbf{s}^-) \Phi(\mathbf{r}_b, \mathbf{s}^-) ds^-, \quad \mathbf{s} \in \Omega^+. \quad (9)$$

Функция  $F^{in}(\mathbf{r}, \mathbf{s})$  представляет плотность источников излучения внутри сферической оболочки;  $F^b(\mathbf{r}_b, \mathbf{s}^+)$  и  $F^t(\mathbf{r}_t, \mathbf{s}^-)$  есть источники излучения на границах, определенные для лучей  $\mathbf{s}$ , направленных внутрь сферической оболочки.

Оператор  $R$  описывает закон отражения излучения от подстилающей поверхности, которая располагается на нижней границе сферической оболочки с радиус-вектором  $\mathbf{r} = \mathbf{r}_b$ ; параметр  $0 \leq \varepsilon \leq 1$  фиксирует акт взаимодействия излучения с подстилающей поверхностью. Если  $R \equiv 0$  (или  $\varepsilon = 0$ ), то имеем дело с первой краевой задачей переноса излучения (ПКЗ)

$$K\Phi_a = F^{in}, \quad \Phi_a|_t = F^t, \quad \Phi_a|_b = F^b \quad (10)$$

для сферической оболочки с неотражающими абсолютно «черными» границами или с прозрачными, «вакуумными» граничными условиями.

Общая краевая задача (5) с операторами (5)-(9) линейная относительно источников и её решение можно представить в виде суперпозиции:  $\Phi = \Phi_a + \Phi_q$ . Фоновое излучение атмосферы  $\Phi_a$  определяется как решение ПКЗ (10). Вклад излучения  $\Phi_q$ , обусловленного отражением от подстилающей поверхности, находится как решение ОКЗ

$$K\Phi_q = 0, \quad \Phi_q|_t = 0, \quad \Phi_q|_b = \varepsilon R\Phi_q + \varepsilon E, \quad (11)$$

в которой яркость подстилающей поверхности, созданная отраженным фоновым излучением  $E = R\Phi_a$ , служит источником инсоляции.

Решение первой краевой задачи

$$K\Phi = 0, \quad \Phi|_t = 0, \quad \Phi|_b = f(\mathbf{s}^h; \mathbf{r}_\perp, \mathbf{s}); \quad \mathbf{r}_\perp = (\psi, \eta) \in \Omega, \quad d\mathbf{r}_\perp = \sin\psi d\psi d\eta, \quad (12)$$

можно записать в форме линейного функционала – «интеграла суперпозиции»

$$\Phi(\mathbf{s}^h; \mathbf{r}, \mathbf{r}_\perp, \mathbf{s}) \equiv (\Theta, f) \equiv \frac{1}{2\pi} \int_{\Omega^+} d\mathbf{s}_h^+ \frac{1}{4\pi} \int_{\Omega} \Theta(\mathbf{s}_h^+; \mathbf{r}, \mathbf{r}_\perp - \mathbf{r}'_\perp, \mathbf{s}) \times f(\mathbf{s}^h; \mathbf{r}'_\perp, \mathbf{s}_h^+) \sin \psi' d\psi' d\eta' .$$

Его ядром является функция влияния  $\Theta(\mathbf{s}_h^+; \mathbf{r}, \mathbf{r}_\perp, \mathbf{s})$  - решение ПКЗ (Модель 1)

$$K\Theta = 0, \quad \Theta|_l = 0, \quad \Theta|_b = f_\delta$$

с параметром  $\mathbf{s}_h^+ \in \Omega^+$  и источником  $f_\delta(\mathbf{s}_h^+; \mathbf{r}_\perp, \mathbf{s}) = \delta(\mathbf{r}_\perp) \delta(\mathbf{s} - \mathbf{s}_h^+)$ .

Если источник  $f(\mathbf{r}_\perp)$  - изотропный (ламбертовский) и горизонтально неоднородный, то решение ПКЗ (8) есть линейный функционал – «интеграл свертки»

$$\Phi(\mathbf{r}, \mathbf{r}_\perp, \mathbf{s}) = F_c(f) \equiv (\Theta_c, f) \equiv \frac{1}{4\pi} \int_{\Omega} \Theta_c(\mathbf{r}, \mathbf{r}_\perp - \mathbf{r}'_\perp, \mathbf{s}) f(\mathbf{r}'_\perp) \sin \psi' d\psi' d\eta'$$

с ядром – функцией влияния

$$\Theta_c(\mathbf{r}, \mathbf{r}_\perp, \mathbf{s}) = \frac{1}{2\pi} \int_{\Omega^+} \Theta(\mathbf{s}_h^+; \mathbf{r}, \mathbf{r}_\perp, \mathbf{s}) d\mathbf{s}_h^+,$$

которая удовлетворяет ПКЗ (8) с осевой симметрией (Модель 2)

$$K\Theta_c = 0, \quad \Theta_c|_l = 0, \quad \Theta_c|_b = \delta(\mathbf{r}_\perp).$$

Для анизотропного и горизонтально однородного источника

$$\Phi(\mathbf{s}^h; r, \mathbf{s}) = F_r(f) \equiv (\Theta_r, f) \equiv \frac{1}{2\pi} \int_{\Omega^+} \Theta_r(\mathbf{s}_h^+; r, \mathbf{s}) f(\mathbf{s}^h; \mathbf{s}_h^+) d\mathbf{s}_h^+$$

с ядром линейного функционала

$$\Theta_r(\mathbf{s}_h^+; r, \mathbf{s}) = \frac{1}{4\pi} \int_{\Omega} \Theta(\mathbf{s}_h^+; r, r_\perp, \mathbf{s}) \sin \psi d\psi d\eta.$$

Функция влияния  $\Theta_r$  есть решение одномерной сферической ПКЗ с азимутальной зависимостью (Модель 3)

$$K_r\Theta_r = 0, \quad \Theta_r|_l = 0, \quad \Theta_r|_b = \delta(\mathbf{s} - \mathbf{s}_h^+).$$

При изотропном (ламбертовом) и горизонтально однородном источнике решение ПКЗ (12)

$$\Phi(r, \mathbf{s}) = fW(r, \mathbf{s}), \quad f = \text{const},$$

рассчитывается через функцию влияния

$$\begin{aligned} W(r, \mathbf{s}) &= \frac{1}{2\pi} \int_{\Omega^+} d\mathbf{s}_h^+ \frac{1}{4\pi} \int_{\Omega} \Theta(\mathbf{s}_h^+; \mathbf{r}, \mathbf{r}_\perp, \mathbf{s}) \sin \psi d\psi d\eta = \\ &= \frac{1}{4\pi} \int_{\Omega} \Theta_c(\mathbf{r}, \mathbf{r}_\perp, \mathbf{s}) \sin \psi d\psi d\eta = \frac{1}{2\pi} \int_{\Omega^+} \Theta_r(\mathbf{s}_h^+; r, \mathbf{s}) d\mathbf{s}_h^+, \end{aligned}$$

которая также называется функцией пропускания, отягощенной многократным рассеянием, и определяется как решение одномерной сферической ПКЗ со сферической симметрией (Модель 4)

$$K_r W = 0, \quad W|_l = 0, \quad W|_b = 1.$$

На основе теории регулярных возмущений с помощью ряда

$$\Phi_q(\mathbf{s}^h; \mathbf{r}, \mathbf{s}) = \sum_{k=1}^{\infty} \mathcal{E}^k \Phi_k$$

ОКЗ (11) сводится к рекурсивной системе ПКЗ (12)

$$K\Phi_k = 0, \quad \Phi_k|_l = 0, \quad \Phi_k|_b = E_k \tag{13}$$

с источниками  $E_k = R\Phi_{k-1}$  для  $k \geq 2$ ,  $E_1 = E$ .

Вводится оператор, описывающий единичный акт взаимодействия падающего излучения с подстилающей поверхностью через функцию влияния

$$[Gf](s^h; \mathbf{r}_b, \mathbf{s}) \equiv R(\Theta, f) = \int_{\Omega^-} q(\mathbf{r}_b, \mathbf{s}, \mathbf{s}^-)(\Theta, f) ds^-.$$

Решения системы ПКЗ (13) находятся как линейные функционалы

$$\Phi_1 = (\Theta, E), \quad \Phi_k = (\Theta, R\Phi_{k-1}) = (\Theta, G^{k-1}E).$$

Асимптотически точное решение ОКЗ (11) получается в форме линейного функционала – оптического передаточного оператора

$$\Phi_q = (\Theta, Y),$$

где оптическое изображение «сценария» или яркость подстилающей поверхности

$$Y \equiv \sum_{k=0}^{\infty} G^k E = \sum_{k=0}^{\infty} R\Phi_k \quad (14)$$

дается рядом Неймана по кратности отражения излучения от подстилающей поверхности с учетом многократного рассеяния внутри оболочки атмосферы. «Сценарий» (14) удовлетворяет уравнению Фредгольма второго рода  $Y = R(\Theta, Y) + E$ , которое называют «приземной фотографией». Полное излучение САЗ и «космическая фотография» (изображение, получаемое при наблюдении из космоса) есть «суперпозиция»

$$\Phi = \Phi_a + (\Theta, Y) \quad (15)$$

#### 4. О супервычислениях и параллельных алгоритмах

Цель разработки обеспечить максимально возможную переносимость «унаследованного» комплекса программ, который развивается по мере появления новых суперкомпьютеров со своими архитектурами, и обеспечить прозрачную работу в распределенной сетевой среде. В космической области для обеспечения преемственности принято писать вычислительные блоки на Fortran. Управление расчетами осуществляется на языке «сценариев» (с 1996 года использовали динамический язык программирования высокого уровня общего назначения Perl для системного администрирования).

Используются следующие *приемы распараллеливания вычислений*:

- 1) распределенные вычисления по физическим моделям:
  - многоспектральные, в том числе гиперспектральные (по длине волны);
  - по оптико-геофизической погоде (по коэффициентам общей краевой задачи);
  - по источникам излучения;
- 2) распределенные вычисления на основе методического распараллеливания - декомпозиции краевых задач:
  - по моделям переноса излучения, т.е. по приближениям теории переноса излучения;
  - по подобластям;
  - по параметрам функций влияния;
  - по компонентам векторов функций влияния;
  - по параметрам пространственно-частотных характеристик;
  - по компонентам векторов пространственно-частотных характеристик;
  - по компонентам векторных функционалов;
- 3) алгоритмическое распараллеливание для многомерных моделей:
  - однократное рассеяние по характеристикам;
  - многократное рассеяние по интегралам столкновений;
  - по квадрантам угловых разностных сеток;
  - по подобластям с разными сеточно-характеристическими схемами.

*Основные составные части математического обеспечения:*

- банки данных по оптико-метеорологическим моделям атмосферы и земной поверхности;
- система автоматизированного расчета спектро-энергетических и других радиационных характеристик атмосферы и Земли в различных диапазонах спектра от УФ до ММВ;
- банки данных радиационных характеристик (функции влияния локальных возмущений параметров или источников в атмосфере, дымах, облаках, гидрометеорах, океане и на земной

поверхности, пространственно-угловые и спектральные распределения яркости системы Земля-атмосфера, функции пропускания и сферическое альbedo атмосферы и т.д.);

- пакеты программ обработки, визуализации и диагностики результатов численного эксперимента и аэрокосмических данных.

Библиотека программ численного решения краевых задач теории переноса излучения в рассеивающих, поглощающих и излучающих средах (атмосфера, океан, облачность, дымы, гидрометеоры, водные бассейны) составляется из набора программ на Fortran, каждая из которых позволяет рассчитывать радиационные характеристики при заданных модели и методике (краевая задача теории переноса, геометрия, численный метод и т.д.) в определенном диапазоне длин волн.

С учетом источников и процессов трансформации излучения выделяются *четыре основные физико-математические модели*, отвечающие спектральным диапазонам:

- оптический диапазон (источник - Солнце, многократное рассеяние и поглощение);

- ближний ИК-диапазон (источники - Солнце и собственное излучение, многократное рассеяние и поглощение);

- ИК-диапазон (источники - Солнце и собственное излучение, без многократного рассеяния, сложная структура спектров поглощения);

- ММВ диапазон (источник - собственное радиоизлучение, многократное рассеяние в гидрометеорах и облаках, сложные спектры поглощения).

Программные комплексы создаваемой системы автоматизированного расчета, обработки и анализа радиационных характеристик Земли и решения задач дистанционного зондирования разрабатываются на многопроцессорных суперЭВМ с параллельными вычислениями под управлением через сеть с «рабочего места», организованного на РС.

Создаваемая система содержит *три группы программных комплексов*.

Первая группа программ - формирование оптико-метеорологических моделей среды: программы работы с архивом и базами данных моделей атмосферы, облаков, дымов, земной поверхности, океана; банк спектров поглощения атмосферных газов; банк характеристик аэрозольного рассеяния и поглощения; формирование модели атмосферы; пакеты данных к программам расчета радиационных характеристик и т.д.

Вторая группа программ - численное решение скалярного уравнения переноса излучения быстрыми приближенными и репрезентативными высокоточными методами для плоской геометрии: для системы свободная атмосфера-дымовая завеса, для системы атмосфера-океан, для системы атмосфера с многоярусными облаками, для функции влияния атмосферы, дымов, облачности, гидрометеоров, океана, для функции пропускания атмосферы, отягощенной многократным рассеянием, и т.д.

Третья группа программ - обработка и диагностика результатов расчетов: аналитическая аппроксимация и параметризация табличных функций; компьютерная графика и визуализация; решение обратных задач по восстановлению параметров среды и т.д.

## 5. Заключение

Население планеты настолько быстро привыкло к удобствам, которые обеспечивает космос, что полеты космонавтов и космические следования уже не привлекают их внимание. Только одно беспокоит – чтобы не было «космической войны». И согласны, что за Землей нужно из космоса присматривать.

В 2021 году отмечаем 110-летие со дня рождения М.В.Келдыша, 115-летие – А.Н.Тихонова, 120-летие – Е.С.Кузнецова. Безмерно благодарна своим Учителям, которые всем своим образом жизни и научной деятельности показывали Образец служения Родине и науке и оказывали неявное, но существенное влияние на формирование мировоззрения

Это были Гении. Гении не нуждаются в почитании, память о них нужна нам, живущим, и тем, кто придет нам на смену. Когда рвется ниточка памяти, протянутая из прошлого в будущее, нация деградирует и погибает. Помним ли мы об этом?!

## Литература

1. Келдыш М.В. Творческий портрет по воспоминаниям современников. М.: Наука, 2001. 416 с.
2. Сушкевич Т.А. Главный Теоретик М.В.Келдыш и Главный Конструктор космонавтики С.П.Королев – покорители космоса // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т. 8. № 1. С. 9-25.
3. Сушкевич Т.А. М.В.Келдыш – организатор международного сотрудничества в космосе и первой советско-американской Программы «Союз-Аполлон» (ЭПАС) // Современные проблемы дистанционного зондирования Земли из космоса. 2011. Т. 8. № 4. С. 9-22.
4. Сушкевич Т.А. К 50-летию первой отечественной полупроводниковой ЭВМ «Весна» и отечественной компьютерной графики: Труды XVI Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров», г. Новороссийск, 22-27 сентября 2014 года. Российская Академия Наук, Суперкомпьютерный консорциум университетов России. М.: Изд-во МГУ им. М.В.Ломоносова, 2014.
5. Масленников М.В., Сигов Ю.С., Сушкевич Т.А. Численное решение задачи о стационарном обтекании тела разреженной плазмой: Тезисы докладов. Четвертое совещание по магнитной гидродинамике, Рига, 22-27 июня 1964 г. Рига: Изд. АН Латв. ССР, 1964.
6. Сушкевич Т.А. О пионерских работах по математическому моделированию радиационного поля Земли при освоении космоса // Современные проблемы дистанционного зондирования Земли из космоса. 2008. В. 5. Т. 1. С. 165-180.
7. Сушкевич Т.А. К истории первого научного эксперимента по дистанционному зондированию Земли на пилотируемом космическом корабле // Современные проблемы дистанционного зондирования Земли из космоса. 2008. В. 5. Т. 1. С. 315-322.
8. Сушкевич Т.А. Осесимметричная задача о распространении излучения в сферической системе. / Отчет ИПМ АН СССР, № О-572-66. М.: ИПМ АН СССР, 1966. 180 с.
9. Гермогенова Т.А., Копрова Л.И., Сушкевич Т.А. Исследование угловой, пространственной и спектральной структуры поля яркости Земли для характерной модели сферической атмосферы // Изв. АН СССР. Серия Физика атмосферы и океана. 1969. Т. 5. № 12. С. 1266-1277.
10. Альтовская Н.П., Розенберг Г.В., Сандомирский А.Б., Сушкевич Т.А. Поле яркости зари, наблюдаемой с космических кораблей // Изв. АН СССР. Серия Физика атмосферы и океана. 1971. Т. 7. № 3. С. 279-290.
11. Альтовская Н.П., Розенберг Г.В., Сандомирский А.Б., Сушкевич Т.А. Некоторые результаты фотометрических исследований дневного горизонта Земли с космических кораблей «Союз-4» и «Союз-5» // Изв. АН СССР. Серия Физика атмосферы и океана. 1971. Т. 7. № 6. С. 590-598.
12. Сушкевич Т.А. Об одном методе решения уравнения переноса для задач с двумерной сферической геометрией. М.: 1972, 31 с. (Препринт / ИПМ АН СССР, № 15). Депонирован, № 5557-73 от 28.02.73.
13. Розенберг Г.В., Сандомирский А.Б., Сушкевич Т.А., Матешвили Ю.Д. Исследование стратификации аэрозоля в стратосфере по программе «Союз-Аполлон» // Изв. АН СССР. Серия Физика атмосферы и океана, 1980. Т. 16. № 4. С. 861-864.
14. Численное решение задач атмосферной оптики // Сборник научных трудов ИПМ им. М.В.Келдыша АН СССР. / Под ред. Масленникова М.В. и Сушкевич Т.А. М.: ИПМ им. М.В.Келдыша АН СССР, 1984. 234 с.
15. Сушкевич Т.А., Стрелков С.А., Иолтуховский А.А. Метод характеристик в задачах атмосферной оптики. М.: Наука, 1990. 296 с.
16. Сушкевич Т.А. Математические модели переноса излучения. М.: БИНОМ. Лаборатория знаний, 2005. 661 с.

# Моделирование магнитных свойств двумерной металлической пены\*

О.Д. Клименкова

Национальный исследовательский университет «Высшая школа экономики»

Проведено численное моделирование магнитных свойств пенной двумерной структуры. Моделирование основано на предложенных алгоритмах генерации двумерной пены и модели пенного магнетика. Применение алгоритма генерации двумерной пены позволяет создавать двумерные механически жесткие структуры с минимальной плотностью, в сотни раз менее плотной, чем обычное кристаллическое твердое тело. Предложенная простейшая модель магнитных свойств пенной структуры корректно воспроизводит магнитные свойства пены палладия, наблюдаемые экспериментально. Для исследования магнитных свойств мы применили два метода – локальный метод Метрополиса и кластерный метод Вольфа. Для представления взаимодействия магнитных спинов предложена специальная структура – карта соседей, элементы которой зависят как от реализации неупорядоченной разреженной пены, так и от дополнительного параметра, радиуса взаимодействия.

*Ключевые слова:* модель Изинга, двумерная пена, компьютерное моделирование

## 1. Введение

Интерес к пенным структурам обусловлен, в частности, их использованием в качестве хранилища водорода в водородной энергетике. Отличительная черта пенной структуры состоит в очень низкой, относительно сплошного материала с тем же химическим составом, плотностью. При этом, пенная структура даже при плотности в доли процента от объемной имеет хорошую механическую стабильность [1, 2]. Имеются указания на то, что особо эффективные свойства хранилищ водорода, изготовленных из пены палладия, могут быть связаны с особенными магнитными свойствами палладия [3]. В то время как палладий в сплошной форме является диамагнитным, его низкоразмерные формы могут иметь свойства магнетика.

Настоящая работа посвящена изучению простейшей модели магнитной двумерной пены. Для изучения свойств магнитной пены необходимо решить две подзадачи. Во-первых, требуется разработка методики генерации двумерной пены, близкой по свойствам к реальной. Во-вторых, требуется разработать магнитную модель с использованием пространственной геометрии пены.

Первая подзадача решена в нашей статье [14] и в настоящей статье мы излагаем алгоритм в разделе 2. В настоящей статье мы предлагаем модель магнитной пены в разделе 3 и исследуем ее свойства в разделе 5. Описание использованных алгоритмов и детали их реализации изложены в разделе 4.

## 2. Алгоритм генерации двумерной пены

Для генерации двумерной пены в работе [14] был предложен следующий алгоритм:

1. Задаем квадратную область с линейным размером  $L$ ,
2. Фиксируем длину прямолинейного волокна  $l$ ,

---

\*Работа выполнена при поддержке РФФИ грант 20-37-90085-аспиранты "Разработка алгоритмов генерации разреженных жестких структур"



3. Генерируем тройку случайных чисел  $(x_i, y_i), \phi_i$ , из которых два числа  $(x_i, y_i)$  задают координаты начальной точки волокна внутри квадрата ( $0 < x_i, y_i < L$ ), а  $\phi_i \in [0, 2\pi)$  задает угол волокна относительно горизонтальной оси,
4. Учитываем только ту часть волокна, которая оказалась внутри квадрата,
5. Запоминаем все пересечения волокна со всеми предыдущими волокнами,
6. Повторяем шаги 3, 4 и 5 алгоритма до тех пор, пока волокна не составят кластер, имеющий пересечения с каждой из четырех сторон квадрата,
7. Из полученного кластера итеративно удаляем все волокна, имеющие с ним только одно пересечение. Эта процедура похожа на описанное в экспериментах “промывание” пены [5].
8. Сохраняем полученный кластер для дальнейшего моделирования.

Для описания волокна введем класс Needle:

```
public fields:
    double x_start;
    double y_start;
    double phi;
    double x_end;
    double y_end;
    double length;
    std::vector<Point> Spins;
    // number of cluster to which the needle belongs
    int cluster;
    // flags of crossing the sides of the box
    bool intersect_0;
    bool intersect_1;
    bool intersect_2;
    bool intersect_3;
public functions:
    // find intersection with other Needle
    Point find_intersect(Needle otherNeedle)
```

Для описания кластера введем класс Cluster:

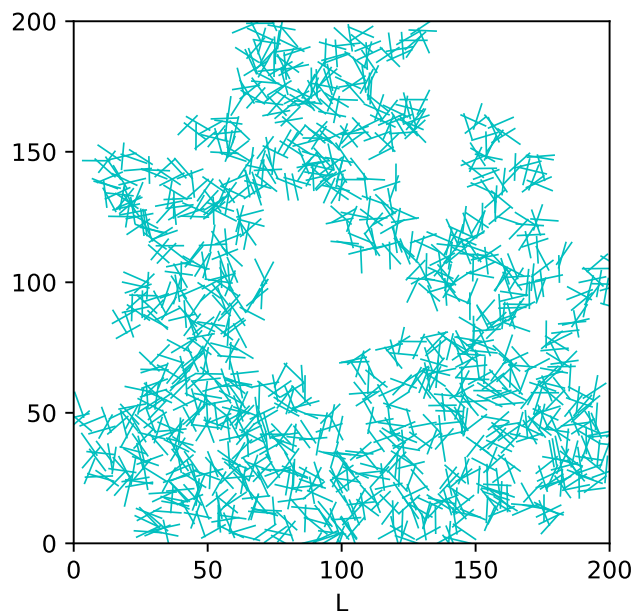
```
public fields:
    int numberCluster;
    bool intersect_0;
    bool intersect_1;
    bool intersect_2;
    bool intersect_3;
    std::vector<int> Needles;
public functions:
    Cluster union_cluster(Cluster otherCluster)
```

В начальный момент создается четыре кластера, которые состоят из специальных волокон на внешних сторонах квадрата. При генерации каждого нового волокна (Needle), оно будет либо добавлено в новый кластер, если пересечений ни с какими уже сгенерированными волокнами не обнаружено, либо это волокно будет добавлено к существующему кластеру, если имеется с ним пересечение. Соединение кластером четырех внешних сторон (волокон длиной  $L$ ) и будет окончанием процедуры добавления волокон.

Далее мы удаляем из рассмотрения все волокна не попавшие в этот кластер (список волокон хранится в структуре Cluster.Needles). Затем кластер подвергается “промыванию” – итеративно удаляем те волокна, которые имеют только одно пересечение, если это не пересечение со сторонами квадрата.

Сгенерированные таким образом структуры имеют низкую плотность, ниже плотности протекающего кластера, и при этом являются “жесткими” – они лежат на краях квадрата и имеют внутри не менее одного пересечения между волокнами.

На рис. 1 изображен пример сгенерированной двумерной структуры. Этот алгоритм работает достаточно быстро (менее часа) на одном узле кластера НИУ ВШЭ sHARISMa, без дополнительного распараллеливания и использования GPU, для наших исследуемых размеров бокса и длины волокна.



**Рис. 1.** Типичный пример сгенерированной двумерной пены. Линейный размер квадрата  $L = 200$ , длина волокна  $l = 10$ .

### 3. Модель магнитной пены

Нам не известны вычислительные работы по исследованию термодинамических свойств пенных структур. Мы предлагаем простейшую модель магнитной пены.

На рис. 2 разноцветными кружками отмечены положения магнитных спинов на модели двумерной пены. Используя метод Монте-Карло будем вычислять термодинамические средние:

намагниченность

$$M(t) = N_{up} - N_{down},$$

теплоемкость

$$C = \frac{\partial E}{\partial T} = \frac{(\Delta E)^2}{T^2} = \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2},$$

восприимчивость

$$\chi = \frac{\partial M}{\partial T} = \frac{(\Delta M)^2}{T^2} = \frac{\langle M^2 \rangle - \langle M \rangle^2}{T^2},$$

где  $N_{up}$  и  $N_{down}$  - количество спинов со значением +1 (up) и -1 (down) соответственно,  $E$  - внутренняя энергия,  $T$  - температура в единицах энергии и  $J$  - константа взаимодействия спинов.

В одномерном случае (одиночное волокно) корреляционная длина зависит от температуры как  $\xi \propto (1/t)$  где  $t = \exp(-2J/T)$  [15]. Таким образом, если  $\xi < p/4$ , где  $p$  - средний

периметр многоугольника образованного волокнами, то система будет вести себя, как большое количество конечных одномерных объектов. Ожидается, что с ростом размера решетки переход вокруг  $\xi \approx p/4$  будет все более заметен. Кроме того, в системе при  $\xi > p/4$  - будет наблюдаться поведение двумерной примесной системы, то есть максимум теплоемкости будет расти с увеличением размера решетки, также и восприимчивости.

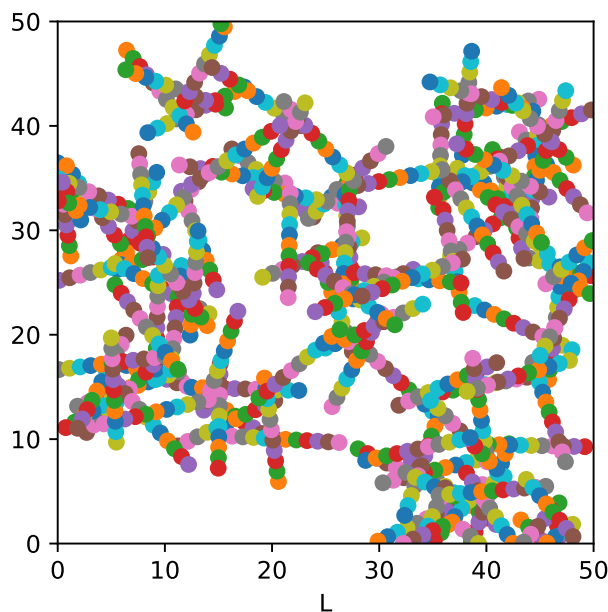


Рис. 2. Спины на волокнах смоделированной пены, размер квадрата  $L = 50$ , размер волокна  $l = 10$ .

#### 4. Алгоритм моделирования магнитной пены

Алгоритм реализован на языке Python 3.7. Расчеты велись на высокопроизводительном вычислительном кластере НИУ ВШЭ sHARISMA состоящем из 40 вычислительных узлов, в том числе 29 специализированных узлов с большим объемом оперативной памяти, модель процессора Intel Xeon Gold. В данном исследовании GPU использованы не были.

Взаимодействие спинов определяет энергию

$$E = - \sum_{r \leq R} J s_i s_j. \quad (1)$$

В дальнейшем мы полагаем константу взаимодействия  $J = 1$ . Начальное распределение спинов случайное, соответствующее бесконечной температуре  $T$ . Дополнительный параметр нашей модели – радиус взаимодействия спинов  $R$ . Иными словами, суммирование в выражении (1) производится по всем парам спинов  $s_i$  и  $s_j$ , расстояние  $r$  между которыми не превосходит значение  $R$ .

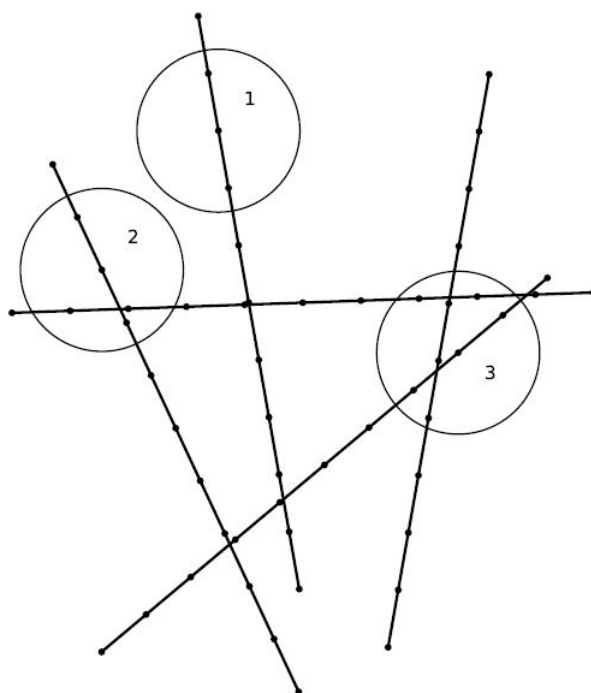
Для реализации алгоритма Монте-Карло мы задаем карту взаимодействующих спинов, это нужно чтобы быстрее пересчитывать энергию системы, учитывая только соседей рассматриваемого спина. Для этого строится матрица  $N \times N$ . Заполняем эту матрицу только под главной диагональю - ставим единицы, где спины  $i, j$  (столбец, строка) находятся на расстоянии не более  $R$ . Так же составляем словарь, где ключ - номер спина, а значение - массив из соседей спина, находящихся на расстоянии не более  $R$ .

Перед началом моделирования считаем энергию всей системы  $E$ , используя выражение (1).

Для каждой пары значений  $R$  и  $T$  моделирование осуществляется параллельно с использованием всех 44 ядер узла. Сложность такой реализации  $O(M * N)$ , где  $M$  это число Монте-Карло шагов и  $N$  – количество спинов. За один шаг Монте-Карло принимаем  $N$  элементарных шагов по Метрополису.

Элементарный шаг по Метрополису осуществляется следующим образом. Выбираем случайно спин и пересчитываем энергию с учетом изменения знака выбранного спина, используя при этом только нужных спинов из словаря соседей. Если энергия системы уменьшилась или не изменилась, то изменяем значение спина на противоположное. Если энергия системы увеличилась, то изменяем значение спина на противоположное с вероятностью  $\exp(-(E_{new} - E_{old})/T)$ . При изменении значения спина принимаем новое значение энергии.

На рис. 3 изображен пример взаимодействия спинов на расстоянии  $R$ . У каждого спина свое количество соседей, и в алгоритме мы запоминаем всех его соседей, потому что во время моделирования количество соседей не меняется.



Акт  
11-12

Рис. 3. Примеры соседей на расстоянии  $R$ .

Один Монте-Карло шаг можно представить так

```
E_old = calculate_energy(map_spins, map_distance)
for i_spin in range(Number_spins):
    random_index = random.randint(0, len(map_spins)-1)
    E_oldspin =
        calculate_spin_energy(map_spins, map_distance_dict, random_index)
    map_spins[random_index] = -map_spins[random_index]
    E_newspin = -E_oldspin
    E_new = E_old - E_oldspin + E_newspin
    if (E_new <= E_old):
        E_old = E_new
        continue
    else:
        probability = np.exp(-(E_new - E_old)/T)
        value = random.uniform(0,1)
```

```

if (value < probability):
    E_old = E_new
else:
    map_spins[random_index] = -map_spins[random_index]

```

Термализовать в системе можно также с помощью алгоритма Вольфа [16], один шаг которого состоит из построения кластера коррелированных спинов

1. Выбираем случайно спин  $(i, j)$ ,
2. Определяем по списку всех соседей выбранного спина на расстоянии  $R$ , имеющих совпадающее направление спина.
3. Связь с такими соседями принимаем с вероятностью  $p = 1 - e^{-2K}$ , где  $K = \frac{J}{T}$  и при этом помещаем соседей в стек.
4. Меняем значение спина  $(i, j)$  на противоположное – таким образом мы исключаем его повторный выбор,
5. Далее из стека выбираем следующий спин  $(i, j)$ ,
6. Повторяем шаги 2-5, до тех пор, пока стек не пуст.

Для обхода всех соседей спина используется метод обхода в графе “в глубину”. Для этого используется структура стек. Всех соседей для каждого спина считаем один раз и складываем в словарь.

```

def Wolff_Algorithm(map_spins, map_distance_dict, random_index, prob):
    cluster = []
    cluster.append(random_index)
    base_spin = map_spins[random_index]
    map_spins[random_index] = -map_spins[random_index]
    while (len(cluster) != 0):
        last_element = cluster.pop()
        for neighbor in map_distance_dict[last_element]:
            if (map_spins[neighbor] == base_spin):
                value = random.uniform(0,1)
                if (value < prob):
                    cluster.append(neighbor)
                    map_spins[neighbor] = -map_spins[neighbor]
    return map_spins

```

## 5. Результаты

Для  $R = 1.2888$  и  $T = 6.0$  время вычисления одного Монте-Карло шага равно 0.01492(2) секунд, а для одного прохода алгоритма Вольфа это время больше 0.6327(4). Однако, алгоритм Вольфа имеет значительно меньшее время корреляций между шагами, что позволяет достичь большей точности вычислений. Среднее количество соседей спинов для  $R = 1.2888$  равно 4.8.

На рис. 4 представлен график теплоемкости для разных значений  $R$ , а на рис. 5 показаны графики восприимчивости для тех же значений радиуса взаимодействия  $R$ .

## 6. Заключение

В работе изложен алгоритм для генерации двумерной пенной структуры, а также предложена модель магнитных свойств такой структуры. Предварительное моделирование указывает на качественное сходство вычисленных термодинамических величин с полученными экспериментальными данными. Заметим, что в эксперименте пенная структура трехмерная. Совпадение численных и экспериментальных результатов указывает на то, что по-видимому

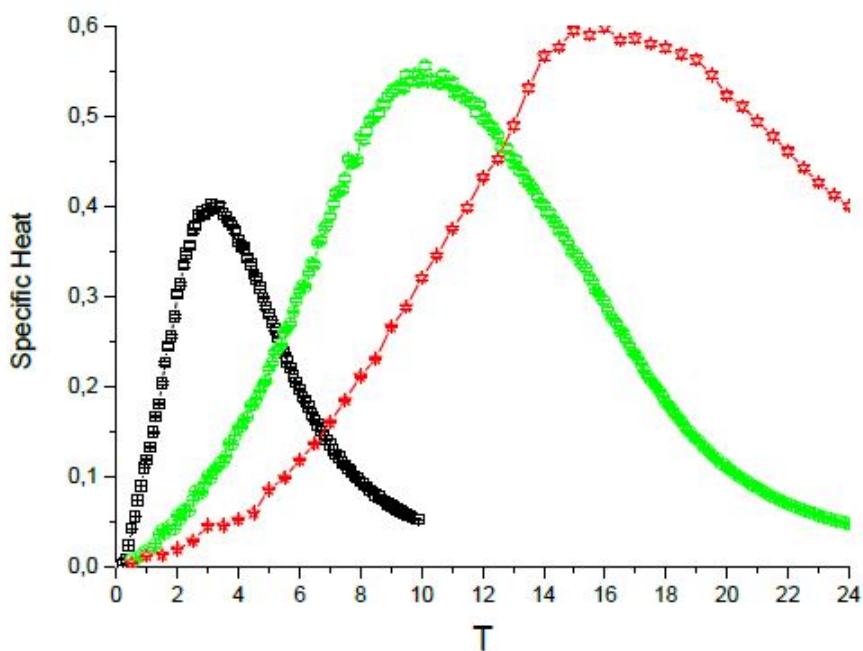


Рис. 4. График теплоемкости для  $R=1.2888$  (квадраты), 2.4265 (кружки), 3.222 (звездочки).

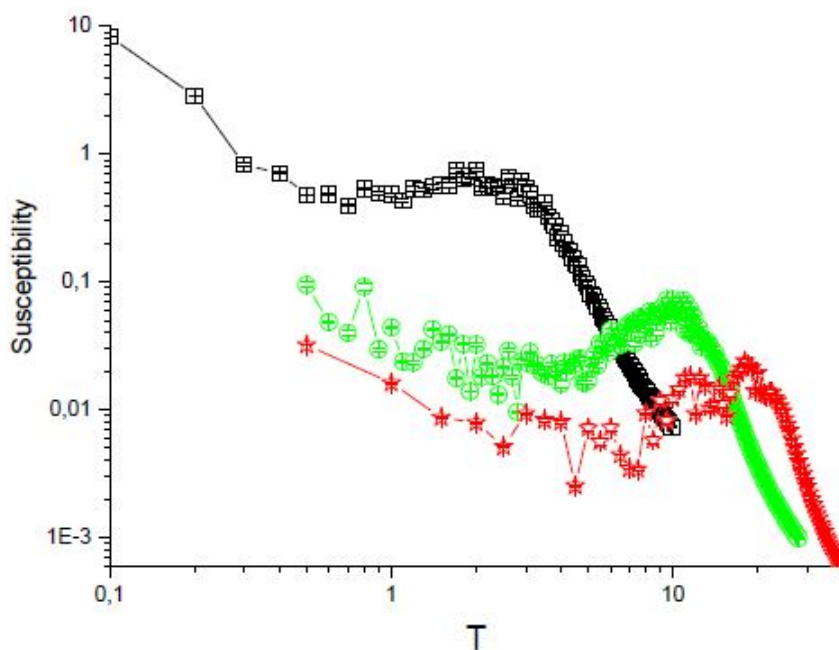


Рис. 5. График восприимчивости для  $R=1.2888$  (квадраты), 2.4265 (кружки), 3.222 (звездочки).

при низких температурах магнитное поведение определяется квази-одномерными свойствами.

Автор благодарит Л.Н. Щура за научное руководство и помощь в подготовке рукописи.

## Литература

1. M. R. Islam and R. C. Picu, Phys. Rev. E 99 (2019) 063001.
2. R. C. Picu, Soft Matter 7 (2011) 6768.
3. D.K. Pradhan et al., Sci. Reports 9 (2018) 1685.
4. E. Antolini, Energy Environ sci. **2** (2009) 915.
5. D.A. Gilbert et al., Chemistry of Materials **29** (2017) 9814.
6. P.I. Richards, PNAS, **52** (1964), 1160.
7. A. Kumar, M.M. Mohammadi, and M.T. Swihart, Nanoscale, **11** (2019) 19058.
8. A.M. Stein, D.A. Vader, L.M. Jawerth, D.A. Weitz, and L.M. Sander, J. Microscopy, **232** (2008) 463.
9. J. Hoshen and R. Kopelman, Phys. Rev. B. **14** (1976) 8.
10. The generated structures are not sensitive to the choice of the random numbers as we check using functions MT19937, MRG32K3A, and LFSR113 from the libraries [11, 12].
11. L.Yu. Barash, L.N. Shchur, Comput. Phys. Commun., **182** (2011) 1518; L.Yu. Barash, L.N. Shchur, Comput. Phys. Commun., **184** (2013) 2367.
12. M.S. Guskova, L.Yu. Barash, L.N. Shchur, Comp. Phys. Commun., **200** (2016) 402.
13. M. de Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, (Springer, 2008)
14. O. Klimenkova, L. Shchur, Algorithm for foam generation in plane Journal of Physics: Conference Series. 2021. Vol. 1740. No. 012030. P. 1-5.
15. Baxter R. J. Exactly solved models in statistical mechanics. – Elsevier, 2016.
16. Wolff, U. Collective Monte Carlo updating for spin systems. Physical Review Letters 62.4 (1989): 361.

# Нейросетевой метод сравнительного анализа трехмерных структур белков на основе суперсемейств доменов

А.В. Мазеев<sup>1</sup>, Н.Н. Попова<sup>1</sup>, Д.А. Сулатов<sup>2</sup>

<sup>1</sup>МГУ им. М.В. Ломоносова, факультет вычислительной математики и  
кибернетики,

<sup>2</sup>МГУ им. М.В. Ломоносова, факультет биоинженерии и биоинформатики

Исследование белков является одной из важных современных задач биологии. При решении многих задач биоинформатики возникает необходимость по заданной структуре белка определить структурно схожие с этим белком известные белки. Поиск схожих по трехмерной структуре известных белков помогает понять функции белка. В работе предлагается нейросетевой метод сравнительного анализа структур заданной пары белков. Предложенный метод позволяет быстро отобрать в базе данных структур потенциально схожие по структурам белки. В работе обсуждаются архитектура и параметры предложенной нейросети и предлагается метод выделения признаков рассматриваемых структур. Обучение нейросети предлагается проводить с использованием имеющихся баз данных с выделенными доменами структурно похожих белков. В качестве решения для обучения нейросети предлагается использовать базу данных доменов CATH S40. Рассматривается метод перевода трехмерной структуры белка в вектор признаков при помощи анализа расстояний между парами центральных атомов аминокислот. Предлагается несколько вариантов предложенного подхода. Приводятся результаты экспериментального исследования разработанного метода.

*Ключевые слова:* парное сравнение белков, трехмерная структура белка, нейросеть

## 1. Введение

Сравнительный анализ трехмерных структур белков широко используется при решении многих задач биоинформатики. Наиболее важным источником белковых структур является база данных PDB (Protein Data Bank) [1], в которой собирается информация об экспериментально определенных белковых структурах и их атомных координатах. Число структур в этой базе постоянно увеличивается. Поиск белков, структурно похожих на заданный белок, является актуальной задачей.

Поиск похожих белков по всей базе PDB существующими методами с применением структурного выравнивания занимает десятки минут, а иногда даже много часов. Одним из подходов, направленных на сокращение этого времени, является так называемый двухфазный подход. Согласно этому подходу на первом этапе происходит быстрый фильтр заведомо неподходящих белков, а на втором этапе — среди возможно подходящих белков более точными и, соответственно, медленными методами отбираются похожие белки (для заданного белка). В итоге с помощью такого подхода можно существенно сократить время поиска.

Целью данной работы является разработка метода предварительной фильтрации базы данных, способного с высокой точностью отбирать белки из базы данных для последующего применения более точных методов с построением выравнивания. В статье описывается новый метод парного сравнения белков без построения выравнивания. Для решения задачи парного сравнения белков рассматривается возможность обучения нейросети на базе данных доменов белков CATH S40 [2].

База данных CATH S40 представляет собой множество доменов белков, разбитых на классы гомологов, то есть родственных и схожих по строению белков. Соответственно, для



любой пары белков из одной произвольной группы известно, что они схожи по трехмерной структуре, а для любой пары белков из различных групп известно, что они различаются по трехмерной структуре. Отмеченный факт может быть использован для построения обучающей выборки, используемой для обучения нейросети. Для этого необходимо разработать метод для преобразования белка в объект (вектор или матрицу) фиксированного размера. Анализ трехмерных структур белков методами машинного обучения считается непростой задачей, так как белки имеют различный размер.

## 2. Обзор методов выравнивания трехмерных структур белков

Выравнивание белков помогает определить наличие эволюционной связи между белками, что в свою очередь позволяет понять функции белка. Пространственное выравнивание подходит для сравнения белков с непохожими последовательностями в том случае, когда эволюционная связь не может быть установлена стандартными методами выравнивания последовательностей.

Методы сравнения трехмерных структур белков активно развиваются. Наиболее известные реализации этих методов — MADOKA [3], FAST [4], DALI [5], 3D-Hit [6], CE [7], SAL [8], FATCAT [9], TM-align [10], Fr-TM-align [11], DeepAlign [12], MICAN-SQ [13], и другие. Авторы одной из последних работ — метода MADOKA [3] утверждают, что их решение позволяет выполнить поиск белков, похожих на заданный, по всей базе данных PDB за полчаса. Однако, авторы отмечают, что время поиска зависит от входного белка. MADOKA — двухфазный метод, на первом шаге которого происходит вычисление наибольшей общей подпоследовательности для последовательностей вторичных структур белков. На втором шаге выполняется более точный метод с применением выравнивания трехмерных структур белков.

Для большинства методов выравнивания задача поиска похожих белков по базе PDB может занимать много часов или даже дней.

Применение метода машинного обучения для анализа белков рассматривается в ряде работ. Например, в работах [14, 15] предлагается преобразовать белок в изображение, используя нейросетевой подход. Преобразование белка в изображение (или матрицу) для дальнейшего анализа — известный подход, который, однако, имеет много вариаций.

Эвристические подходы для сравнения трехмерных структур белков без построения выравнивания и без применения машинного обучения рассматриваются в работах [16–18].

## 3. Предлагаемый нейросетевой метод

Для решения задачи сравнительного анализа трехмерных структур белков в данной работе предлагается нейросетевой метод. Метод основан на использовании в качестве обучающей выборки базы данных доменов белков CATH S40. Основная проблема такого решения — не ясно, как подавать структуры белка на вход нейросети. С одной стороны, есть известный, упомянутый выше, метод построения матрицы расстояний для заданного белка:

$$D[i, j] = \text{dist}(C\alpha[i], C\alpha[j]),$$

где  $C\alpha[i]$  —  $C\alpha$  (центральный) атом  $i$ -ой аминокислоты белка,  $\text{dist}(a, b)$  — евклидово расстояние между атомами  $a$  и  $b$ , а  $1 \leq i, j \leq N$ ,  $N$  — количество атомов структуры белка.

Однако, для белков различных размеров, размеры соответствующих матриц также различны. Кроме того, размер такой матрицы квадратично зависит от размера белка, что в свою очередь требует больших затрат памяти при хранении матриц для всех структур из крупной базы данных, таких как PDB или CATH.

Для унификации представления различных структур белков предлагается ввести вектор  $V$  распределения расстояний между всеми парами  $C\alpha$  атомов. Действительно, если

трехмерные структуры белков схожи, то и соответствующие векторы признаков также будут похожи, и наоборот, для различных белков векторы признаков также будут различаться.

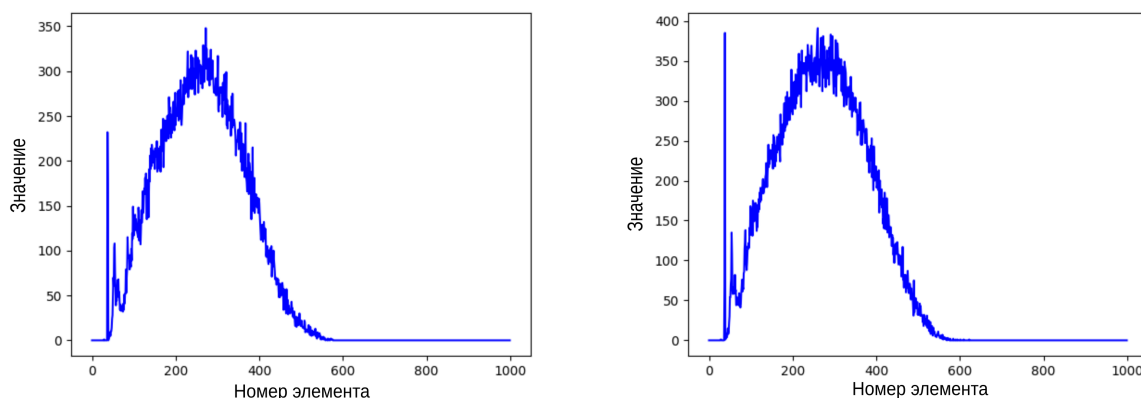
Вектор предлагается строить следующим образом. Расстояния больше 100 ангстрем не учитываются — подавляющее большинство расстояний между атомами белков меньше указанного значения. Обозначим длину вектора —  $L$ . В данной работе рассматривается два варианта выбора параметра  $L$ :  $L = 100$  и  $L = 1000$ .

$$V[i] = f(i * 100/L, (i + 1) * 100/L),$$

где  $0 \leq i < L$ ,  $f(a, b) = \sum_{i \neq j} \sigma(i, j)$ , где

$$\sigma(i, j) = \begin{cases} 0.5, & \text{dist}(C\alpha[i], C\alpha[j]) \in (a, b), \\ 0, & \text{dist}(C\alpha[i], C\alpha[j]) \notin (a, b) \end{cases}$$

Пример векторов признаков  $V$  при  $L = 1000$  для похожей пары доменов белков (гомологов) показан на рис. 1, а для пары различных доменов белков (не гомологов) — на рис. 2.



а) Вектор признаков для домена 1h12A00      б) Вектор признаков для домена 3gzkA02

**Рис. 1.** Пример векторов признаков для похожих доменов белков (гомологов).

Можно заметить, что для похожих доменов вектора получаются похожие, и наоборот.

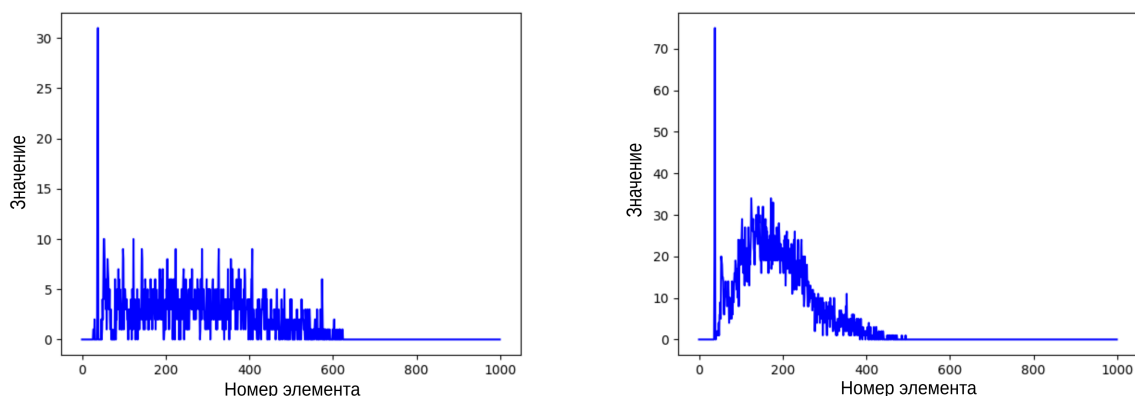
Соответственно, для каждого белка в базе данных CATH S40 можно построить вектор  $V$ , и сформировать тем самым выборку для обучения и тестирования нейронной сети. С помощью данного подхода можно обучать нейросеть и другие модели машинного обучения с целью дальнейшего применения их к сравнению структур белков.

Рассмотрено два варианта построения и обучения нейронной сети с использованием построенных векторов признаков. В случае первого варианта для векторов признаков  $V_1$  и  $V_2$  двух доменов белков можно вычислять абсолютную разницу соответствующих элементов, то есть строить вектор  $V_{diff}$ :

$$V_{diff}[i] = |V_1[i] - V_2[i]|,$$

где  $0 \leq i < L$ . Второй вариант заключается в том, чтобы нейросеть получала на вход оба вектора  $V_1$  и  $V_2$ . При таком подходе на вход нейросети каждый элемент тренировочной выборки подается дважды: пара  $(V_1, V_2)$  и наоборот — пара  $(V_2, V_1)$ .

Программная реализация предложенного подхода выполнена на языке программирования Python, с использованием библиотеки keras. Используется метод обратного распространения ошибки (англ. backpropagation).



а) Вектор признаков для домена 3u88M00      б) Вектор признаков для домена 2wzpR02

Рис. 2. Пример векторов признаков для непохожих доменов белков (не гомологов).

#### 4. Экспериментальное исследование предложенного метода

Для решения рассматриваемой задачи была выбрана многослойная полносвязная нейросеть. В результате тестирования различных вариантов была определена архитектура с пятью слоями. Большее количество слоев не улучшало качество, однако увеличивало время обучения. Размеры слоев (через запятую):  $L, 1.2 \cdot L, 0.4 \cdot L, 0.1 \cdot L, 2$  — для варианта с разницей векторов. В случае обучения на парах векторов размеры слоев (кроме выходного слоя) были вдвое больше.

Как было отмечено выше для обучения и тестирования модели нейронной сети была выбрана база данных SATH S40. Эта база часто используется в подобных задачах. База данных SATH S40 состоит из множества доменов белков, процент схожести аминокислотных последовательностей которых не более 40%.

Выборка для обучения и тестирования нейросети в данном эксперименте состояла из 100000 случайно выбранных пар доменов белков из базы данных SATH S40. Половина пар — похожие домены, то есть гомологи, другая половина — непохожие домены (не гомологи). Тестовая выборка состояла из 20%, тренировочная, соответственно — 80% от всей выборки.

Размеры базы данных SATH S40: количество семейств доменов — 6576, общее количество доменов белков — 31885 (суммарно по всем семействам). Время предобработки базы данных (вычисление векторов признаков для всех доменов) — 1868 секунд. Также выполнена параллельная реализация предобработки данных с использованием библиотеки multiprocessing в языке программирования Python. Время выполнения на четырех ядрах — 761 секунда (ускорение по сравнению с последовательной версией равно 2.45). Это время предобработки верно как для  $L = 100$ , так и для  $L = 1000$ , так как вычислительная сложность в обоих случаях одинаковая. Такую предобработку всех белков в базе необходимо выполнить только один раз для заданного  $L$ .

Вычисления проводились на обычном ноутбуке: процессор Intel Core i5-8250U, 4 ядра, 6 МБ кэш-памяти.

Для оценки качества решения рассматривались две величины: точность (англ. accuracy) — доля правильных ответов нейросети, и AUC (англ. Area Under the ROC Curve) — площадь под ROC кривой. ROC кривая (англ. Receiver Operating Characteristic) — график, позволяющий оценить качество бинарной классификации. ROC кривая отображает соотношение между долей объектов от общего количества носителей признака, верно классифицированных как несущие признак, и долей объектов от общего количества объектов, не несущих признака, ошибочно классифицированных как несущие признак, при варьировании порога решающего правила.

Таблица 1. Результаты обучения нейронной сети.

Параметры подхода	Точность (англ. accuracy)	AUC	Время обучения (сек.)	Время применения (сек.)
L=100, пара векторов	87.81	0.949	407.38	1.01
L=100, разница векторов	83.19	0.909	218.08	0.88
<b>L=1000, пара векторов</b>	<b>88.95</b>	<b>0.9517</b>	10721.65	23.99
L=1000, разница векторов	82.56	0.887	2507.57	7.85

AUC может принимать значения в отрезке  $[0, 1]$ . Чем больше значение AUC, тем качественнее классификатор. Значение менее 0.5 говорит о том, что классификатор будет работать лучше, если отрицательные вердикты заменить на положительные, и наоборот.

На рис. 3 продемонстрированы ROC кривые для разных вариантов разработанного метода. Оказалось, что варианты с подачей обоих векторов на вход нейросети показывают более хороший результат. Наилучшее качество на тестовой выборке достигнуто с параметром  $L = 1000$ , при обучении нейросети на парах векторов.

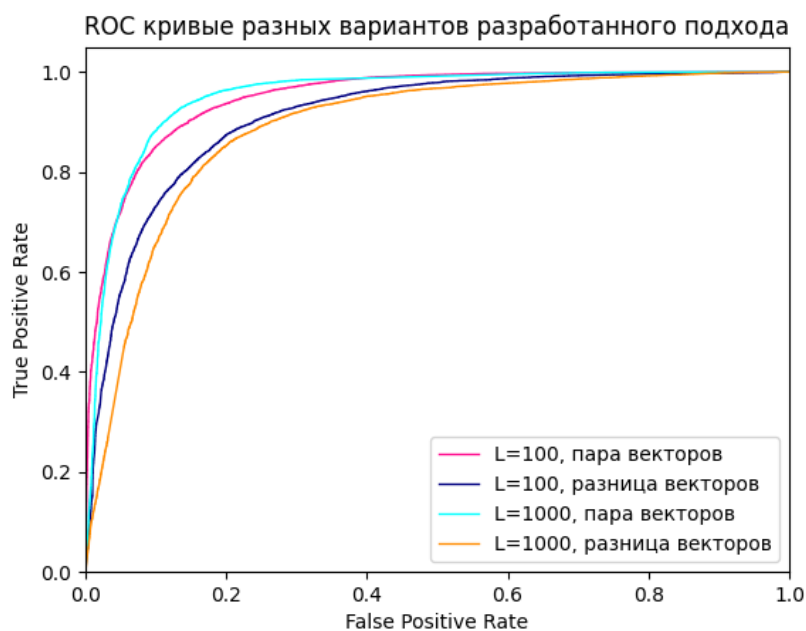


Рис. 3. График ROC кривых для различных вариантов разработанного подхода.

В таб. 1 приведены точность, AUC, время обучения нейросети и время применения обученной модели. Время применения определялось для 100000 случайно взятых пар доменов из базы CATH S40.

## 5. Заключение

В работе предложен нейросетевой метод для сравнительного анализа трехмерных структур белков. Разработан метод выделения признаков рассматриваемых структур. Предложенная нейросетевая модель представляет собой 5-слойную полносвязную нейросеть. Обучение нейросети проведено с использованием имеющихся баз данных с выделенными группами структурно похожих белков. Предложенный метод позволяет быстро отобрать в базе данных структур потенциально схожие по структуре белки. Проведенный эксперимент продемонстрировал достаточно высокую точность (ассигасу) — 88.95. Время применения обученной модели к 100000 случайных пар доменов белков составило 23.99 секунд. Для сравнения — известная среди биологов программа МАТТ (англ. Multiple Alignment with Translations and Twists) работает в 2339 раз дольше на этих же данных. Однако МАТТ при этом работает с точностью, близкой к 100%, а также строит выравнивание структур белков. Эксперимент выполнен на обычном ноутбуке.

В дальнейшие планы входит применение других моделей машинного обучения для рассматриваемой задачи, улучшение качества решения, исследование возможности применения разработанного подхода к цепям белков, вместо доменов, а также проведение поиска схожих белков по всей базе данных PDB.

## 6. Благодарности

Работа выполнена при финансовой поддержке гранта РФФИ № 20-07-01053 (Мазеев А.В. и Попова Н.Н.) и гранта РФФИ № 18-29-13060 (Суплатов Д.А.).

## Литература

1. Berman J., Westbrook Z., Feng G., et al. The Protein Data Bank // *Nucleic Acids Research*. 2000. Vol. 28. P. 235–242. DOI: 10.1093/nar/28.1.235.
2. Jones S., Jones D., Swindells M., Thornton J. CATH – a hierarchic classification of protein domain structures // *Structure*. 1997. Vol. 5, No. 8. P. 1093–1109. DOI: 10.1016/s0969-2126(97)00260-8.
3. Deng L., Zhong G., Liu C., et al. MADOKA: an ultra-fast approach for large-scale protein structure similarity searching // *BMC Bioinformatics*. 2019. Vol. 20, No. 19, 662 p. DOI: 10.1186/s12859-019-3235-1.
4. Zhu J., Weng Z. Fast: a novel protein structure alignment algorithm // *Proteins*. 2005. Vol. 58, No. 3. P. 618–627. DOI: 10.1002/prot.20331.
5. Holm L., Sander C. Protein structure comparison by alignment of distance matrices // *Journal of Molecular Biology*. 1993. Vol. 233, No. 1. P. 123–138. DOI: 10.1006/jmbi.1993.1489.
6. Bieniasz-Krzywiec Ł., Cytowski, M., Rychlewski L., et al. 3D-Hit: fast structural comparison of proteins on multicore architectures // *Optim Lett*. 2014. Vol. 8. P. 1783–1794. DOI: 10.1007/s11590-013-0697-3.
7. Shindyalov I., Bourne P. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path // *Protein Engineering*. 1998. Vol. 11, No. 9. P. 739–747. DOI: 10.1093/protein/11.9.739.
8. Kihara D., Skolnick J. The pdb is a covering set of small protein structures // *Journal of Molecular Biology*. 2003. Vol. 334, No. 4. P. 793–802. DOI: 10.1016/j.jmb.2003.10.027.

9. Ye Y., Godzik A. Flexible structure alignment by chaining aligned fragment pairs allowing twists // *Bioinformatics*. 2003. Vol. 19, No. 2. P. 246–255. DOI: 10.1093/bioinformatics/btg1086.
10. Zhang Y., Skolnick J. Tm-align: a protein structure alignment algorithm based on the tm-score // *Nucleic Acids Res*. 2005. Vol. 33, No. 7. P. 2302–2309. DOI: 10.1093/nar/gki524.
11. Pandit S., Skolnick J. Fr-tm-align: a new protein structural alignment method based on fragment alignments and the tm-score // *BMC Bioinformatics*. 2008. Vol. 9, No. 531. DOI: 10.1186/1471-2105-9-531.
12. Wang S., Ma J., Peng J., Xu J. Protein structure alignment beyond spatial proximity // *Scientific Reports*. 2013. Vol. 3, No. 1448. DOI: 10.1038/srep01448.
13. Minami S., Sawada K., Ota M., Chikenji G. Mican-sq: a sequential protein structure alignment program that is applicable to monomers and all types of oligomers // *Bioinformatics*. 2018. Vol. 34, No. 19. P. 3324–3331. DOI: 10.1093/bioinformatics/bty369.
14. Sikosek T. Protein structure featurization via standard image classification neural networks. URL: <https://www.biorxiv.org/content/10.1101/841783v1> (дата обращения: 05.04.2021).
15. Tsuchiya Y., Tomii K. Neural networks for protein structure and function prediction and dynamic analysis // *Biophys. Rev*. 2020. Vol. 12, No. 2. P. 569–573. DOI: 10.1007/s12551-020-00685-6.
16. Zhang L., Bailey J., Konagurthu A.S., et al. A fast indexing approach for protein structure comparison // *BMC Bioinformatics*. 2010. Vol. 11. DOI: 10.1186/1471-2105-11-S1-S46.
17. Petrova D. Protein structure comparison based on distances between secondary structure elements and backbone dihedral angles // *Proceedings of the 12th International Conference on Computer Systems and Technologies, CompSysTech 2011, June, Vienna, Austria*. P. 359–364. DOI: 10.1145/2023607.2023668.
18. Iakovidou N., Tiakas E., Tsihlias K. DISCO: A new algorithm for detecting 3D protein structure similarity // *Proceedings of the conference: IFIP International Conference on Artificial Intelligence Applications and Innovations, 2012*. Vol. 382. Springer, Berlin, Heidelberg. P. 622–631. DOI: 10.1007/978-3-642-33412-2\_64.

## Опыт реализации магистерской программы на базовой кафедре

И.И. Левин<sup>1,2</sup>, И.Ю. Кузнецова<sup>1,2</sup>, Б.Е. Механцев<sup>1,2</sup>

<sup>1</sup>Южный федеральный университет  
<sup>2</sup>НИЦ супер-ЭВМ и нейрокомпьютеров

В публикации рассмотрен опыт реализации магистерской программы «Прикладная математика для высокопроизводительных вычислительных систем» на базовой кафедре НИЦ супер-ЭВМ и нейрокомпьютеров и Южного федерального университета. Описаны особенности учебного процесса, его связь с научно-исследовательской и проектной деятельностью студентов для обеспечения высокого уровня подготовки магистров.

*Ключевые слова:* магистратура, организация учебного процесса, практическая подготовка, базовая кафедра.

### 1. Введение

Процессы модернизации высшего образования в России заставляют обратить пристальное внимание на исследование подходов к обучению студентов в магистратуре. Согласно Федеральному закону «Об образовании в Российской Федерации» магистратура является вторым уровнем высшего образования, целью которого является «обеспечение подготовки высококвалифицированных кадров по всем основным направлениям общественно полезной деятельности в соответствии с потребностями общества и государства, удовлетворение потребностей личности в интеллектуальном, культурном и нравственном развитии, углублении и расширении образования, научно-педагогической квалификации» [1]. При этом ни в законе «Об образовании в Российской Федерации», ни в Федеральных образовательных стандартах высшего образования нет требований непрерывности связки «бакалавриат + магистратура», что расширяет спектр возможностей перед поступающими и ставит ряд вызовов перед образовательными организациями и руководителями магистерских программ, отвечающих за качество подготовки специалистов.

В исследованиях последних лет можно видеть следующие индикаторы качества магистерской подготовки [2, 3]:

- уровень профессорско-преподавательского состава;
- высокий уровень подготовки в бакалавриате;
- уровень научных исследований учебного заведения;
- инфраструктура и техническое оснащение учебного заведения.

Одной из тенденций последних лет является изменение среднего возраста студентов магистратуры. Все большее число обучающихся являются сложившимися специалистами, желающими повысить уровень своей квалификации и получить недостающие знания в области, связанной с профессиональной деятельностью. В связи с этим у самих поступающих на магистерские программы возрастают требования к уровню магистерской подготовки и ее практической направленности.

Деятельность базовых кафедр с подготовкой магистрантов по направлению «Прикладная математика для высокопроизводительных вычислительных систем» направления 01.04.02 Прикладная математика и информатика слабо представлена в современной России. Следует обратить внимание на опыт базовой кафедры компьютерного моделирования Московского физико-технического института (МФТИ), г. Жуковский, ведущей подготовку магистрантов по направлениям «Математическая физика и математическое моделирование», «Математические и информационные технологии» и «Системы обработки информации и управления» с базовыми предприятиями Концерн ВКО «Алмаз-Антей», ПАО НПО «Алмаз» и ФГУП «Центральный аэрогидродинамический институт им. профессора Н.Е. Жуковского

(ЦАГИ)». Основные направления деятельности кафедры компьютерного моделирования и структура внешнего взаимодействия описана в [4,5].

Также представляет интерес базовая кафедра № 231 информационных процессов и систем МИРЭА, возглавляемая генеральным директором Международного центра по информатике и электронике (ИнтерЭВМ) президентом ФГАНУ ЦИТиС, д.т.н, профессором, заслуженным деятелем науки и техники Российской Федерации, лауреатом Государственных премий Российской Федерации, премий Правительства Российской Федерации А.В. Старовойтовым [6]. Кафедра ведет подготовку магистрантов по направлению «Программная инженерия» во взаимодействии со многими фирмами через центр ИнтерЭВМ.

Поскольку всякая подготовка специалистов во взаимодействии с базовыми предприятиями подразумевает схожую структуру взаимодействия кафедры и предприятий, задачи решаются схожим путем. Взаимодействие с базовыми предприятиями кафедры осуществляется как минимум через проведение практик и подготовки магистерских диссертаций на базовых предприятиях. Более полный и эффективный вариант подразумевает постоянную работу магистранта на базовом предприятии в режиме совмещения этой учебно-производственной деятельности с исследовательской и учебной.

Основным результатом такой деятельности является довольно форсированное по времени формирование достаточно зрелого специалиста в необходимом для высокотехнологичного производства и осуществления современных прикладных разработок направлении.

## **2. Особенности магистерской программы «Прикладная математика для высокопроизводительных вычислительных систем»**

В 2015 году осуществлен первый набор на магистерскую программу «Прикладная математика для высокопроизводительных вычислительных систем» направления 01.04.02 Прикладная математика и информатика, реализуемую на кафедра интеллектуальных и многопроцессорных систем Института компьютерных технологий и информационной безопасности Южного федерального университета.

Данная кафедра является базовой кафедрой ООО «НИЦ супер-ЭВМ и нейрокомпьютеров» – высокотехнологичного производственного предприятия с 20-летней историей работ в области высокопроизводительной вычислительной техники, одного из крупнейших производителей реконфигурируемых вычислительных систем на основе программируемых логических интегральных схем (ПЛИС) в России. Созданный в 1993 году и прошедший путь от малого предприятия до ведущего поставщика реконфигурируемых вычислительных систем в России, сегодня «НИЦ супер-ЭВМ и нейрокомпьютеров» – динамично развивающийся разработчик и поставщик самых сложных печатных плат не только в России, но и в Европе для сверхвысокопроизводительной вычислительной техники, с производительностью вычислительных блоков, соответствующей первой десятке списка самых мощных компьютеров мира TOP-500.

Уникальные по эффективности решения прикладных задач вычислительные системы «НИЦ супер-ЭВМ и нейрокомпьютеров» обеспечивают многократное преимущество в таких областях науки и промышленности, как моделирование и разработка новых веществ, символьная обработка больших массивов данных, мониторинг глобальных компьютерных сетей и систем связи, моделирование физических процессов и др.

«НИЦ супер-ЭВМ и нейрокомпьютеров» принимает активное участие в обучении студентов вузов. Например, в 2020 году «НИЦ супер-ЭВМ и нейрокомпьютеров» разработал для МФТИ (факультет аэродинамики и летательной техники) курс по программированию реконфигурируемых вычислительных систем.

### **2.1 История кафедры**

Опыт «НИЦ супер-ЭВМ и нейрокомпьютеров» выявил необходимость подготовки высококвалифицированных специалистов, способных строить и анализировать математические модели природных и техногенных систем, таких как процессы сейсморазведки и нефтедобычи,



заниматься моделированием лекарств, морских и воздушных систем, цифровой обработкой сигналов и др. Кроме этого весьма актуально умение реализовывать эффективные численные методы и алгоритмы в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента на суперкомпьютерных вычислительных системах различной архитектуры, в том числе на реконфигурируемых вычислительных системах на базе ПЛИС.

Запрос на специалистов, умеющих работать с реконфигурируемыми вычислительными системами, неуклонно растет, так как за последние годы производительность реконфигурируемых вычислительных систем выросла более чем в 4000 раз. Сейчас реконфигурируемые вычислительные системы применяются при решении широкого класса задач, в том числе в режиме реального времени.

Поэтому в 2015 году кафедра интеллектуальных и многопроцессорных систем Института компьютерных технологий и информационной безопасности Южного федерального университета была преобразована в базовую кафедру «НИЦ супер-ЭВМ и нейрокомпьютеров» с целью подготовки высококвалифицированных специалистов в области прикладной математики и создания программного обеспечения для высокопроизводительных вычислительных систем, в том числе с реконфигурируемой архитектурой.

Первоначально кафедра интеллектуальных и многопроцессорных систем была создана в 2004 году как базовая кафедра Южного научного центра Российской академии наук в Таганрогском государственном радиотехническом университете (ныне – Инженерно-технологическая академия Южного федерального университета) для подготовки аспирантов и докторантов по научной специальности «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей». Заведующим кафедрой в 2004 году был назначен член президиума Южного научного центра РАН, член-корреспондент РАН (ныне – академик РАН), доктор технических наук, профессор, заслуженный деятель науки РФ, лауреат Государственной премии РФ Игорь Анатольевич Каляев. С 2015 года заведывание кафедрой осуществляет доктор технических наук, профессор, лауреат Государственной премии РФ Илья Израилевич Левин.

На кафедре реализуются исключительно программы магистерской подготовки и подготовки кадров высшей квалификации (аспирантура). При этом специфика базовой кафедры определила ряд особенностей учебного процесса:

- дисциплины, изучаемые в рамках образовательных программ, связаны с основными направлениями деятельности «НИЦ супер-ЭВМ и нейрокомпьютеров» и направлены на практическую подготовку обучающихся;
- преподавательский состав кафедры не «включает», как нередко, но состоит из специалистов-практиков (сотрудников «НИЦ супер-ЭВМ и нейрокомпьютеров»), постоянно работающих в направлении, соответствующему преподаваемому предмету;
- ряд практических занятий проводится на оборудовании, предоставленном «НИЦ супер-ЭВМ и нейрокомпьютеров» и относящимся к наиболее современному и эффективному из выпускаемого в мире.

## 2.2 Организация учебного процесса

Основным принципом организации учебного процесса по магистерской программе «Прикладная математика для высокопроизводительных вычислительных систем» является интеграция образовательного, научного и проектно-конструкторского видов деятельности за счет вовлечения в реальные разработки профильных научных и производственных организаций. Такой подход позволяет не только повысить уровень подготовки студента, но и привлекает в магистратуру специалистов, желающих углубить знания в профессиональной области или расширить сферу профессиональных интересов.

Дисциплины образовательной программы направлены на:

- изучение современных методов математического моделирования природных и техногенных систем, таких как процессы сейсморазведки и нефтедобычи, моделирования лекарств, морских и воздушных систем, цифровой обработки сигналов и др.;

- разработку, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий;
- реализацию эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента на суперкомпьютерных вычислительных системах.

Все дисциплины включают в себя значительный практический компонент, подразумевающий практическое использование современных САПР и систем программирования как отечественной, так и зарубежных разработок для решения стоящих перед магистрантами задач.

Практики магистрантов в большей части проходят в «НИЦ супер-ЭВМ и нейрокомпьютеров» в условиях взаимодействия с реальными разработчиками и включают участие в решении реальных технических задач.

Подавляющее большинство практических и лабораторных занятий, а также производственные практики носят проектный характер. Это связано в первую очередь с особенностями учебных курсов и тем, что практические задачи и методики их решения являются по своей сути мини-проектами различной степени сложности и объемности со всеми особенностями именно проектов, а не решения частных задач.

Стоит отметить, что развиваемый Институтом компьютерных технологий и информационной безопасности Южного федерального университета показывает отличные результаты уже при обучении на уровне бакалавриата. Например, под научным руководством сотрудника базовой кафедры интеллектуальных и многопроцессорных систем студенты первого курса смогли провести разработку аппаратно-программного комплекса на ПЛИС, обеспечивающего шифрование и дешифрование и использующего реализованный в виде IP ядра процессор Microblaze для связи с «окружающим миром». Данный подход позволяет повысить качество подготовки студента, привлечь его к научной работе и заинтересовать в дальнейшем обучении и повышении уровня профессиональной подготовки. В дальнейшем планируется расширение этого направления учебной деятельности.

Проектный подход позволяет решить задачу совмещения учебы и адаптации молодых специалистов к научно-производственной деятельности даже с учетом того, что обучение магистрантов занимает лишь два учебных года, причем предыдущее образование нередко включало совершенно другие основные направления учебной деятельности. Среди успешно окончивших магистратуру и защитивших выпускную квалификационную работу на «отлично» есть как выпускники различных кафедр Института компьютерных технологий и информационной безопасности Южного федерального университета, так и выпускники педагогических и политехнических ВУЗов, ранее обучавшихся, например, по направлению технологии машиностроения. Опыт показывает, что выпускник бакалавриата, знающий высшую математику и основы программирования на обычном уровне технического ВУЗа, вполне адаптируется к новым условиям учебной деятельности за достаточно короткое время.

Постоянное взаимодействие между теоретическими и практическими компонентами учебного процесса, высокая межпредметная связность и связь между обычной учебой и производственной практикой дают более высокий уровень фиксации усвоенных умений и компетенций и интеграции их со знаниями. На практике это означает отсутствие во время учебного процесса «таймаутов» между получением и фиксацией знаний. Лекционный материал разбирается и реализуется в лабораторно-практических занятиях до уровня разработки небольшого проекта с его верификацией и таким образом фиксируется как в долговременной памяти, так и в качестве усвоенных умения и навыков. Согласно [7, 8] это можно интерпретировать как то, что если процесс усвоения навыков по отдельным дидактическим единицам можно описать логистической кривой со значительным размахом, то процесс забывания – либо логистической кривой с малым размахом, либо логистической кривой с очень большой постоянной времени. Разумеется, те компоненты, которые оказываются использованными в выпускной квалификационной работе, фиксируются и усваиваются еще эффективнее и на большее время.

Таким образом, магистранты получают уникальную возможность на одной кафедре получить профессиональные компетенции по методам и средствам эффективного программирования большинства существующих архитектур высокопроизводительных

вычислительных систем, практические навыки выполнения научно-исследовательских и опытно-конструкторских задач и стать высококвалифицированными специалистами в области информационных технологий и прикладной математики, способными создавать конкурентоспособную продукцию мирового уровня.

### 2.3 Научная-исследовательская составляющая учебного процесса

Стоит отметить, что все больший процент результатов выпускных квалификационных работ студентов магистерской программы внедрен в деятельность различных организаций, включая «НИЦ супер-ЭВМ и нейрокомпьютеров». Это является показателем не только уровня подготовки студентов, но и качества проводимых ими научно-исследовательских работ.

Магистранты образовательной программы участвуют в реализации проектов мирового уровня:

- создание реконфигурируемых вычислительных систем нового поколения с иммерсионной (погружной) системой охлаждения, удельная производительность которых превышает мировой уровень;
- создание оригинальных материнских плат и элементов операционных систем (BIOS, драйверы, системы мониторинга и др.);
- автоматический распараллеливающий компилятор с языка C для реконфигурируемых вычислительных систем;
- новый язык архитектурно-независимого программирования Set@1;
- новые методы и программные средства для решения задач цифровой обработки сигналов и изображений;
- разработка прикладных программ различных предметных областей для РВС;
- построение и исследование математических моделей сложных технических и природных систем.

Научно-исследовательскую работу магистранты проводят под непосредственным руководством ведущих специалистов организации. Результаты научной работы студенты представляют на научных семинарах, проводимых на кафедре и в «НИЦ супер-ЭВМ и нейрокомпьютеров», и конференциях различного уровня.

Такой подход к организации научной работы обучающихся позволяет не только закрепить теоретические знания, но и получить навыки командной работы и реализации проектов в области создания передовых высокопроизводительных систем и программного обеспечения для них.

### 2.4 Материально-техническое обеспечение учебного процесса

При реализации образовательных программ в сфере суперкомпьютерных технологий, критически важно наличие современной технической базы. Студентам магистерской программы «Прикладная математика для высокопроизводительных вычислительных систем» предоставлен доступ к передовым высокопроизводительным вычислительным системам и их программным комплексам для решения вычислительно-трудоемких задач различных областей. При обучении и проведении научно-исследовательских работ магистранты работают на вычислительных системах с общей и распределенной памятью, а также на реконфигурируемой вычислительной системе РВС-7 «Плеяда» производительностью 62 Тфлопс и ряде вычислительных модулей производства «НИЦ супер-ЭВМ и нейрокомпьютеров» на основе программируемых логических интегральных схем семейств Xilinx Virtex-6, Virtex-7, Virtex UltraScale.

В случае прохождения производственной практики в «НИЦ супер-ЭВМ и нейрокомпьютеров» магистрантам предоставляется доступ к современным программным средствам и наиболее перспективным суперкомпьютерам, производимым организацией, например, к системам с жидкостным охлаждением. Также благодаря многолетней работе «НИЦ супер-ЭВМ и нейрокомпьютеров» с компанией Xilinx, с 2018 года магистранты кафедры имеют доступ к программному обеспечению, разрабатываемому данной компанией.

### 3. Заключение

Опыт реализации магистерской программы «Прикладная математика для высокопроизводительных вычислительных систем» показывает хорошие результаты совместных усилий индустрии и образовательной организации для повышения уровня подготовки специалистов. С 2017 г. по 2020 г. 54 студента получили дипломы магистра в рамках данной образовательной программы. В последние два года все выпускники программы имели предложения от предприятий-партнеров (главным образом от «НИЦ супер-ЭВМ и нейрокомпьютеров») по трудоустройству по специальности. Общий уровень трудоустройства по специальности выпускников программы составляет примерно 80 %. Интеграция учебного и научного процессов с проектной деятельностью при своевременном обновлении материально-технического обеспечения способствует повышению качества образования и сохранению заинтересованности студента в обучении в течение всего процесса. Но при этом стоит отметить, что необходимым условием подготовки высококвалифицированного специалиста в магистратуре является наличие у поступающего базовых знаний, заинтересованность в обучении и понимание перспектив по окончании обучения.

### Литература

1. Федеральный закон № 273-ФЗ «Об образовании в Российской Федерации» от 29.12.2012 (ред. от 30.04.2021) URL: <http://www.kremlin.ru/acts/bank/36698> (дата обращения: 10.05.2021).
2. Мухаметзянова Ф.Г., Хайрутдинов Р.Р., Сигачёва Н.А. Современные подходы к моделированию магистерского уровня подготовки студентов // Человек и образование. 2017. №2 (51). С. 77–81.
3. Чистова Я.С. Динамическое моделирование системы подготовки магистров профессионального обучения: автореф. дис. канд. пед. наук. Екатеринбург: РГАУ – МСХА имени К.А. Тимирязева, 2016. 26 с.
4. Кафедра компьютерного моделирования Московского государственного технического института (национального исследовательского университета), г. Жуковский. URL: <https://mipt.ru/education/chairs/km/> (дата обращения: 10.05.2021).
5. Базовая кафедра компьютерного моделирования Московского государственного технического института (национального исследовательского университета), г. Жуковский. URL: <https://mipt.ru/dafe/abiturientam/tselevoynabor.php> (дата обращения: 10.05.2021).
6. Базовая кафедра № 231 – информационных процессов и систем МИРЭА - Российского технологического университета URL: <https://www.mirea.ru/education/the-institutes-and-faculties/institute-of-information-technology/the-structure-of-the-institute/base-department-231-information-processes-and-systems/> (дата обращения: 10.05.2021).
7. Ritter F.E., Schooler L.J. International Encyclopedia of the Social and Behavioral Sciences, chapter The learning curve. Amsterdam: Pergamon, 2002. P. 8602–8605.
8. Салливан Б., Томпсон Х. Эффект плато. Как преодолеть застой и двигаться дальше / Боб Салливан и Хью Томпсон; пер. с англ. Павла Миронова. Москва: Манн, Иванов и Фербер, 2014. 320 с.

# Применение графических ускорителей для поиска оптимального совмещения пар жестких трехмерных структур методом Кабша<sup>1</sup>

И.А. Тимохин<sup>1</sup>, Н.Н. Попова<sup>1</sup>, Д.А. Суплатов<sup>2</sup>

<sup>1</sup>Факультет вычислительной математики и кибернетики, <sup>2</sup>Научно-исследовательский институт физико-химической биологии имени А.Н.Белозерского

В статье рассматривается применение графических ускорителей для поиска оптимально совмещения пар жестких структур методом Кабша. Метод часто используется при решении многих прикладных задач. В статье предлагается реализация метода Кабша для графических ускорителей. Перенос вычислений на графические ускорители выполняется для построения большого количества оптимальных трехмерных совмещений пар структур. Разработанная реализация метода позволяет достичь значительного ускорения по сравнению с реализацией на центральном процессоре.

*Ключевые слова:* Метод Кабша, графический процессор, CUDA, совмещение жестких структур

## 1. Введение

При решении задач биоинформатики, компьютерной химии, распознавания образов часто возникает необходимость поиска большого количества оптимальных совмещений пар жестких структур. Под поиском оптимального совмещения предполагается поиск преобразования смещения и вращения двух последовательностей точек, описывающих выравниваемые объекты, минимизирующего их среднеквадратичное отклонение. Одним из способов решения данной задачи является метод Кабша [1, 2].

Так, например, в алгоритме множественного структурного выравнивания белков МАТТ [3] количество применений метода Кабша может быть грубо оценено как  $O(m^2l^2)$ , где  $l$  — длина сравниваемых структур, а  $m$  — их количество. Таким образом, при выравнивании 50 структур со средней длиной в 350 элементов число требуемых совмещений превысит  $3 * 10^8$ . Другим примером задачи, при решении которой возникает большое количество совмещений, является задача поиска белка, схожего с данным, в базе данных. Для этого требуется выполнить парные выравнивания искомой структуры с соединениями, содержащимися в базе, что также ведет к большому количеству применений алгоритма Кабша.

В данной работе предлагается метод организации вычислений для выполнения множества выравниваний трехмерных структур алгоритмом Кабша на графических ускорителях. Мотивацией такого рассмотрения является идея сокращения времени выполнения вычислений.

## 2 Описание метода Кабша

Рассматриваемый метод был описан Вольфгангом Кабшем, в честь которого и был назван, в 1976 году в статье "A solution for the best rotation to relate two sets of vectors" [1, 2]. В данном разделе будет описана постановка задачи, решаемой данным методом, а также представлен алгоритм, реализующий метод. Стоит отметить, что алгоритм может быть использован для размещения наборов точек любой размерности, но в данной работе рассматривается его

---

<sup>1</sup> Работа выполнена при поддержке грантов РФФИ № 20-07-00970 и №20-07-01053 с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова [7].

применение по отношению к трехмерным структурам, поэтому первая часть описания, посвященная рассмотрению самого метода, будет представлена в общих терминах. Во второй части будет рассмотрен метод, используемый для нахождения сингулярного разложения, уже будет учитывать специфику малой размерности искомой матрицы.

Рассмотрим постановку задачи метода Кабша. Пусть даны два набора точек:

$$P = \{p_1, p_2, \dots, p_k\}, p_i \in R^n, i = 1 \dots k \quad (1),$$

$$Q = \{q_1, q_2, \dots, q_k\}, q_i \in R^n, i = 1 \dots k \quad (2)$$

по  $k$   $n$ -мерных точек в каждом. Необходимо найти матрицу вращения  $M \in R^{n \times n}$  и вектор  $D \in R^n$ , преобразующих набор точек  $Q$  в набор точек

$$Z = \{Mq_1 - D, Mq_2 - D, \dots, Mq_k - D\}, z_i \in R^n, i = 1 \dots k \quad (3),$$

и таких, что при этих  $M$  и  $D$  достигается минимум среднеквадратичного отклонения набора точек  $P$  и  $Z$ , то есть достигается

$$\min \text{RMSD}(P, Z) = \sqrt{\frac{1}{k} \sum_{i=1}^k |p_i - z_i|^2} \quad (4).$$

Обозначим геометрические центры наборов точек  $P$  и  $Q$  как

$$C_p = \frac{1}{k} \sum_{i=1}^k p_i, C_p \in R^n \quad (5) \text{ и}$$

$$C_q = \frac{1}{k} \sum_{i=1}^k q_i, C_q \in R^n \quad (6)$$

соответственно. Тогда, если сделать допущение, что оптимальным вектором смещения является вектор, совмещающий геометрические центры данных наборов точек

$$D = C_q - C_p \quad (7),$$

то алгоритм Кабша позволяет найти оптимальную матрицу поворота  $M$ .

Алгоритм работает в три этапа: смещение, построение кросс-ковариационной матрицы и вычисление матрицы поворота. На первом этапе точки из данных наборов смещаются таким образом, чтобы их геометрические центры совпадали с началом координат:

$$P^0 = \{p_1 - C_p, p_2 - C_p, \dots, p_k - C_p\}, P^0 \in R^n \quad (8),$$

$$Q^0 = \{q_1 - C_q, q_2 - C_q, \dots, q_k - C_q\}, Q^0 \in R^n \quad (9).$$

На втором этапе строится кросс-ковариационная матрица для набора точек  $P^0$  и  $Q^0$ :

$$H = \sum_{i=1}^k P_i^0 (Q_i^0)^T \quad (10).$$

На третьем этапе вычисляется матрица поворота  $R$  на основе матрицы  $H$ . В работе В. Кабша было показано, что матрица поворота может быть выражена через матрицу кросс-ковариации следующим образом:

$$M = (H^T H)^{\frac{1}{2}} H^{-1} \quad (11).$$

Также в работе было показано, что матрица поворота может быть вычислена на основе сингулярного разложения матрицы  $H$ :

$$H = U \Sigma V^T, U, \Sigma, V \in R^{n \times n}, \quad (12),$$

$$d = \text{sign}(\det(VU^T)) \quad (13),$$

$$M = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} U^T \quad (14),$$

для чего сначала вычисляется непосредственно сингулярное разложение кросс-ковариационной матрицы, затем вычисляется  $d$ , показывающее необходимо ли совершать корректировку для сохранения правосторонней системы координат, и вычисляется матрица поворота  $M$ .

Метод, основанный на сингулярном разложении, более универсален, так как лишен недостатков, связанных с обработкой особых случаев при вырожденной матрице  $H$ , поэтому будем рассматривать именно этот подход. Задача поиска сингулярного разложения сводится к задаче поиска собственных чисел и векторов матрицы  $H$ . Так как в работе рассматриваются трехмерные структуры, то есть  $n = 3$ , то и матрица, для которой ищутся собственные числа и векторы, будет иметь размерность  $3 \times 3$ . Одним из наиболее эффективных методов решения данной задачи для матриц малой размерности является QR-алгоритм [4, 5, 6], основанный на QR разложении рассматриваемой матрицы. Кратко поясним суть данного метода.

Основой QR алгоритма является одноименное разложение матрицы – любая квадратная матрица  $A \in R^{n \times n}$  может быть разложена на произведение унитарной матрицы  $Q \in R^{n \times n}$  и верхнетреугольной матрицы R:

$$A = QR(15).$$

Сам же алгоритм является итерационным. В качестве начального приближения берется сама матрица

$$A_0 = A(16).$$

На каждой итерации  $k$  вычисляется QR разложение матрицы  $A_k$ , на основе которого вычисляется следующее приближение :

$$A_k = Q_k R_k, R_k, Q_k \in R^{n \times n}(17),$$

$$A_{k+1} = R_k Q_k = Q_k^{-1} Q_k R_k Q_k = Q_k^{-1} A_k Q_k = Q_k^T A_k Q_k(18).$$

В качестве критерия останова используется порог  $th$  на сумму модулей значений матрицы  $A_k$ , находящихся ниже главной диагонали:

$$\sum_{i,j=1..n, i < j} |(a_k)_{i,j}| < th(19).$$

В результате работы алгоритма получаем следующее преобразование:

$$\lambda \approx A_k = (Q_k^T Q_{k-1}^T \dots Q_0^T) A (Q_0 \dots Q_{k-1}^T Q_k^T) = (Q_0 \dots Q_{k-1} Q_k)^T A (Q_0 \dots Q_{k-1} Q_k)(20),$$

$$S = Q_0 Q_1 \dots Q_k \Rightarrow \lambda \approx A_k = S^T A S(21).$$

В результате находим собственные числа A:  $\lambda_1^2 \geq \lambda_2^2 \geq \dots \geq \lambda_n^2$  (при необходимости столбцы матрицы S переставляются, для достижения нужного порядка собственных значений), столбцы матрицы S нормализуются, чтобы получить нормированные собственные вектора. Далее вычисляются уже непосредственно элементы сингулярного разложения матрицы H:

$$\Sigma = diag(\lambda_1, \lambda_2, \dots, \lambda_n), U = S, V = AU(22),$$

исходя из которых, уже ищется сама матрица поворота M (14).

### 3 Описание предложенного метода переноса вычислений на GPU

В данном разделе будет описан предлагаемый метод реализации вычислений алгоритма Кабша на графическом ускорителе. Напомним, что в данной статье рассматривается случай, когда необходимо произвести большое количество выравниваний трехмерных структур методом Кабша — сотни тысяч и миллионы.

В качестве технологии для реализации вычислений на GPU была выбрана CUDA, поэтому для удобства данный раздел будет описан в терминах, используемых данной технологией.

Как видно из описания алгоритма, представленного в предыдущем разделе, само по себе одно вычисление трехмерной матрицы поворота методом Кабша является достаточно маленькой независимой задачей с крайне малым внутренним параллелизмом. Такая задача может быть выполнена одним cuda-поток. Таким образом, на графическом ускорителе одновременно будет выполняться большое количество вычислений матриц поворота методом Кабша. Рассматриваемую задачу можно разделить на две подзадачи: вычисление кросс-ковариационной матрицы (5 – 10) и вычисление ее сингулярного разложения (12 – 22) с последующим вычислением матрицы поворота. Предлагается каждую из этих подзадач вынести в собственное cuda-ядро. В дальнейшем эти подзадачи будем называть kernel\_1 и kernel\_2 соответственно. Таким образом, формирование матриц поворота для некоторого набора задач состоит из следующих шагов: копирование необходимых данных на графический ускоритель (copy\_to), работа cuda-ядер kernel\_1 и kernel\_2, копирование построенных матриц с графического ускорителя (copy\_from).

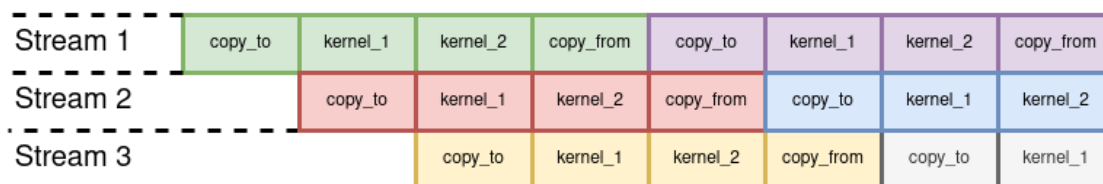


Рис. 1. Наглядное изображение работы конвейера вычислений на графическом ускорителе при использовании трех cuda-stream.

Для более оптимальной загрузки графического ускорителя предлагается не проводить все вычисления разом, а разделить их на блоки фиксированного размера (например, по 2048 вычислений за раз) и использовать cuda-stream для организации конвейера вычислений. Для предложенного разделения задачи на 2 ядра, оптимальная ширина конвейера будет равна 3. На рис. 1 наглядно изображен участок конвейера вычислений, получаемый при использовании предлагаемого метода. Для удобства каждый блок вычислений окрашен в свой цвет. Данный подход позволит одновременно отправлять и получать данные, а также скрывает обмены данными за вычислениями. Вместе с тем, одновременно будут считаться 2 cuda-ядра, что позволит более эффективно нагружать графический процессор.

При обработке одного блока входных данных в силу того, что каждая матрица поворота вычисляется одновременно с остальными, зачастую в один момент времени разные cuda-потоки будут обращаться к элементам с одинаковым индексом в матрицах и векторах, используемых в ходе вычислений. Поэтому для организации оптимального доступа к памяти графического ускорителя предлагается перейти от подхода AoS (array of structures) к SoA (structure of arrays) при хранении на GPU используемых данных. Массив векторов будет храниться в линейном массиве таким образом, что сначала хранятся первые элементы всех векторов, затем вторые и третьи. Аналогично и для матриц, сначала хранятся элементы первой строки первого столбца каждой матрицы, потом - первой строки второго столбца, и так далее. Для наглядности метод перехода от AoS к SoA продемонстрирован на рис. 2 и рис. 3 для векторов и матриц соответственно. Для удобства на изображениях элементы, принадлежащие к одной структуре, окрашены в один цвет.



Рис. 2. Переход от AoS к SoA для векторов.

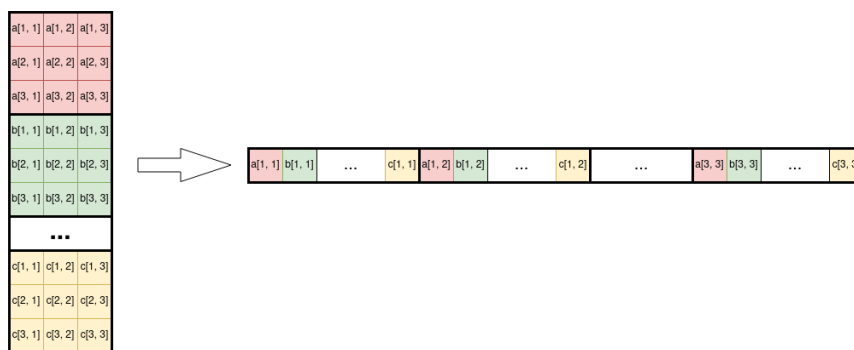


Рис. 3. Переход от AoS к SoA для матриц.

#### 4 Анализ эффективности предложенного метода

Для оценки эффективности предложенного метода был проведен вычислительный эксперимент на платформе IBM Power8 для сравнения времени работы последовательного алгоритма на CPU на одном и 10 потоках с временем работы реализации предложенного метода на GPU. В данном разделе приводится методика проведения вычислительного эксперимента, дается описание вычислительной платформы, на которой проводился эксперимент и обсуждаются полученные результаты.

Для определения эффективности предложенного метода была взята приближенная к реальной постановка задачи. Имеется две последовательности точек в пространстве. Назовем блоком длины  $l$  последовательность из  $l$  подряд идущих точек, принадлежащих одному набору. Назовем парой блоков длины  $l$  совокупность двух блоков равной длины из разных



последовательностей. Необходимо найти совмещающую матрицу поворота методом Кабша для всех пар блоков длины от 5 до 10, содержащихся в данной паре последовательностей.

Эксперимент проводился на многопроцессорном кластере Polus, состоящем из 4-х вычислительных узлов. Основные характеристики вычислительного узла:

- 2 процессора IBM POWER 8 (10 ядер у каждого)
- 2 графических ускорителя Nvidia Tesla P100 (16 Gb, NVLink)

Для проведения эксперимента использовался один процессор и один графический ускоритель.

Для проведения эксперимента были разработаны следующие варианты реализации алгоритма Кабша на языке C++: (1) последовательный вариант для оценки времени счета на центральном процессоре; (2) OpenMP вариант, в котором внешний цикл, в котором непосредственно применяется метод для поиска совмещающей матрицы, выполняется параллельно с использованием 10 потоков; (3) реализация предложенного метода с использованием технологии CUDA для оценки времени работы на графическом ускорителе компании Nvidia.

Для компиляции кода для центрального процессора использовался компилятор IBM xlc++\_r с третьим уровнем оптимизации. Для компиляции кода для графического ускорителя использовался компилятор nvcc с третьим уровнем оптимизации и целевой архитектурой sm\_60.

**Таб. 1.** Времена счета в секундах различных реализаций при выполнении указанного числа совмещений методом Кабша.

Кол-во применений алгоритма Кабша	Время sru, сек.	Время sru+omp, сек.	Время cuda, сек.
10000	0,0074	0,0090	0,0005
50000	0,0370	0,0053	0,0015
100000	0,0741	0,0106	0,0025
500000	0,3667	0,0526	0,0139
1000000	0,7366	0,1049	0,0240
5000000	3,4322	0,3920	0,1355
10000000	6,3618	0,7687	0,2409

**Таб. 2.** Ускорении CPU+OpenMP и CUDA реализаций по отношению к последовательной реализации.

Кол-во применений алгоритма Кабша	Ускорение sru+omp	Ускорение cuda
10000	0,82	15,28
50000	6,99	24,24
100000	7,00	29,24
500000	6,97	26,43
1000000	7,02	30,68
5000000	8,76	25,33
10000000	8,28	26,41

### Времена счета различных реализаций

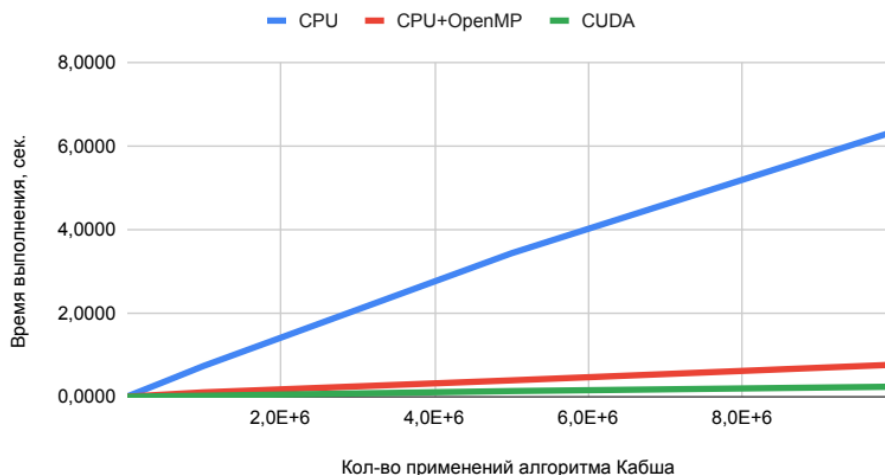


Рис. 4. Время счета различных реализаций в зависимости от количества выполняемых совмещений методом Кабша.

### Достижимые ускорения

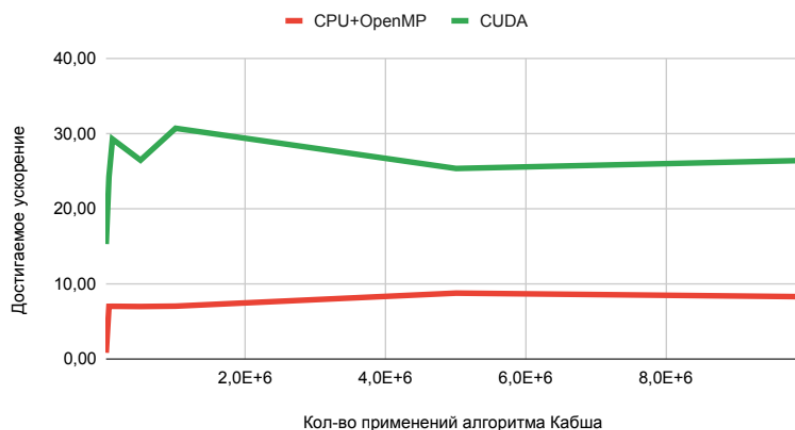


Рис. 5. Достижимое параллельными реализациями ускорение по отношению к последовательной версии.

В таблице таб. 1 представлены времена работы, затраченные для совершения указанного числа применения алгоритма Кабша для описанных выше реализаций. В таблице таб. 2 для удобства представлены значения достигаемых ускорений для CPU реализации с использованием OpenMP и для CUDA реализации. На рис. 4 и рис. 5 приведены графики, наглядно иллюстрирующие содержание таб. 1 и таб. 2 соответственно.

Как видно из представленных данных, предложенная реализация вычислений большого количества выравниваний методом Кабша, позволила достичь ускорения в 25-30 раз по сравнению с последовательной версией алгоритма и более чем в два раза по сравнению с OpenMP реализацией. Полученные результаты позволяют прогнозировать значительное сокращение времени выполнения таких алгоритмов, как, например, МАТТ [3], или поиск схожих структур по базе данных при использовании предложенной реализации алгоритма Кабша на GPU. Дальнейшие работы будут направлены на исследование эффективности применения разработанной реализации в других задачах.

## 5 Благодарности

Работа выполнена при финансовой поддержке гранта РФФИ № 20-07-00970 (Суплатов Д.А., Тимохин И.) и № 20-07-01053 (Попова Н.Н.)

## Литература

1. Kabsch, W. (1976), A solution for the best rotation to relate two sets of vectors. *Acta Cryst. A*, 32: 922-923. <https://doi.org/10.1107/S0567739476001873>.
2. Kabsch, W. (1978), A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Cryst. A*, 34: 827-828. <https://doi.org/10.1107/S0567739478001680>.
3. Menke M, Berger B, Cowen L (2008) Matt: Local Flexibility Aids Protein Multiple Structure Alignment. *PLOS Computational Biology* 4(1): e10. <https://doi.org/10.1371/journal.pcbi.0040010>
4. J. G. F. Francis, The QR Transformation A Unitary Analogue to the LR Transformation—Part 1, *The Computer Journal*, Volume 4, Issue 3, 1961, Pages 265–271, <https://doi.org/10.1093/comjnl/4.3.265>.
5. J. G. F. Francis, The QR Transformation—Part 2, *The Computer Journal*, Volume 4, Issue 4, 1962, Pages 332–345, <https://doi.org/10.1093/comjnl/4.4.332>.
6. V.N. Kublanovskaya, On some algorithms for the solution of the complete eigenvalue problem, *USSR Computational Mathematics and Mathematical Physics*, Volume 1, Issue 3, 1962, Pages 637-657, ISSN 0041-5553, [https://doi.org/10.1016/0041-5553\(63\)90168-X](https://doi.org/10.1016/0041-5553(63)90168-X)
7. Воеводин Вл.В., Жуматий С.А., Соболев С.И., Антонов А.С., Брызгалов П.А., Никитенко Д.А., Стефанов К., Воеводин В.В. Практика суперкомпьютера «Ломоносов» // Открытые системы. – 2012. – Т. 7 – С. 36–39



# **Аннотации стендовых докладов**

## Builder-like Design Pattern for effective In-Situ Processing

Boris Morose, Alex Margolin

Huawei Tel-Aviv Research Center

With expanding gap between computation power and IO ability in most HPC systems, the importance of the In-Situ computations constantly increases. In a traditional approach, the simulation produces some output that is processed post hoc. It becomes problematic when a huge amount of data needs to be saved for a time-consuming application. In-Situ processing asks to perform all needed computation during simulation itself. The main idea of In-Situ analysis and visualization is to delegate all needed data to dedicated cores for further processing [1]. The benefit of this technique is a reduced volume of output data that need to be saved, while the increased number of cores and an additional network traffic became its downside.

In current research we consider applications that use spatial partition to achieve parallelization. The simulations organize an iterative procedure to investigate an advance along a time. In concurrent runs, each process is responsible for specific region and performs all needed computations for every timestep of the simulation. The processes communicate to share the data needed for next timestep. For example, in Molecular Dynamic simulation, each process usually has its own and ghost atoms. The later actually belong to the neighbor processes. To perform simulation of its own atoms, the process has to receive the information of ghost atoms. The late is achieved by inter-process communications for each timestep. Widely recognized that these communications produce a bottleneck and may cause a poor scalability. Our numerical experiments on nine node cluster shows that for some configurations, the communication time takes significant part of the total simulation time – 50% and more.

In this work we propose to utilize the communication delay to perform calculations needed for analysis and visualization. We expect the significant reduce of the total time of simulation and In-Situ processing.

The proposed parallel solution is generalized and is formulated as a Design Pattern similar to the well-known sequential analog Builder [2]. This pattern can be implemented in wide range of different type of simulations.

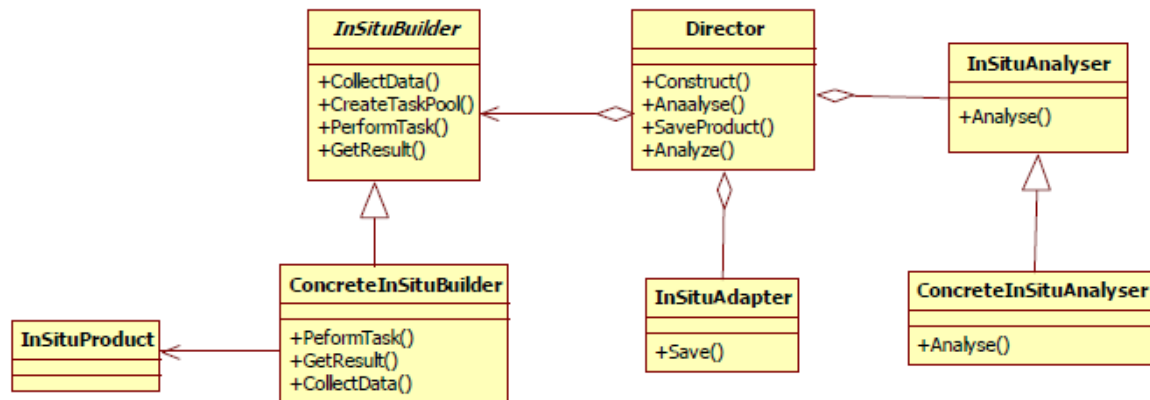


Fig. 1. Class Diagram of In-Situ Builder Design Pattern

```
initialize simulation
bridge::initialize

do
    if timestep for in-situ analysis
        initialize InSituBuilder
        collect data
        start in-situ processing

        if during in-situ processing
            if waiting for MPI communication
                perform in-situ task
            if all in-situ tasks are finished
                bridge::execute
                finalize InSituBuilder

        compute new state
    while !done

bridge::finalize
finalize simulation
```

**Fig. 2.** Pseudo-code of In-Situ Builder Design Pattern incorporated into Sensei simulation loop [1]

The proposed Design Pattern is demonstrated using LAMMPS - Molecular Dynamics Simulator [3]. As a reference, billion-atom Atomic Fluid with Lennard-Jones potential (lj) and Metallic Solid (eam) scalable benchmarks were used.

## References

1. Hank Childs, Cyrus Harrison, et.al. In Situ Analysis and Visualization with SENSEI and Ascent, SC19 - The International Conference for High Performance Computing, Networking, storage, and Analysis, 2019. URL: <https://sc19.supercomputing.org/presentation/?id=tut141&sess=sess199> (accessed: 10.04.2021).
2. Gamma, E, Helm, R, et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1994, 416 p. DOI: 10.1007/978-3-319-02192-8\_16
3. LAMMPS - Molecular Dynamics Simulator. URL: <https://lammps.sandia.gov/> (accessed: 10.04.2021)

# First Results of Performance Evaluation of Geospatial Raster Data Processing Systems

K.V. Bykov, R.A. Rodrigues Zalipynis

HSE University

The area of this research is processing and analysis of geospatial raster data, or Earth remote sensing data. This type of data is an important asset in numerous domains. For example, it facilitates analyzing forests' composition, soil types of agricultural fields, understanding water content, and others [1, 3, 4]. Geospatial raster data (which is also referred to as raster geodata) comes from Earth satellites, unmanned aerial vehicles, numerical simulations, and other sources.

Dozens of possible processing operations are available for manipulating geospatial raster data. Map algebra is one of the most popular sets of such operations. In this work we focus on pixel-wise operations. We use real-world data: multi-spectral satellite scenes. We tested algebraic operations with different arities. Vegetation indices are vivid representatives of pixel-wise map algebra operations [2]. Dozens of vegetation indexes are used for vital tasks in numerous practical fields. We ran the computing of vegetation indexes by geodata processing systems. They behave very differently and it is not obvious at a glance which system should be utilized for which purposes. The insights from our study will help to make the correct choice of a system. Such a choice should significantly influence the productivity of geodata users.

Petabytes of Earth remote sensing raster geodata are freely available from Earth remote sensing satellites. In this work, we used the data from the Landsat 8 NASA satellite [9]. It acquires 11 spectral bands in different wavelengths [9]. In this work we used 6 bands since others are not required as inputs for the vegetation indices under consideration.

A large amount of data comes from Earth remote sensing satellites daily [5, 6]. This makes the processing of raster geodata a real big data problem. There is a large number of software dedicated to manipulating the geodata [7]. Hence, the problem of making the proper choice of a tool for processing is important. Therefore, it is necessary to have up-to-date knowledge about the systems. We explored 3 distributed raster data processing systems: GeoTrellis (based on Apache Spark) [8], RasterFrames (also based on Apache Spark) [10], and SciDB (array DBMS) [11]. We collected the measurements on several important indicators: CPU utilization, RAM utilization, and time needed to complete a calculation. We did this for different scenarios to better understand when a system performs better than other systems.

Experimental setup. We used 4 virtual machines in Microsoft Azure Cloud with Ubuntu 18.04-LTS, 8 GB RAM, 30 GB HDD, and 2 virtual CPUs. This setup is sufficient to obtain the proper measurements and demonstrate the differences between the systems. All the virtual machines were prepared for the calculations by installing and configuring the systems. GeoTrellis [8] and RasterFrames [10] require Apache Spark installed and SciDB [11] requires SciDB workers installed. VMs use a shared network and were configured as a computer cluster. We developed the code for each system to compute the vegetation indices. The results are in fig. 1.

GeoTrellis demonstrates good performance and finishes calculations in approximately 5-6 minutes. Calculation time decreases when the number of nodes increases in the computer cluster. A large number of map algebra operations are supported by the system. RasterFrames has the best performance among the systems under investigation and finishes calculations in 2-3 minutes: twice faster than GeoTrellis. As a drawback, only a limited number of map algebra operations are supported by RasterFrames to date. The operation of raster join demonstrates the worst result and takes over 15 minutes. This is unacceptable result that limits the applicability of the system. SciDB takes 1.5 hours to ingest the data. This makes it unusable for quick calculations. However, SciDB supports basic map algebra operations. Calculations take place on SciDB arrays and most of the operations are implemented in C/C++.

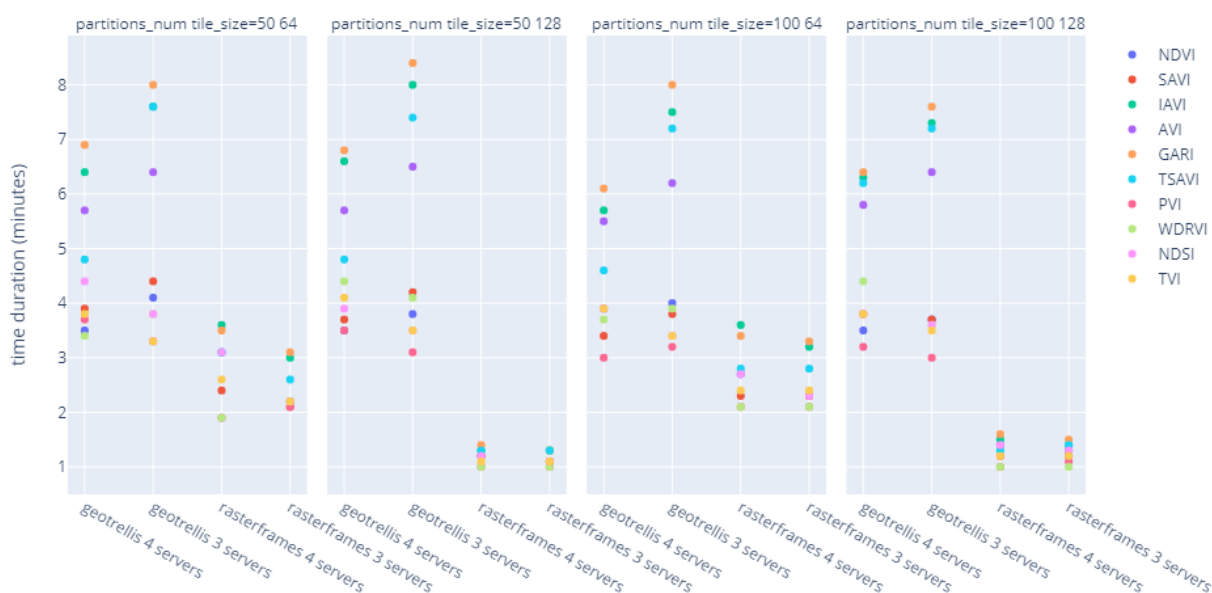


Figure 1. The Results of the Experiments

## References

1. R.A. Rodrigues Zalipynis. “Ecologic assessment of air pollution by nitrogen dioxide over the territory of Europe using Earth remote sensing data.” In: *Inform. Cybern. Comput. Eng.* 1.19 (2014), pp. 126–130.
2. X. Jinru and B. Su. “Significant Remote Sensing Vegetation Indices: A Review of Developments and Applications”. In: *J. of Sensors* (2017), pp. 1–17. DOI: 10.1155/2017/1353691.
3. R.A. Rodrigues Zalipynis. “Array DBMS in environmental science: satellite sea surface height data in the Cloud”. In: *IDAACS. IEEE*, 2017, pp. 1062–1065.
4. R.A. Rodrigues Zalipynis et al. “Retrospective Satellite Data in the Cloud: An Array DBMS Approach”. In: *CCIS. Vol. 793. Springer*, 2017, pp. 351–362.
5. R.A. Rodrigues Zalipynis et al. “Array DBMS and satellite imagery: towards big raster data in the Cloud”. In: *LNCS. Vol. 10716. Springer*, 2018, pp. 267–279.
6. R.A. Rodrigues Zalipynis. “Distributed In Situ Processing of Big Raster Data in the Cloud”. In: *LNCS. Vol. 10742. Springer*, 2018, pp. 337–351.
7. R.A. Rodrigues Zalipynis. “Generic distributed in situ aggregation for Earth remote sensing imagery”. In: *LNCS. Vol. 11179. Springer*, 2018, pp. 331–342.
8. *GeoTrellis Documentation*. URL: <https://geotrellis.io/documentation>.
9. *Landsat 8 Overview*. URL: <https://landsat.gsfc.nasa.gov/landsat-8/>.
10. *RasterFrames*. URL: <https://rasterframes.io/index.html>.
11. *SciDB Documentation 20.10*. URL: <https://paradigm4.atlassian.net/wiki/spaces/scidb/overview>.



# Maximizing the loop tiling ability via static analysis

Al. Levchenko

SPbPU Supercomputer Center

Spatial locality is an essential factor in the performance of loop-intensive scientific algorithms. Loop tiling technique can target spatial locality using the automatic polyhedral optimization of the low-level intermediate representation via LLVM/Polly [1]. A crucial prerequisite of this paper is the presumption of the polyhedral compilation effectiveness, assuming that the fully-polyhedral approach can be highly profitable per se comparing with purely syntactic or even hybrid approaches [2]. In a polyhedral framework, a composition of loop transformations can be implemented through modifications of schedule trees, a structured schedule representation which was proposed in a pioneer work [3]. The fine point of this procedure is that only valid static control parts (SCoPs) can be translated from LLVM-IR to a polyhedral description, represented in the form of schedule trees and transformed advantageously. Moreover, current algorithms for optimizing locality in loop nests are constrained by semantic preservation, and, hence, overapproximation of such rigorous constraints is not straightforward. As a result, LLVM/Polly rejects loop nests with invalid SCoPs, which were incorrectly formulated by humans and passed along the compilation pipeline [4]. Therefore, submission to the loop optimizer the input code that meets the valid SCoP criteria remains a significant challenge that should be mitigated in advance at the early level. Another aspect is building formal static diagnostic rules for the correct application of Clang loop tiling pragmas [5], providing the fine-tuning of tile size [6].

This paper presents an LLVM/Clang-based tool, which introduces a strict formalism to the static analysis of loop nests where tiling is beneficial for performance. The first contribution of the presented approach includes a static analysis, which adheres to the strict syntactic description of SCoP proposed in the seminal work [1]. This analysis pursues the goal of reducing quasi-SCoPs to the canonical syntactic description by correcting errors that lead to the SCoP rejection by the polyhedral loop optimizer without further transformations into the form of schedule trees. The set of rejection reasons was inspired by the statistical results demonstrated in the original work [4] with that difference that invalid SCoPs are eliminated now at the earliest stage of static analysis, being the starting point for using pragma directives for loop tiling transformation. In particular, the implied syntactic description of SCoPs admits the presence, mainly, affine lower and upper bounds of the loop nest, affine expressions used for memory access, the absence of complex structures in CFG, the absence of non-canonical induction variables. Thus, once the resulting potential SCoP is detected, it is considered for placement pragma directive for loop tiling and, finally, marked as valid for translation to a polyhedral description.

The second contribution of the proposed approach concerns the extraction of several tiled loop parameters. These parameters are applied in the Tile Size Selection (TSS) algorithm for rectangular loop tiling, which is considered a starting point for scaling the tiled loop nests with the optimal locality. The range of such parameters that can be co-obtained by means of static analysis includes the computation volume of a tile, the memory footprint of the tile, the intra-tile loop overhead. These software parameters form the basis of the analytical model for the total execution time of the tiled loop, which allows the formulation of the objective function for the case of multi-level tiling. On the other hand, hardware parameters are obtained separately and, thus, an analytical cost model considers several non-trivial parameters of non-uniformity. As a result, the multi-level optimal tile size selection problem can be formulated and solved as an integer geometric program. An essential aspect of the TSS stage is that the obtained estimates of optimal tile sizes represent a unique performance footprint of the symbiosis of a particular computational kernel and a specific target architecture. As a result, the static analysis increases coverage of tiled loop nests, more strictly determining the level of tiling with optimal size.

Finally, the third contribution of the proposed approach gives insight into the placement of the pragma directive for loop tiling guided by the first stage, static analysis. Simultaneously the TSS results derived from the second stage can be explicitly defined via a clause specific to a pragma directive for loop tiling. The resulting SCoPs are translated from LLVM-IR into polyhedral representation, or, more precisely, into the form of schedule trees [3]. In this context, the primary node of interest in the schedule tree is the band node, which includes a partial schedule, i.e., a tuple of piece-wise quasi-affine expressions, to define loop statements following the lexicographic order. The tileable band nodes can be tiled by inserting the new partial schedule for loop tiling, which gives rise to new band nodes. Since, in practice, the band node can be tiled either fully automatically or even conversely manually, the presented approach aims to make such loop transformations more controllable via explicit sequences of pragma directives. In this regard, it is supposed to make these loop transformations more pragma-based, capable of optimization only part of a schedule tree and generation nested schedule trees for the performance portability of code for upcoming target architectures.

Experimental runs were carried out using available ccNUMA macronodes with deep memory hierarchies, and the early-stage results are reported. The SCoP rejections ratio was obtained using a comprehensive guideline for SCoP validation. To this end, the HPC-relevant polyhedral benchmark prototype was developed, and improved results of locality and speedup were evaluated. Computational kernels currently include 2D and 3D stencils. The main result of applying the approach was obtaining tiled code with optimal locality for performance-critical pragma-defined code sections. Future work will include further diagnostic of SCoP rejections and TSS optimization using the developed benchmark with an extension of the TSS algorithm for cases beyond rectangular tiling.

## References

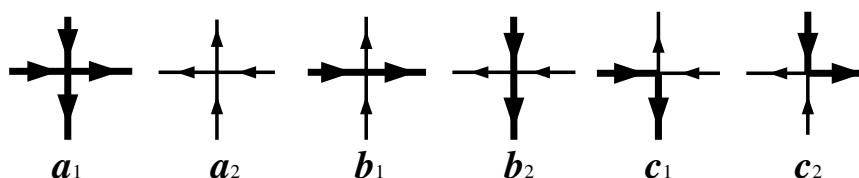
1. Tobias Grosser, Armin Größlinger, Christian Lengauer. Polly — Performing Polyhedral Optimizations on a Low-Level Intermediate Representation // Parallel Processing Letters. 2012. Vol. 22. No. 04. DOI: 10.1142/S0129626412500107
2. Oleksandr Zinenko, Sven Verdoolaege, Chandan Reddy, Jun Shirako, Tobias Grosser, Vivek Sarkar, Albert Cohen. Modeling the Conflicting Demands of Parallelism and Temporal/Spatial Locality in Affine Scheduling // 27th International Conference on Compiler Construction (CC'18). Vienna, Austria. February 24–25, 2018. P. 3–13. DOI: 10.1145/3178372.3179507
3. Sven Verdoolaege, Serge Guelton, Tobias Grosser, Albert Cohen. Schedule Trees // 4th International Workshop on Polyhedral Compilation Techniques (IMPACT). Vienna, Austria. January 20, 2014.
4. Johannes Doerfert, Clemens Hammacher, Kevin Streit, Sebastian Hack. SPolly: Speculative Optimizations in the Polyhedral Model // 3rd International Workshop on Polyhedral Compilation Techniques (IMPACT). Berlin, Germany. January 21, 2013.
5. Michael Kruse, Hal Finkel. User-Directed Loop-Transformations in Clang // 5th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC). Dallas, TX, USA. November 12, 2018. P. 49–58. DOI: 10.1109/LLVM-HPC.2018.8639402
6. Xingfu Wu, Michael Kruse, Prasanna Balaprakash, Hal Finkel, Paul Hovland, Valerie Taylor. Autotuning PolyBench Benchmarks with LLVM Clang/Polly Loop Optimization Pragmas Using Bayesian Optimization // Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS). GA, USA. November 12, 2020. P. 61–70. DOI: 10.1109/PMBS51919.2020.00012

# Monte Carlo simulations of the two-point correlation functions in the six-vertex lattice model\*

P. A. Belov

Spin Optics Laboratory, Saint-Petersburg State University

This report is devoted to a computation of correlation functions in the six-vertex model by the Markov chain Monte-Carlo (MC) simulations [1]. This model was introduced to describe the crystal where the oxygen groups form a square lattice with a hydrogen atom between each pair of lattice sites [2]. The states in this model are designed as configurations of arrows on edges which satisfy the ice rule [3]: for each vertex there are two incoming and two outgoing arrows, see Fig. 1. Equivalently, configurations of arrows can be regarded as configurations of lattice paths such that paths may meet at a vertex, turn or pass [4]. The lattice paths are denoted in the figure as bold edges. The weight of a vertex depends on the configurations of paths on adjacent edges. The Boltzmann weight of a configuration is a product of Boltzmann weights assigned to vertices.



**Figure 1.** Local configurations and weights of the six-vertex model.

The parameter  $\Delta = (a^2 + b^2 - c^2)/(2ab)$ , where  $a, b, c$  denote Boltzmann weights, defines the phase of the six-vertex model [4]. When  $\Delta = 0$ , the partition function and correlation functions can be computed in terms of the determinant and the minors of the Kasteleyn matrix [5], respectively.

In the last decades, a particular attention is paid to the six-vertex model with so-called domain wall boundary conditions (DWBC) on a square lattice. These boundary conditions correspond to paths coming through the top side of the square and leaving through the right side. In this case, the partition function can be calculated exactly as a determinant [6].

Locally, lattice paths of the six-vertex model on a planar lattice can be regarded as level curves of the integer valued step-function  $\chi(n, m)$ , called the height function and defined on faces [4]. It is a random variable, but we additionally assume that it increases when we move to the right and up. In the large volume limit (the thermodynamic limit,  $N \rightarrow \infty$ ), there is the limit-shape phenomenon: the properly normalized height function converges, as a random variable, to a deterministic function  $h_0(x, y)$  known as the limit-shape height function [7, 8]:

$$\chi(n, m) \rightarrow Nh_0\left(\frac{n}{N}, \frac{m}{N}\right) + \phi\left(\frac{n}{N}, \frac{m}{N}\right). \quad (1)$$

Here, the random variable  $\phi(x, y)$  is a free Gaussian quantum field in the Euclidean space-time with the metric determined by the height function  $h(x, y)$ . The two-point correlation function of points  $(x_i, y_i)$  and  $(x_j, y_j)$  can be calculated as

$$\langle \phi(x_i, y_i), \phi(x_j, y_j) \rangle = \langle \chi(x_i, y_i)\chi(x_j, y_j) \rangle - \langle \chi(x_i, y_i) \rangle \langle \chi(x_j, y_j) \rangle. \quad (2)$$

\*The report is supported by the Russian Science Foundation, grant No. 21-11-00141. The calculations were carried out using the facilities of the ‘‘Computational Center of SPbU’’.

In our numerical study, we use the Markov chain MC simulation to generate a sequence of random states of the six-vertex model with DWBC. We construct the Markov chain based on the transition probabilities which satisfy the detailed balanced condition and allow to transfer from an arbitrary distribution to the desired one [9,10]. When a random process is constructed, the expectation values of observables with respect to the Boltzmann distribution can be computed by averaging along the random process [11]. This procedure is especially effective since the Boltzmann distribution is concentrated in a small neighborhood of the limit-shape.

In this report, we calculate the two-point correlation functions based on the MC generated configurations. Our results are explicitly given in Ref. [1]. The brief description is following. When  $\Delta = 0$ , both the limit-shape height function and the correlation functions are known from the exact solution. We calculate the values of these functions to test our algorithm. In an agreement with the exact solution, our numerical method confirms the logarithm-like dependence of correlation functions on the distance between points. After that, we apply our algorithm for other values of  $\Delta$ , for which the exact solutions are unknown. When  $|\Delta| \leq 1$ , the model is critical, i.e. the Gaussian field  $\phi(x, y)$  is a massless field on the space-time with the metric induced by the limit-shape. The numerics shows the logarithm-like dependence of correlation functions at short distances for this phase. This is natural since the case  $\Delta = 0$  is included in the disordered phase  $|\Delta| \leq 1$ . When  $\Delta < -1$ , an anti-ferroelectric diamond-shape droplet forms in the middle of the limit-shape. Since the antiferroelectric ground state is double degenerate, the Markov process gets stuck in one of the ground states for very long time. Nevertheless, we managed to obtain the exponential decay of the correlation functions for this phase.

The practical realization of the algorithm includes the parallel MC generation of random configurations. The OpenMP parallelization is applied for the CPU version of the algorithm. The GPU implementation using OpenCL is based on the code [12]. The parallel CPU and GPU implementations are discussed and compared in this report.

## References

1. P. A. Belov, N. Yu. Reshetikhin. The two-point correlation function in the six-vertex model. arXiv:2012.05182 (2020).
2. L. Pauling. J. Am. Chem. Soc. 57, 2680 (1935).
3. R. J. Baxter. Exactly solved models in statistical mechanics. San Diego, AP, 1982.
4. N. Reshetikhin. Lectures on the six-vertex model. OUP, Oxford, 2010. p. 197.
5. P. W. Kasteleyn. Physica 27, 1209 (1961).
6. V. E. Korepin, N. M. Bogoliubov, and A. G. Izergin. Quantum inverse scattering method and correlation functions. CUP, 1993.
7. F. Colomo and A. G. Pronko. J. Stat. Mech. 2005, P05010 (2005).
8. A. Okounkov. Bull. Am. Math. Soc. 53 2, 187-216 (2016).
9. D. Allison and N. Reshetikhin. Ann. Inst. Fourier 55, 1847 (2005).
10. N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. J. Chem. Phys. 21, 1087 (1953).
11. D. P. Landau and K. Binder. A guide to Monte Carlo simulations in statistical physics. CUP, 2005.
12. D. Keating and A. Sridhar. J. Math. Phys. 59, 091420 (2018).

# Parallel computing of a cavitation bubbles formation in a two-stage reciprocating pump-compressor

A.V. Zanin

Omsk State Technical University

Nowadays, applied fluid mechanics is developing as well as various fields of information technology. When one of the fluid mechanics' problems is faced with processing large amounts of data, high-performance computing (HPC) technologies, especially parallel computing, become indispensable tools.

This research aims to study the problem of cavitation bubbles formation and the cavitation bubbles dynamics between stages of a two-stage reciprocating pump-compressor. Bubbles formation is a hot topic in mechanical engineering, so a deeper understanding of this problem will help avoid future equipment failure.

Based on previous studies [1], it is planned to use a mathematical model of the fluid movement of such a machine [2]. It is necessary to add a free movement of bubbles during fluid dynamics between the stages of a pump in this model.

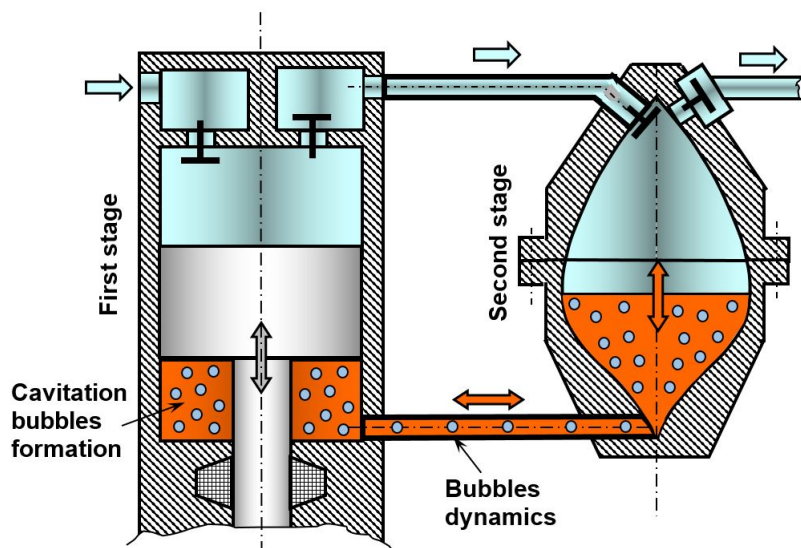


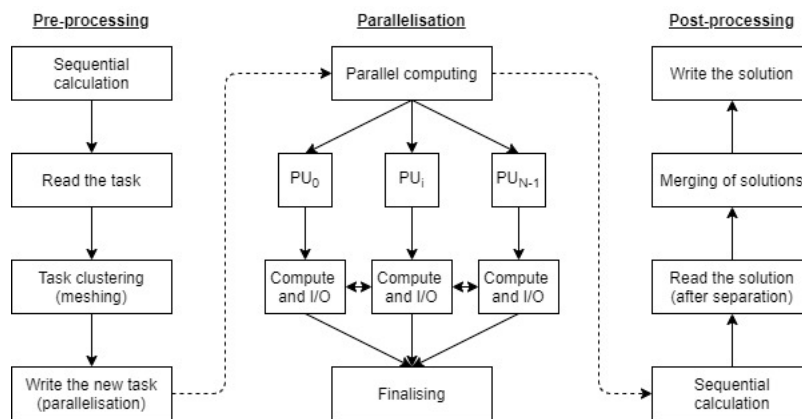
Figure 1. General view on a problem

This study assumes to use of parallel computing algorithms. Such algorithms are necessary for parallelising each bubble formation with a separate task, which simpler sub-tasks will further separate. It will also require developing an algorithm for each bubble dynamics, where hundreds or thousands of them can be in addition with different sizing. It depends on the desire to obtain more accurate results.

Modern flow solvers have to efficiently run on an extensive range of supercomputers, from single-core/processing unit (PU) to massively parallel platforms. The most common systems used for HPC applications are the Multiple Instruction Multiple Data (MIMD) based platforms [3]. From this category can be further used the following sub-category, Fig. 2.

Single Program, Multiple Data (SPMD). Multiple autonomous PUs simultaneously execute the same program (but at independent points) on different data. The flow solver Palabos and elsA are two examples of an SPMD program [4, 5].

Palabos will be used for the formation of the bubbles, where take place a complex physics



**Figure 2.** SPMD concept as a modern flow solver model

problem. Palabos is a C++ software platform developed since 2010 for Computational Fluid Dynamics simulations and Lattice Boltzmann modelling, which explicitly targets applications with complex, coupled physics. The software proposes a comprehensive modelling framework capable of addressing many applications of interest in the Lattice Boltzmann community yet exhibits solid computational performance.

ElsA will be used for the bubbles dynamics problem, where take place complex turbulent flows. The elsA software is a multi-application CFD simulation platform and deals with internal and external fluid-/aero-dynamics from low subsonic to hypersonic flow regime. The compressible 3D Navier-Stokes equations for arbitrary moving bodies are considered in several formulations according to the use of absolute or relative velocities. A large variety of turbulence models from eddy viscosity to full differential Reynolds stress models is implemented for the Reynolds-averaged Navier-Stokes (RANS) equations, including criteria to capture laminar-turbulent transition phenomena for academic and industrial geometries.

As a result of this research, unique parallel computing algorithms will be obtained that will help to solve problems of this type with various input and boundary conditions. Also, a methodology will be developed based on the research data.

## References

1. Zanin A., Pavlyuchenko E., Shcherba V. Numerical and experimental study on fluid compressibility in a two-stage reciprocating pump-compressor. *Appl. Therm. Eng.*, 194 (2021), Article 117106, DOI: 10.1016/j.applthermaleng.2021.117106.
2. Shcherba V.E., Zanin A.V. Nosov E.Y. Mathematical Modeling of Fluid Flow in a Positive Displacement Two-Stage Hybrid Power Reciprocating Machine with a Profiled Second Compression Stage. *Chem Petrol Eng* 56, 809-821 (2021), DOI: 10.1007/s10556-021-00846-8.
3. Flynn M.J. Some Computer Organizations and Their Effectiveness. *IEEE Transactions on Computers*, Vol. C-21, no. 9, 948-960, Sept. 1972, DOI: 10.1109/TC.1972.5009071.
4. Latt J. et.al. Palabos: Parallel Lattice Boltzmann Solver. *Computers Mathematics with Applications*, Vol. 81, 334-350 (2021), DOI: 10.1016/j.camwa.2020.03.022.
5. Gicquel Laurent Y.M. et.al. High performance parallel computing of flows in complex geometries. *Comptes Rendus Mecanique*, Vol. 339, Is. 2-3, 104-124 (2011), DOI: 10.1016/j.crme.2010.11.006

# Research of explicit numerical methods calculations on CPU with x86 and ARM architecture \*

Egor Elchinov<sup>1</sup>, Vladislav Furgailo<sup>1</sup>, Nikolay Khokhlov<sup>1</sup>

Moscow Institute of Physics and Technology (National Research University)<sup>1</sup>

Explicit numerical methods are used for a wide range of scientific problems in which there is a grid dependence between the calculated values at neighboring nodes of the computational grid. However, for high performance computing requires to use special techniques for processing and storing data. We examined the question of the optimal use of SIMD vector instructions [1] of CPU in stencil computation and improved the use of the hierarchical structure of the memory of the CPU caches by optimizing data locality - loop tiling [2–6]. We have applied optimization algorithms for x86 and ARM architectures. However, the novelty of this research is the optimization of the ARM architecture for the task of computing by the FDTD method and the assessment of the effectiveness of using the ARM architecture for 3D solving acoustics equation by FDTD method.

The following benchmarks were conducted on the four-core Cortex-A53 CPU with AArch64 ISA and Neon SIMD extension for ARM architecture and on the six-core Intel (R) Xeon (R) E5-2620 v2 (2.6 GHz) with AVX extension and Hyper-threading technology for the x86 architecture. The gcc v.8.1.0 and g++ v.8.1.0 compiler and the OpenMP framework for parallel computation with double precision numbers were used.

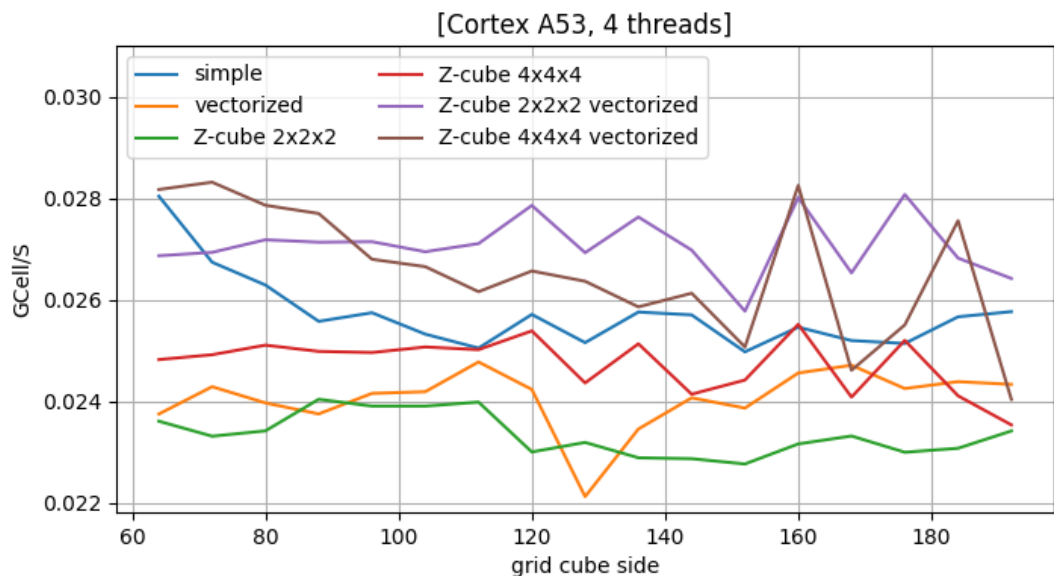
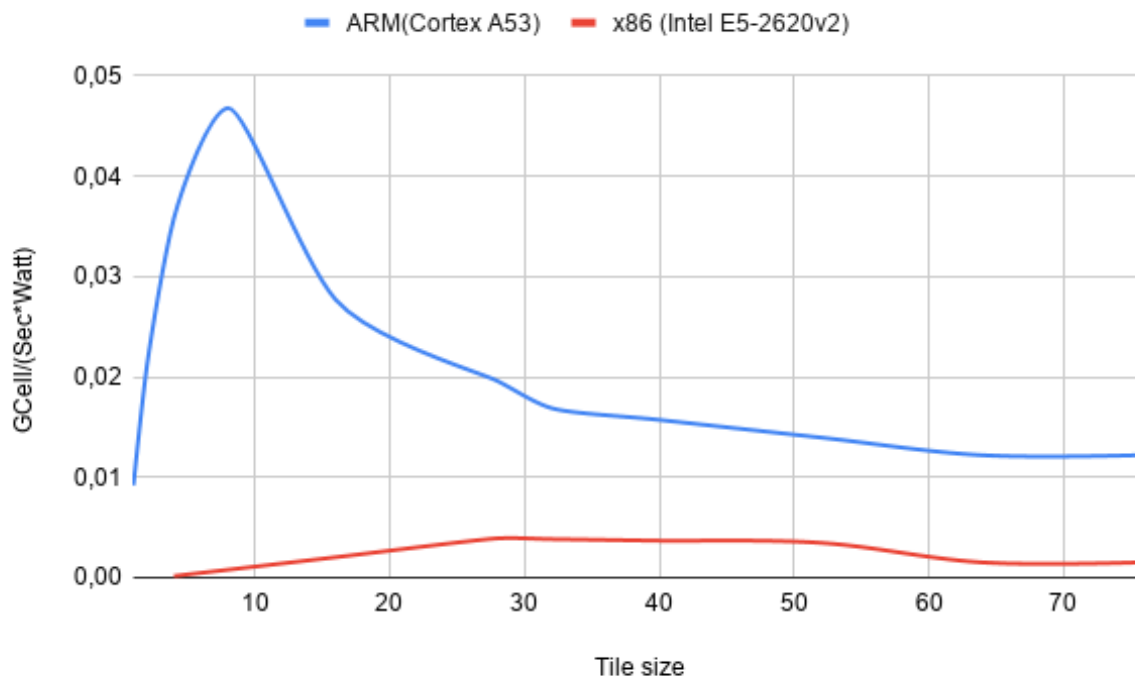


Figure 1. Z-cube efficiency for the variable grid size

In our work, we implemented new data locality algorithm that increase the performance of multi-threaded explicit stencil computation. Vectorization over the outer space of iterations and Z-Cube recursive tiling were applied to achieve data locality and to speed up multi-threaded computing as shown on Fig 1. Low performance of Neon-computing causes due to an overflow of Cortex-A53 data cache [7]. Consequently, Z-cube tiling improve the performance of utilizing Neon data-registers and instructions. However, non-recursive tiling remains a more effective data localization algorithm to FDTD problem.

\*The reported study was funded by RFBR according to the research project No. 18-07-00914 A.

Also, as shown in Fig. 2, the computation of explicit numerical equations on the ARM architecture by non-recursive tiling is 12 times more energy efficient in peak power consumption. In this respect, extending our experiments on ARM-cluster computing with increasing performance of non-recursive and recursive tiling would be of interest.



**Figure 2.** Graph of non-recursive vectorized tiling performance/power effectiveness of x86 and ARM.

## References

1. S. M. et. al., “Vector instructions to enable efficient synchronization and parallel reduction operations,” U.S. Patent WO2009120981A2, Oct. 2009.
2. J. Xue, “On tiling as a loop transformation,” *Parallel Processing Letters*, vol. 07, no. 04, pp. 409–424, 1997.
3. V. Levchenko and A. Perepelkina, “Locally recursive non-locally asynchronous algorithms for stencil computation,” *Lobachevskii Journal of Mathematics*, vol. 39, pp. 552–561, 05 2018.
4. V. Levchenko, A. Perepelkina, and A. Zakirov, “Diamondtorre algorithm for high-performance wave modeling,” *Computation*, vol. 4, p. 29, 08 2016.
5. A. Perepelkina and V. Levchenko, “The diamondcandy algorithm for maximum performance vectorized cross-stencil computation,” *Keldysh Institute Preprints*, pp. 1–23, 01 2018.
6. V. Furgailo, A. Ivanov, and N. Khokhlov, “Research of techniques to improve the performance of explicit numerical methods on the cpu,” pp. 79–85, 09 2019.
7. J. Bakos, *Embedded Systems: ARM Programming and Optimization*. Elsevier Science, 2015.



## Генератор пакетных файлов для планировщика задач SLURM\*

И.А. Михайлов, П.С. Костенецкий

Национальный исследовательский университет «Высшая школа экономики»

Данная статья посвящена разработке веб-приложения для автоматизации процесса формирования sbatch-файла, с помощью которого пользователь ставит свои вычислительные задачи в очередь суперкомпьютера НИУ ВШЭ через планировщик заданий SLURM. Генератор пакетных файлов содержит набор необходимых для запуска опций, таких, как количество и тип вычислительных узлов, количество графических процессоров, число ядер центральных процессоров, предельное время расчета, параметры параллелизма, а также функцию уведомления по электронной почте. Для упрощения работы пользователя, генератор автоматически подгружает с суперкомпьютера актуальные параметры, в частности набор установленных модулей.

Были рассмотрены существующие решения, в частности генератор пакетных файлов Brigham Young University [1] и научного центра NERSC [2]. Однако программная конфигурация и аппаратура суперкомпьютерных вычислительных комплексов сильно различаются, а опции генератора пишутся под конкретный вычислитель. В частности, индивидуальными параметрами являлись: 1) особенности работы с одной очередью задач и четырьмя типами вычислительных узлов, 2) приоритеты выбора конкретного типа узла, 3) возможность выбора сочетания сразу нескольких типов узлов, удовлетворяющих запросу пользователя, 4) выбор научного проекта, к которому относится задача.

Было принято решение о разработке нового генератора пакетных файлов специально под суперкомпьютер НИУ ВШЭ «CHARISMa» [3]. Для проектирования системы были определены следующие функциональные требования, в рамках которых система должна:

- синхронизироваться с актуальной конфигурацией суперкомпьютера;
- проверять и форматировать данные, введенные пользователем;
- повышать осведомленность пользователей о существующих возможностях кластера.

Нефункциональные требования, сформулированы следующим образом. Система должна:

- использовать открытое программное обеспечение суперкомпьютера SLURM и Cron;
- использовать открытый JavaScript фреймворк Vue.js для создания пользовательских интерфейсов и библиотеку jQuery;

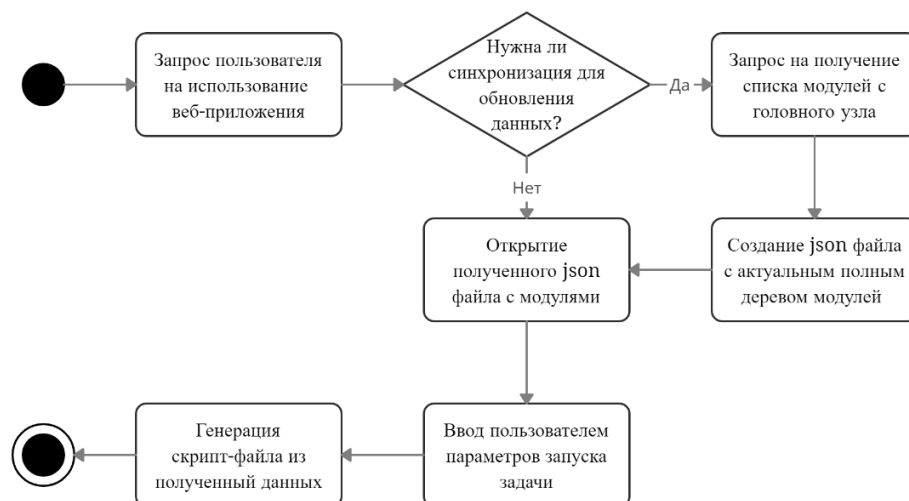


Рис. 1. Алгоритм работы системы

\* Исследование выполнено с использованием суперкомпьютерного комплекса НИУ ВШЭ [3].

На суперкомпьютере “сHARISMa” НИУ ВШЭ выполняются частые обновления программных библиотек и инструментов, поэтому веб-приложение автоматически подгружает актуальный список модулей и настроек перед использованием. Кроме того, генератор позволяет выбрать и подключить те модули, которые изначально требуют наличия зависимостей для своего отображения в списке доступных на инициализацию. Алгоритм работы системы представлен на Рис. 1.

При реализации проект получил компонентную архитектуру согласно идеям платформы Vue.js. Данная технология реализует способность веб-приложения модифицировать свою структуру и поведение для создания адаптивного дизайна. При помощи данной платформы, к веб-приложению подключены корпоративные стиливые файлы, используемые в разработках НИУ ВШЭ.

Разработанное web-приложение подходит не только для опытных пользователей, но и для тех, кто работает с суперкомпьютером впервые, например, проходит практику в рамках учебной дисциплины.

В дальнейшем разработанное приложение будет встроено в Единый личный кабинет сотрудника НИУ ВШЭ, а также интегрировано с системой мониторинга эффективности задач суперкомпьютера [4]. В дальнейшем планируется реализация возможности выбора научного проекта для каждой задачи, что предварительно потребует организации единой авторизации и взаимодействия между несколькими системами управления суперкомпьютерным комплексом, необходимой для получения списка проектов и определения прав пользователя на них.

## Литература

1. Cox R. Scheduling and Resource Management: Slurm // The 20th LCI Workshop, Urbana, 4-8 August, 2014, Proceedings.
2. He Y., Cook B., Deslippe J., et al. Preparing NERSC users for Cori, a Cray XC40 system with Intel many integrated cores. *Concurrency and Computation: Pract Exper.* 2018;30:e4291.
3. Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics // *Journal of Physics: Conference Series.* 2021. Т. 1740, № 1.
4. Шамсутдинов А.Б., Костенецкий П.С. Разработка системы мониторинга эффективности задач на суперкомпьютере сHARISMa // *Параллельные вычислительные технологии ПаВТ'2021*, г. Волгоград.

## Методы моделирования электронного газа в плоском диоде

И.В. Куликова

АО «НПП «Исток» им. Шокина»

Электронный поток в вакууме представляет собой однокомпонентную холодную (бесстолкновительную) плазму, в которой необходимо учитывать дальние взаимодействия, и описывается уравнениями Власова – Пуассона [1].

Прежде чем переходить к многомерным задачам было решено исследовать методы решения на одномерном случае. В качестве тестовой модели был выбран одномерный вакуумный диод и поставлена задача построить вольт-амперную характеристику (ВАХ) диода во всем диапазоне напряжений, используя одну модель.

Было использовано три метода: Эйлера, Лагранжа и Эйлера – Лагранжа (ALE метод). Во всех трех случаях одномерный электронный пучок двигался в двумерном фазовом пространстве  $\langle z, v_z \rangle$ . Уравнения движения электронного газа и уравнение Пуассона решались согласованно. Для сравнения методов были написаны программы на языке MATLAB, которые запускались на машине Intel Core i7-4770 3.2 GHz CPU, 32 GB RAM.

Метод Эйлера представляет собой решение уравнения Власова – Пуассона методом контрольных объемов с использованием схемы расщепления [2,3]. Для расчета концентрации и плотности тока использовалось численное интегрирование функции распределения по скорости. Разброс по скоростям задается граничными условиями на катоде и ограничен только количеством узлов и областью значений по скорости. На расчет одной точки ВАХ уходило 1,8 часа, размерность сетки  $500 \times 500$  элементов (см. рис. 1).

Метод трубок тока в фазовом пространстве реализует концепцию метода Лагранжа. При расчете концентрации по формуле  $n = I / (v_z q_e)$  в процессе отражения частицы от потенциального барьера возникают трудности, поскольку при  $v_z \rightarrow 0$  концентрация  $n \rightarrow \text{inf}$ . Поэтому для расчета концентрации применялись различные алгоритмы интерполяции и ограничители. И даже, не смотря на это, график концентрации в области торможения имел зубчатый вид. Разброс по скоростям в данном методе задавался дополнительными частицами. На расчет одной точки ВАХ уходило 4,5 минуты, размерность сетки: 500 элементов и 500 частиц.

Метод подвижной сетки является разновидностью метода Эйлера-Лагранжа. Подвижная сетка была сформирована по скорости. Благодаря этому время расчетов сократилось в

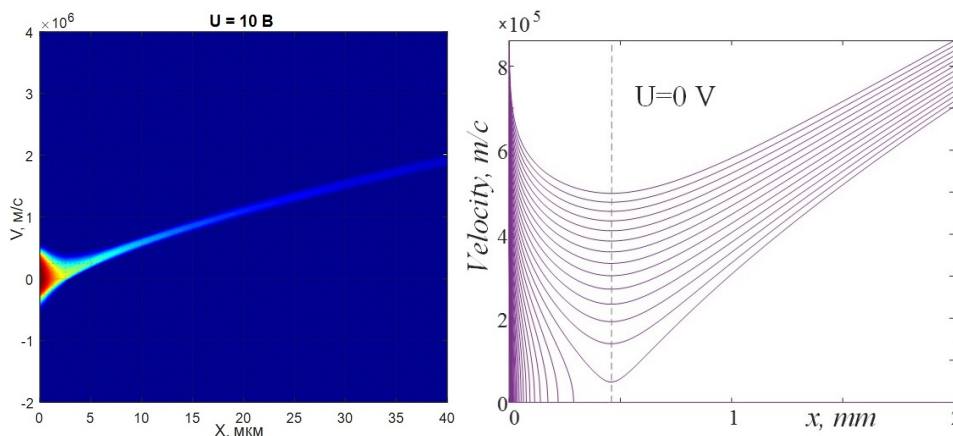
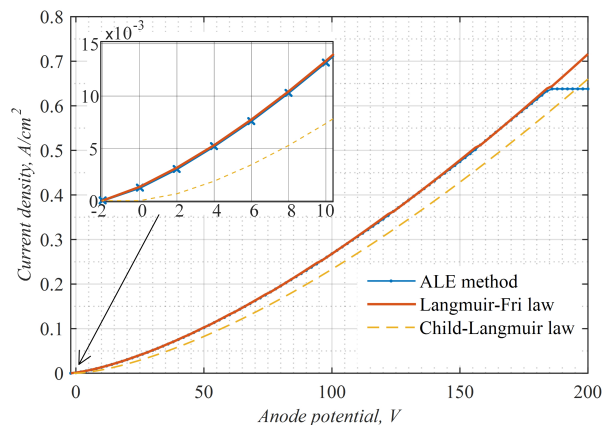


Рис. 1. Функция распределения при расчете методом Эйлера и траектории трубок тока в фазовом пространстве при расчете методом Лагранжа.



**Рис. 2.** ВАХ вакуумного диода, построенная при моделировании поведения электронного газа ALE методом.

10 раз и составило 15 секунд. ВАХ, построенные ALE-методом и с использованием аналитических законов Чайлда-Ленгмюра (закон степени  $3/2$ ) и Ленгмюра – Фрая [4], приведены на рис.2.

Все три метода показали хорошую сходимость во всем диапазоне запирающих и ускоряющих напряжений. ВАХ, построенные по всем трем методам, совпадают с ВАХ, рассчитанной по Ленгмюру – Фраю.

*Выводы:* Применение метод Эйлера – Лагранжа позволило в 100 раз сократить количество элементов по пространству скоростей с применением квадратичных функций без потери точности во всем диапазоне анодных напряжений.

Тормозящее электрическое поле увеличивает разброс по скоростям в электронном потоке. В то же время, ускоряющее поле значительно сокращает данный разброс.

Расстояние от катода до минимума потенциала составляет от единиц микрон до сотен микрон и зависит от приложенного потенциала и расстояния до анода. Потенциальный барьер не превышает двух вольт.

При расчете электровакуумных приборов с характеристическими микронными размерами закон Чайлда – Ленгмюра будет давать заниженное значение тока.

В дальнейшей работе планируется применить данный метод для двумерного случая в цилиндрической системе координат на неструктурированной сетке, в качестве препроцессора и постпроцессора использовать GMSH, а решатель написать на языке программирования GO, используя механизм многопоточности goroutine.

## Литература

1. Власов А.А. Теория многих частиц. М.: Изд-во «Государственное издательство технико-теоретической литературы», 1950. 350 с.
2. G. Vogman, Ph. Colella, U. Shumlak. High-order continuum kinetic method for modeling plasma dynamics in phase space. AIP Conference Proceedings. 2014. P. 146-149. DOI: 10.1063/1.4904797.
3. Куликова И.В. Построение ВАХ вакуумного диода на основе численного решение уравнения Власова–Пуассона // Прикладная физика. 2020. № 2. С. 27–33.
4. Langmuir I. The effect of space charge and initial velocities on the potential distribution and thermionic current between parallel plane electrodes. Phys. Rev. 21, 419. Iss. 4 1923. DOI: 10.1103/PhysRev.21.419.

## Моделирование динамики горения водорода при помощи полносвязной нейронной сети UNET

Я.М. Карандашев, Е.В. Михальченко, М.Ю. Мальсагов, В.Ф. Никитин  
ФГУ ФНЦ НИИСИ РАН

Традиционно модели нейросетей типа UNET применялись для автоматического перевода картинку в картинку, в частности для сегментации изображений. В настоящей работе показано, что подобная архитектура может быть успешно применена к задаче предсказания детерминированных многомерных временных рядов, а именно моделирования химических процессов горения, описываемых жёсткой системой обыкновенных дифференциальных уравнений. Используя её, нам удалось обучить компактную модель, которая может аппроксимировать изменения концентраций веществ в смеси в процессе химических реакций с высокой степенью точности, достаточной чтобы рекуррентно получать предсказание на сотни и даже тысячи шагов интегрирования вперёд, занимая при этом на порядок меньше времени вычисления, чем численное интегрирование.

*Ключевые слова:* численное моделирование химических процессов, горение, детонация, нейронные сети, глубокое обучение, UNET, рекуррентные сети.

В продолжение работы по нейросетевому моделированию химической кинетики [1] мы сделали несколько модификаций в нашем подходе. Рассматривается задача горения водорода с кислородом в присутствии нейтральных элементов. В качестве нейтральных элементов выступают азот (N<sub>2</sub>) и аргон (Ar). Такая система в процессе своего развития содержит 10 компонент: азот, аргон и 8 водородно-кислородных соединений (H<sub>2</sub>, O<sub>2</sub>, H<sub>2</sub>O, OH, HO<sub>2</sub>, H<sub>2</sub>O<sub>2</sub>, H, O), а также описывается 28 уравнениями химических реакций. Еще присутствуют два физических параметра: давление и температура.

С помощью численного моделирования, мы создали датасет, в котором начальная смесь состоит только из молекулярного кислорода и водорода при различных их концентрациях и температурах. Шаг времени 0.1 мкс, давление P = 1.0 бар. Остальные переменные задавались случайным образом: температура в диапазоне значений от 1000 до 2000 градусов Кельвина, молярные плотности компонент H<sub>2</sub> и O<sub>2</sub> от 0 до 20 моль/м<sup>3</sup>, молярные плотности компонент N<sub>2</sub> и Ar<sub>2</sub> от 0 до 15 моль/м<sup>3</sup>, молярные плотности остальных компонент (радикалов и соединений) задавалась равной нулю. Было проведено 10000 экспериментов по 500 шагов времени каждый.

Данный датасет использовался для обучения и тестирования нейронной сети. Для этого датасет был разбит на три части: первые 8000 экспериментов попали в обучающую выборку. Следующие 1600 экспериментов в тестовую выборку. Последние 400 экспериментов в валидационную выборку.

Предварительные эксперименты показали, что в большинстве случаев молярные плотности веществ изменяются очень медленно в самом начале реакции. В связи с этим, было решено сделать логарифмическую перенормировку данных, чтобы малые изменения вблизи нуля стали более заметны:

$$y = \ln(1 + x / \varepsilon) \quad (1)$$

где  $\varepsilon = 10^{-10}$ . Вдобавок к логарифмическому масштабированию (1) был применён стандартный скейлинг данных – вычитание среднего и деление на стандартное отклонение:

$$z = (y - \bar{y}) / \sigma_y \quad (2)$$

Отметим, что две данные трансформации применялись только для молярных плотностей компонент. Для температуры применялся только стандартный скейлинг (2) без логарифмического масштабирования (1).

Для моделирования была выбрана архитектура нейронной сети, в которой присутствует обходная связь (skip-connection) от входа к выходу – типа архитектуры UNET [2], только с полносвязными слоями (см.рис.1). Таким образом нейронной сети необходимо лишь выучить изменение концентрации веществ во времени, что позволяет параметрам нейронной сети обучаться более быстро (обычно 50-100 эпох обучения достаточно).

Как показывают эксперименты, при многократном рекуррентном запуске нейронной сети на несколько сотен шагов медленно, но неизбежно набегают ошибки. Чтобы уменьшить набегающие ошибки и улучшить качество предсказания, было предложено включить в лосс функцию результаты предсказания нейросети на несколько шагов вперёд в рекуррентном режиме (см. формулу (3)). Сеть по-прежнему делает предсказание на один шаг вперёд  $X_{t+1}$ . Далее, это значение  $X_{t+1}$  подаётся на вход, получается выход сети на втором шаге  $X_{t+2}$ . Далее, это значение снова подаётся на вход, получается предсказание на три шага  $X_{t+3}$  и т.д. до некоторого выбранного значения  $n_{steps}$ . Эти значения участвуют в построении лосс функции при обучении:

$$L_{nsteps} = \sum_{i=1}^{nsteps} loss(X_{t+i} - X_{t+i}) \quad (3)$$

где  $loss$  - произвольная функция ошибки (в нашей работе использовалась MSE).

Благодаря предложенной архитектуре и способу обучения, нейронная сеть учится намного точнее, а ошибка уменьшается на 2-3 порядка (см. таблицу 1), при том что количество слоёв в ней значительно всего 5. Такая сеть может работать на порядок быстрее, чем численный расчёт.

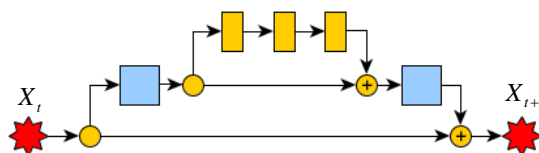


Рис. 1. Архитектура сети, сеть с одним уровнем UNET.

Таблица 1. Сравнение различного количества шагов предсказания  $n_{steps}$  во время обучения.

$n_{steps}$	$\langle MSE \rangle$	$\sigma_{MSE}$	$MSE_{max}$	$\langle MAE \rangle$	$\sigma_{MAE}$	$MAE_{max}$
1	0.00114	0.00254	0.0228	0.0151	0.0124	0.0899
2	0.000728	0.00215	0.0268	0.0110	0.00950	0.0675
5	0.000492	0.00113	0.0139	0.0103	0.00806	0.0715
10	0.000217	0.000526	0.00480	0.00635	0.00553	0.0432
20	8.97E-05	0.000189	0.00172	0.00414	0.00363	0.0250
30	8.77E-05	0.000211	0.00170	0.00388	0.00340	0.0228
40	0.000134	0.000541	0.00755	0.00484	0.00424	0.0387
50	8.86E-05	0.000204	0.00221	0.00416	0.00306	0.0188

## Литература

1. V.B. Betelin, B.V. Kryzhanovsky, N.N. Smirnov, V.F. Nikitin, I.M. Karandashev, M. Yu Malsagov, E.V. Mikhilchenko, Neural network approach to solve gas dynamics problems with chemical transformations, Acta Astronautica, 2020, ISSN 0094-5765, <https://doi.org/10.1016/j.actaastro.2020.11.058>
2. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In International Conference on Medical image computing and computer-assisted intervention, pp. 234-241. Springer, Cham, 2015.

## Моделирование процесса распространения загрязняющих веществ в Геленджикской бухте на высокопроизводительной вычислительной системе\*

Ю.В. Белова<sup>1</sup>, Е.О. Рахимбаева<sup>1</sup>, А.А. Филина<sup>2</sup>, А.В. Никитина<sup>1,2,3</sup>

Донской государственный технический университет<sup>1</sup>,  
 ООО «Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров»<sup>2</sup>,  
 Южный федеральный университет<sup>3</sup>

Развитие промышленности и технологического производства, особенно в течение последних десятилетий, приводит к значительному ухудшению экологического состояния водоемов, которые наиболее подвержены различным бытовым, сельскохозяйственным, промышленным загрязнениям. Ежегодно в водные бассейны попадают тысячи химических веществ с непредсказуемым действием, многие из которых представляют собой новые химические соединения. Работа посвящена построению, исследованию и численной реализации математической модели транспорта загрязняющих веществ, включая нефтепродукты, в мелководном водоеме с учётом ряда важных гидродинамических и гидрофизических факторов, оказывающих влияние на характер их протекания и методов ее численной реализации, позволяющих выполнять предсказательное моделирование распространения загрязнений в акватории мелководных систем в условиях ограниченного времени на высокопроизводительной вычислительной системе [1, 2]. Использование высокопроизводительных вычислительных систем особо необходимо при решении задач водной экологии, построения прогнозов (как краткосрочных, так и долгосрочных) состояния экологических систем, особенно в условиях природных и промышленных вызовах, когда требуется выполнить многочисленные высоконадёжные расчеты в ограниченное время. Отметим, что основным преимуществом использования высокопроизводительных вычислительных систем при моделировании процессов и явлений окружающей среды является учет мультидисциплинарного характера реальных процессов, сложная геометрия моделируемой области, обеспечение высокой точности вычислений, выполнение большого количества расчетов за короткое время.

Данная работа посвящена моделированию процессов распространения загрязняющих веществ, включая нефть и нефтепродукты, с учетом гидрофизических и биологических показателей водоема, сложной геометрии расчетной области, климатических факторов окружающей среды на высокопроизводительной вычислительной системе, позволившей получить результаты моделирования, сопоставимые с данными дистанционного зондирования Земли на качественном и количественном уровнях. Применимость разработанной модели к реальным условиям водоема установлена в соответствии с данными спутникового мониторинга моделируемой области.

Рассмотрим полную 3D систему уравнений распространения 3В загрязняющих веществ, включая нефть и продукты ее переработки, в расчетной области – Геленджикской бухте – с боковой поверхностью  $\sigma$ , невозмущенной поверхностью водоема  $\Sigma_0$ , дном  $\Sigma_H$ , переменной глубиной  $H$  в декартовой системе координат при осях  $Oz$ ,  $Oy$ ,  $Ox$ , направленных вертикально вниз, на север и восток, соответственно. Математическая модель имеет следующий вид:

$$\frac{\partial S_i}{\partial t} + u \frac{\partial S_i}{\partial x} + v \frac{\partial S_i}{\partial y} + (w - w_{gi}) \frac{\partial S_i}{\partial z} + \sigma S_i = \mu_i \left( \frac{\partial^2 S_i}{\partial x^2} + \frac{\partial^2 S_i}{\partial y^2} \right) + \frac{\partial}{\partial z} \left( \nu_i(z) \frac{\partial S_i}{\partial z} \right) + f(x, y, z, t), \quad (1)$$

где  $u, v, w$  – компоненты вектора скорости водного потока;  $w_{gi}$  – скорость гравитационного осаждения  $i$ -ой компоненты, если она находится во взвешенном состоянии;  $\sigma_i$  – коэффициент разложения  $i$ -й примеси;  $f$  – химико-биологический источник (сток);  $\mu_i, \nu_i$  – коэффициенты

\* Исследование выполнено при финансовой поддержке Совета по грантам Президента Российской Федерации в рамках научного проекта № МД-3624.2021.1.1



диффузии в горизонтальном и вертикальном направлениях;  $S_i$  – концентрация  $i$ -й примеси,  $i = \overline{1,6}$ : 1 – ртуть ( $Hg$ ); 2 – свинец ( $Pb$ ); 3 – марганец ( $Mn$ ); 4 – железо ( $Fe^{+2}$ ); 5 – фитопланктон (диатомовая водоросль *Skeletonema costatum*); 6 – нефть и нефтепродукты.

В модель включены ЗВ, концентрация которых в Геленджикской бухте, согласно проведенному анализу научных публикаций и баз экологических данных, превышает предельно допустимую концентрацию (ПДК). Данная модель учитывает движение водного потока; микротурбулентную диффузию; взаимодействие и гравитационное оседание загрязняющих примесей и планктона; биогенный, температурный и кислородный режимы; влияние солености.

К системе (1) добавляются соответствующие начальные и граничные условия.

Наиболее ресурсоёмкой частью программной реализации математической модели является функция решения системы линейных алгебраических уравнений (СЛАУ), возникающей при ее дискретизации. Для ускорения вычислений был разработан параллельный алгоритм, основанный на модифицированном попеременно-треугольном итерационном методе (МПТМ) решения СЛАУ и использующий технологию MPI для осуществления информационного обмена между вычислителями.

Для проведения калибровки, верификации разработанных моделей, а также для проверки адекватности построенных моделей использовались литературные источники, экспедиционные данные, которые получены авторами в ходе различных научно-исследовательских работ, а также спутниковые данные дистанционного зондирования Земли. На рисунке 1 представлена схема аппаратного программного комплекса, предназначенного для моделирования гидрофизических процессов мелководного водоема.

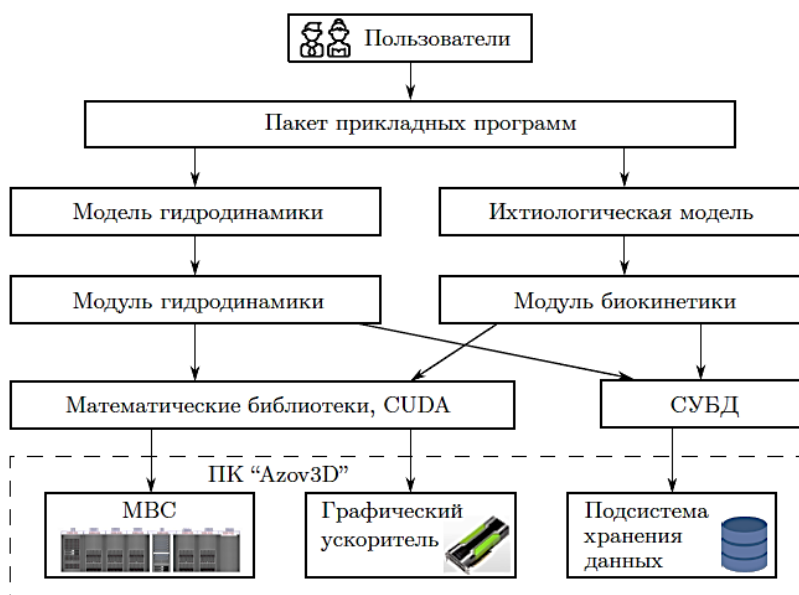


Рис. 1. Схема аппаратно-программного комплекса

Проведенные вычислительные и программно-аналитические эксперименты и расчеты показали адекватность и работоспособность построенной в данной работе математической модели распространения загрязняющих веществ в мелководном водоеме, интегрированной в программный комплекс, что позволяет использовать его при планировании прогностических расчетов определения качества вод в различные периоды времени.

## Литература

1. Игнатов А.В., Кравченко В.В. Информационное моделирование загрязнения водных объектов // Новосибирск: Изд-во СО РАН, 2008. – С. 144-150.
2. Самарский А.А., Вабищевич П.Н. Численные методы решения задач конвекции-диффузии. М.: URSS, 2009. 246 с.



# О ранговом распределении суперкомпьютеров в списке top500\*

С.К. Шикота<sup>1,2</sup>

<sup>1</sup> Национальный исследовательский университет «Высшая школа экономики»

<sup>2</sup> Научный центр РАН в Черноголовке

Мы анализируем ранговые распределения, построенные на основании списков суперкомпьютеров в списке TOP500. Используя модифицированный закон Ципфа, мы извлекли показатель степени, характеризующий ранговое распределение суперкомпьютеров по максимальной мощности на тесте LINPACK. Похоже, что в течение многих лет вычислительная мощность подчиняется закону Ципфа со значением показателя близким к значению 0.66. Мы обнаружили отклонение от этого закона в последних шести последовательных списках, начиная с июня 2018 года до ноября 2020 года, что можно отнести к новому явлению. В основном это суперкомпьютеры средней мощности с максимальной мощностью 1649 TFlops, предоставляющие услуги облачных сервисов, услуги хостинга и другие IT-сервисы. До 2017 г. включительно большая часть суперкомпьютеров из списка была установлена в исследовательских центрах и университетах. В конце 2019 года были ожидания, что эта тенденция продолжится и обсуждался кризис в научном секторе высокопроизводительных вычислений. Однако, развитие вычислительных мощностей в этом секторе продолжилось, как и в предыдущие годы. Более того, заметный рост числа суперкомпьютеров средней мощности в секторе IT услуг остановился и указанный выше закон Ципфа с показателем 0.66 восстановился.

*Ключевые слова:* суперкомпьютеры, top500, ранговое распределение, закон Ципфа.

## 1. Введение

С 1993 года дважды в год публикуется список TOP500 наиболее мощных суперкомпьютерных систем мира [1]. Этот список предоставляет уникальную информацию о суперкомпьютерных системах, которая полезна для анализа технологических направлений развития.

Настоящая заметка посвящена предварительному статистическому анализу одной из основных характеристик из списка TOP500  $R_{max}$  – максимальной производительности суперкомпьютеров, показанных на тесте LINPACK [2].

## 2. Ранговые распределения

Распределения со степенной зависимостью обнаружены в социальных сетях более ста лет назад. Парето [3] обнаружил, что распределение дохода  $D$  во всех странах подчиняется выражению

$$F(f) = 1 - (m/D)^\alpha \quad (1)$$

с показателем экспоненты  $\alpha \approx 1.5$  и некоторой константой  $m$ .

Тридцать лет спустя Ципф [4] обнаружил, что относительная частота  $f$  того, что в английском литературном тексте слово стоит на  $r$ -ом месте упорядоченного по частоте списка, обратно пропорциональна его месту (рангу) в этом списке

$$f_r \propto 1/r, \quad (2)$$

---

\*Работа выполнена при поддержке гранта РФФИ 20-07-00238 "Анализ данных и выявление закономерностей в развитии суперкомпьютерных приложений и суперкомпьютерных систем"

и этот закон выполняется для ранговых распределений в том случае, если среди элементов списка отсутствуют причинно-следственные связи.

В научной литературе применяется также обобщенная форма закона Ципфа (2) с показателем экспоненты  $\alpha$  с показателем экспоненты, принимающим любые положительные значения

$$f_r \propto 1/r^\alpha, \quad (3)$$

и  $0 < \alpha < 1$  в случае, если между элементами списка имеются причинно-следственные связи (корреляции).

Для выравнивания вклада с большими частотами Мандельбротом была предложена модификация, получившая название закона Ципфа-Мандельброта [5]

$$f_r = \frac{b}{(c+r)^\alpha}. \quad (4)$$

Для коррекции вклада малых частот и выравнивания хвоста распределения был введен еще один параметр [6]

$$f_r = a + \frac{b}{(c+r)^\alpha}. \quad (5)$$

что помогло установить с большой достоверностью выполнение оригинального закона Ципфа с показателем  $\alpha = 1$  при анализе рангового распределения в популярности web-сайтов [6].

В нашем анализе мы будем приближать данные с помощью именно такого приближения (5) к закону Ципфа.

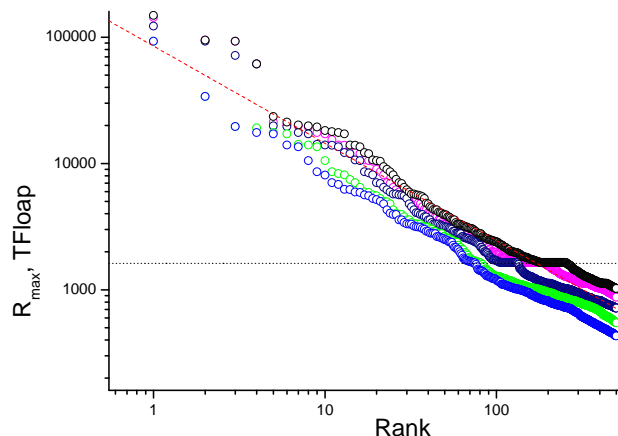
### 3. Ранговое распределение максимальной производительности $R_{max}$ суперкомпьютеров

Максимальная производительность  $R_{max}$  суперкомпьютеров измеряется на пакете линейной алгебры LINPACK [2] в единицах TFlops –  $10^{12}$  операций с плавающей запятой в секунду. На рисунке 1 показано ранговое распределение максимальной мощности  $R_{max}$  суперкомпьютеров из списков TOP500 с 2016 года по 2019 год (в порядке снизу-вверх). Заметим, что в случае степенного поведения по закону Ципфа, на рисунках в дважды логарифмическом масштабе (логарифм  $R_{max}$  – логарифм ранга  $r$ ) данные будут ложиться на прямые линии с наклоном  $\alpha$ . При этом будут отклонения от прямой при малом значении ранга  $r$ , для спрямления которых нужна поправка Мандельброта  $c$ , на и при больших значениях ранга  $r$ , для спрямления которых нужна поправка (смотри выражение (5)).

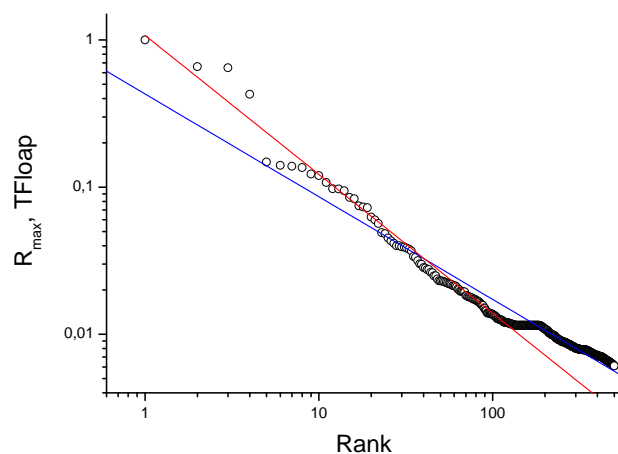
В каждом году публикуется два списка – июньский и ноябрьский. Штриховой линией на рисунке 1 показана прямая с наклоном, соответствующим показателю степени примерно  $\alpha = 0.66$ . Видно, что до 2018 года распределение мощности суперкомпьютеров неплохо соответствует этому значению.

В 2018 году произошло новое явление в ранговом распределении максимальной мощности суперкомпьютеров. На рисунке 2 показано отдельно ранговое распределение июньского списка 2018 года. Синей линией показан наклон прямой с показателем  $\alpha = 0.66$  и красной линией показан наклон прямой с показателем  $\alpha = 0.83$ , который получен с помощью приближения данных по формуле (5) с отбрасыванием данных для ранга более 100. Интересно, что в диапазоне рангов более 200 данные по-прежнему хорошо аппроксимируются наклоном с  $\alpha = 0.66$ . В диапазоне примерно от 100 до 200 видно плато.

Таким образом, в 2018 году в списке top500 нами обнаружено отклонение показателя рангового распределения от предыдущих лет и обнаружено плато.



**Рис. 1.** Ранговое распределение максимальной мощности  $R_{max}$  суперкомпьютеров из списков TOP500 с 2016 года по 2019 год.



**Рис. 2.** Ранговое распределение максимальной мощности  $R_{max}$  суперкомпьютеров из июньского списка TOP500 за 2018 год.

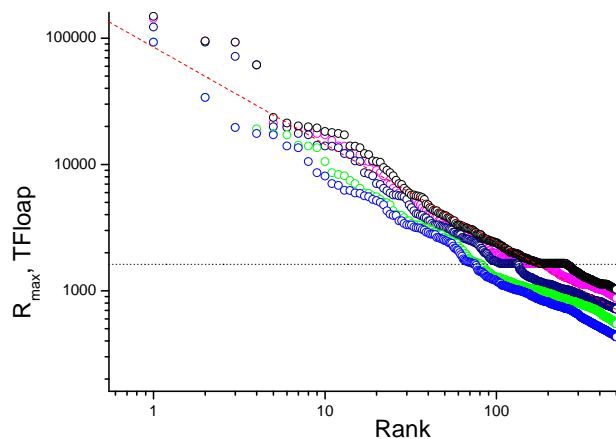
#### 4. О плато в ранговом распределении

Анализ списков TOP500 показал, что начиная с июня 2018 года в списке появилось много суперкомпьютеров с максимальной мощностью 1649 TFlops. Это компьютеры компании Lenovo, модель C1040 с 20-ти ядерными процессорами Xeon E5-2673v4 с 2.3 GHz. Общее число ядер 57600 и памяти 737280 GB суммарно на узлах, объединенных сетью 40G Ethernet. Такие суперкомпьютеры установлены не в традиционных секторах науки и образования, государственных учреждений и гидро-метео центрах, а в компаниях, осуществляющих услуги IT-сервисов хостинга и облачных услуг. Причем, компании расположены во многих странах - США, Австралия, Сингапур, Гонконг, Китай, Голландия, Норвегия и т.д.

Таким образом, новое явление связано с несколькими факторами. Во-первых, это технологическая возможность массового выпуска кластеров большой производительности, которую осуществила китайская компания Lenovo. Во-вторых, это резкое технологическое развитие мобильной связи и интернета-вещей. И в третьих, как отклик - резкий рост рынка облачных услуги и услуг хостинга. Если в первой половине 2018 года таких площадок

появилось 35, то к концу 2019 года их стало 85.

Интересно, что в 2020 году прироста таких площадок не произошло – их по-прежнему 85. Этому есть два возможных объяснения – 1) формирование рынка ИТ-услуг временно закончилось, достаточный задел ресурсов на сегодняшний день и 2) влияние пандемии.



**Рис. 3.** Ранговое распределение максимальной мощности  $R_{max}$  суперкомпьютеров из списков TOP500 с 2016 года по 2020 год.

Интересно, что влияние плато на общее ранговое распределение оказалось временным - на ноябрь 2020 года ранговое распределение показывает то же значение показателя  $\alpha = 0.66$ , что и до массового производства суперкомпьютеров для хостинга фирмой Lenovo. Это можно увидеть на рисунке 3.

## 5. Заключение

В работе был предложен статистический анализ данных о суперкомпьютерах из списка TOP500 на основе модифицированного рангового распределения и представлены предварительные результаты анализа максимальной производительности суперкомпьютеров. Численно определен показатель рангового распределения, который стабилен с 2016 по 2020 год. Исключение составило формирование плато из кластеров ИТ-сервисов хостинга и облачных услуг в 208-2019 годах на основе 85 суперкомпьютеров Lenovo C1040.

Работа выполнена в рамках гранта РФФИ 20-07-00238.

## Литература

1. E. Strohmaier, H.W. Meuer, J. Dongarra, and H.D. Simon, *The TOP500 List and Progress in High-Performance Computing*, Computer **48** (2015) 42.
2. The LINPACK Benchmark. <https://www.top500.org/project/linpack/>
3. V. Pareto, *Cours d'economie politique*, Lausanne, F. Rouge; Paris, Pichon, 1897.
4. G.K. Zipf, *Human Behavior and the Principle of Least-Effort*, Addison-Wesley, Cambridge, MA, 1949.
5. B.B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, New York, 1977.
6. S.A. Krashakov, A.B. Teslyuk, L.N. Shchur, *On the universality of rank distributions of website popularity*, Computer Networks, 50 (2006) 1769-1780.

# Оценка эффективности модифицированных CSR форматов представления матриц при реализации итерационных методов решения систем уравнений \*

Р.М. Куприй<sup>1,2</sup>, Б.И. Краснопольский<sup>1</sup>

<sup>1</sup>Институт механики МГУ имени М.В. Ломоносова,

<sup>2</sup>Факультет ВМК МГУ имени М.В. Ломоносова

Решение разреженных систем линейных алгебраических уравнений (СЛАУ) является одним из распространенных вопросов при моделировании задач математической физики. Реализация вычислительно-эффективных алгоритмов, а также совершенствование их программной реализации являются актуальной проблемой, требующей постоянного внимания в свете развития архитектуры вычислительных систем. Одним из таких вопросов, рассматриваемых в данной работе, является оценка различных форматов представления разреженных матриц, используемых при реализации алгоритмов линейной алгебры.

При решении разреженных СЛАУ значительное время вычислений приходится на выполнение базовой операции умножения матрицы на вектор (SpMV) вида:  $y = Ax$ , где  $A \in \mathbb{R}^{n \times m}$  - разреженная матрица с  $nnz$  ненулевых элементов ( $nnz \ll n \cdot m$ );  $x \in \mathbb{R}^m$ ,  $y \in \mathbb{R}^n$  - плотный вектор.

Особенностью операции SpMV является ее низкая вычислительная интенсивность: на каждую вычислительную операцию сложения или умножения приходится чтение/запись порядка 10 байт данных. Большой трафик данных преобладает над удельным объемом вычислений, поэтому время их выполнения, а значит и эффективность всего алгоритма, ограничиваются, в первую очередь, пропускной способностью шины памяти вычислительной системы [1]. Одним из вариантов повышения эффективности подобных алгоритмов является оптимизация способов хранения данных в памяти, и в частности, форматов представления разреженных матриц.

Среди форматов хранения разреженных матриц широкое распространение получил универсальный формат CSR [2] (Compressed Sparse Row). Он предполагает, что информация о ненулевых элементах матрицы хранится в трех массивах: **Val**, **Row** и **Col**. Размер массивов **Val** и **Col** равен  $nnz$ , а размер массива **Row** равен  $n + 1$ . Массив **Val** хранит значения элементов, массив **Col** хранит индексы столбцов этих элементов, а массив **Row** - информацию о количестве элементов в каждой строке матрицы. Размер целочисленных типов данных, использующихся в двух последних массивах, определяется размером матрицы и количеством ненулевых элементов.

В представленной работе исследуются два модифицированных формата - ICSR (Incremental CSR) [3] и RICSР (Row Incremental CSR). Модификация формата ICSR направлена на избавление от массива **Row**. При этом, вместо массива **Col** с номерами столбцов элементов используется массив приращений **Inc** размера  $nnz + 1$ , содержащий приращения номеров столбцов элементов матрицы. Битность типа данных для хранения элементов массива **Inc** определяется величиной максимального приращения соседних элементов.

Формат RICSР ориентирован на сокращение потребляемой памяти для хранения массива **Col**. Вместо одного массива, содержащего индексы столбцов элементов матрицы, используется два массива **Col\_0** и **Col\_i** размера  $n$  и  $nnz - n$  соответственно. Массив **Col\_0** хранит индекс столбца первого ненулевого элемента строки, тогда как массив **Col\_i** содержит смещения индексов последующих элементов от индекса первого элемента. Выигрыш в

---

\*Работа поддержана грантом РФФ 18-71-10075.

использовании памяти состоит в том, что битность типа данных для массива `Col_i` определяется максимальным смещением последнего элемента строки от первого элемента (*offset*). Поскольку число ненулевых элементов в матрице обычно существенно превышает число столбцов, а ширина ленты матрицы - ограничена (либо могут дополнительно применяться алгоритмы переупорядочения элементов матрицы для сжатия ленты, например [4]), то такая оптимизация может быть практически значимой.

Для рассмотренных форматов представления матриц могут быть получены теоретические оценки времени исполнения операции умножения матрицы на вектор, основанные на объеме считываемых и записываемых данных и пропускной способности шины памяти вычислительной системы. Объем данных для выполнения операции SpMV для рассмотренных форматов составит:

$$\begin{aligned}\Sigma_{CSR} &= F \cdot nnz + P(nnz) \cdot (n + 1) + P(m) \cdot nnz, \\ \Sigma_{ICSR} &= F \cdot nnz + P(2m - 1) \cdot (nnz + 1), \\ \Sigma_{RICSR} &= F \cdot nnz + P(nnz) \cdot (n + 1) + P(m) \cdot n + P(offset) \cdot (nnz - n).\end{aligned}$$

Здесь  $F$  - размер типа данных с плавающей точкой, а  $P(x)$  - размер минимального целочисленного типа данных, в котором может быть размещено число  $x$ . Для тестовой матрицы, соответствующей дискретизации уравнения Пуассона в кубической области на сетке  $150^3$ , получены оценки ожидаемого ускорения операции SpMV, которые составили:  $T_{CSR}/T_{ICSR} \approx 1.05$ ,  $T_{CSR}/T_{RICSR} \approx 1.16$ .

Рассматриваемые выше форматы хранения матриц были реализованы в разработанной библиотеке численных методов для решения систем линейных алгебраических уравнений XAMG [5,6]. Проведенное тестирование показало возможность ускорения операции SpMV и всей процедуры решения СЛАУ в целом за счет использования модифицированных форматов представления матриц. Для указанной тестовой матрицы получено ускорение операции SpMV порядка 5% для формата ICSR, и порядка 12% для формата RICSR, что соответствует теоретическим оценкам. Для реализации метода решения СЛАУ, на основе метода BiCGStab с алгебраическим многосеточным методом в качестве предобуславливателя получено ускорение порядка 3% для формата ICSR, и порядка 13% для формата RICSR, что также соответствует теоретическим оценкам.

## Литература

1. Williams S., Waterman A., Patterson D. Roofline: An Insightful Visual Performance Model for Multicore Architectures // Communications of the ACM. 2009. Vol. 53, No. 4. P. 65–76. DOI: 10.1145/1498765.1498785.
2. Saad Y. Iterative methods for sparse linear systems, 2nd edition. Philadelphia, PA: SIAM, 2003. 528 p. DOI: 10.1137/1.9780898718003.
3. Koster J. Parallel templates for numerical linear algebra, a high-performance computation library, Master's thesis, Utrecht University, Department of Mathematics, July 2002. URL: <https://webspace.science.uu.nl/bisse101/Theses/koster02.pdf>
4. Reid J.K., Scott J.A. Reducing the total bandwidth of a sparse unsymmetric matrix // SIAM Journal on Matrix Analysis and Applications. 2006. Vol. 28, No. 3. P. 805–821. DOI: 10.1137/050629938.
5. Krasnopolsky B., Medvedev A. XAMG: A library for solving linear systems with multiple right-hand side vectors // SoftwareX. 2021. 100695. DOI: 10.1016/j.softx.2021.100695
6. XAMG: source code repository. URL: <https://gitlab.com/xamg/xamg> (дата обращения: 07.04.2021).

## Параллельный алгоритм численного метода моделирования массопереноса двухфазной жидкости в трещиновато-поровом коллекторе

Л.В. Еникеева<sup>1,2</sup>, Р.М. Узьянбаев<sup>2</sup>, А.А. Мазитов<sup>3</sup>, Ю.О. Бобренёва<sup>2</sup>, И.М. Губайдуллин<sup>2,3</sup>

<sup>1</sup>Институт вычислительной математики и математической геофизики Сибирского  
отделения Российской академии наук,

<sup>2</sup>Уфимский государственный нефтяной технический университет,

<sup>3</sup>Институт нефтехимии и катализа УФИЦ РАН

Рассматривается массоперенос в коллекторе трещиновато-порового типа. Математическая модель данного процесса описывается системой дифференциальных уравнений с частными производными. Для численного решения данной системы используется неявная конечно-разностная схема. Для решения уравнений с блочно-тредиагональной матрицей применяется метод скалярной прогонки. Для ускорения трудоемких расчетов рассматривается параллельный алгоритм.

*Ключевые слова:* математическая модель, система дифференциальных уравнений с частными производными, уравнение пьезопроводности, двойная пористость, скалярная прогонка, параллельные алгоритмы.

Трещиновато-пористые коллектора характеризуются интенсивным обменным потоком жидкости между трещинами и пористыми блоками, что вносит определенные коррективы в известные методы определения фильтрационных параметров. Для того чтобы идентифицировать описанные фильтрационные потоки, необходимы такие численные модели, которые бы учитывали также и трещинную составляющую коллектора. Рассматриваются двухфазные уравнения фильтрации жидкости в трещиновато-поровых коллекторах, записанные в дивергентной форме:

$$\begin{aligned} \frac{\partial(\varphi^f \rho_o S_o^f)}{\partial t} + \nabla(\rho_o U_o^f) + \rho_o \cdot \sigma \cdot \lambda_o^m (Pf - Pm) &= \rho_o q_j \\ \frac{\partial(\varphi^f \rho_w S_w^f)}{\partial t} + \nabla(\rho_w U_w^f) + \rho_w \cdot \sigma \cdot \lambda_w^m (Pf - Pm) &= \rho_w q_j \\ \frac{\partial(\varphi^m \rho_o S_o^m)}{\partial t} - \rho_o \cdot \sigma \cdot \lambda_o^m (Pf - Pm) &= \rho_o q_j \\ \frac{\partial(\varphi^m \rho_w S_w^m)}{\partial t} - \rho_w \cdot \sigma \cdot \lambda_w^m (Pf - Pm) &= \rho_w q_j \end{aligned} \quad (1)$$

$$U_o^f = - \frac{k^\alpha k_{ro}(S_o^\alpha)}{\mu_o} \text{grad } P_o^\alpha \quad (2)$$

$$U_w^f = - \frac{k^\alpha k_{rw}(S_w^\alpha)}{\mu_w} \text{grad } P_w^\alpha \quad (3)$$

$$\lambda_i^\alpha = \frac{k^m \cdot k_{ri}(S_i^m)}{\mu_i}, \quad (4)$$

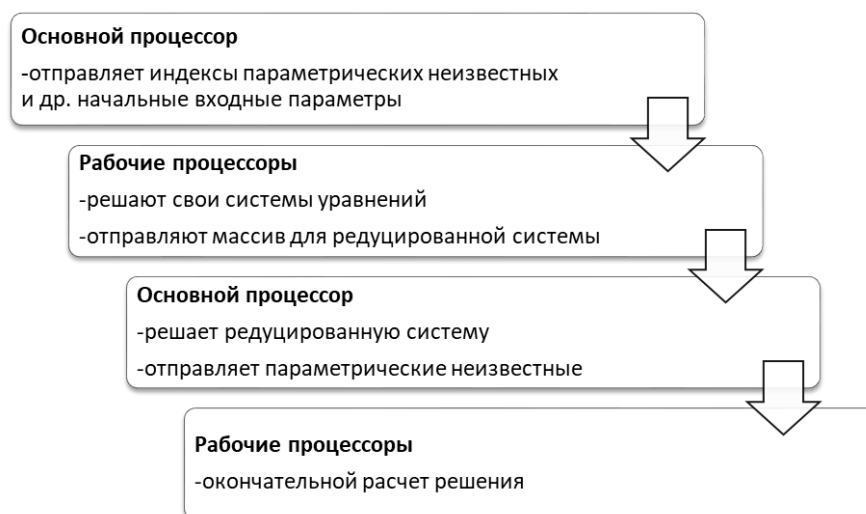
$$\sigma = \frac{12}{L_m^2}, \quad L = a$$

где  $\alpha = f, m$ , где  $f$  – система трещин,  $m$  – система матриц,  $i = o, w$ , где  $o$  – нефть,  $w$  – вода,  $Pf$  – пластовое давление в сети трещин (МПа),  $Pm$  – пластовое давление в матрице (МПа),  $\varphi^\alpha$  –

пористость (д.ед),  $k^{\alpha}$  – абсолютная проницаемость ( $\text{м}^2$ ),  $k_{rw}$ ,  $k_{ro}$  – относительные фазовые проницаемости ( $\text{м}^2$ ),  $\mu_i$  – вязкость ( $\text{Па}\cdot\text{с}$ ),  $q_j$  – дебит жидкости ( $\text{м}^3/\text{сут}$ ),  $S_i^{\alpha}$  – насыщенность,  $\rho_i$  – плотность ( $\text{г}/\text{м}^3$ ),  $U_i^{\alpha}$  – скорость течения фазы,  $\sigma$  – коэффициент трещиноватой породы ( $1/\text{м}^2$ ),  $L$  – размер блоков ( $\text{м}^2$ ),  $a$  – длина стороны блока матрицы ( $\text{м}$ ).

Применение прогонки к решению разностной задачи (1)-(4) приводит к двум основным трудностям: неэкономичностью по числу действий (т.е. большое время счета) и, главным образом, необходимостью в больших ресурсах машинной памяти. Если в расчете матрица имеет большую размерность, то необходимо выделять больше внутренней памяти, а это в свою очередь приводит к увеличению времени расчета.

Для распараллеливания расчетов использовалась технология MPI. Редуцированная система вся решается на одном, управляющем всем счетом 0-процессоре (Rank=0). Также основной процессор решает одну из систем рабочего процессора, что увеличивает степень параллелизма и уменьшает общее время счета. На рисунке 1 описано взаимодействия процессоров.



**Рис. 1.** Схема обмена между процессорами и выполняемой ими работы

Таким образом, в работе представлена математическая модель фильтрации двухфазной жидкости в коллекторе трещиновато-порового типа. Для решения данной задачи выбран метод скалярной прогонки. Продемонстрирована схема обмена между процессорами при распараллеливании скалярной прогонки.

Исследование выполнено за счет гранта Российского научного фонда (проект № 21-71-20047).

## Литература

1. Денк С.О. Проблемы трещиноватых продуктивных объектов. Пермь: Электронные издательские системы, 2004. 334 с.
2. Самарский А.А., Гулин А.В. Численные методы. М.: Наука, 1989 г.
3. Клочков М.А., Марков К.Ю., Митрохин Ю.С., Чиркова Л.С. Организация параллельных вычислений для решения дифференциальных уравнений на blade-сервере: учеб.-метод. Пособие.Ижевск: Изд-во «Удмуртский университет», 2011.



## Повышение производительности расчетов *ab initio* молекулярной динамики при помощи GPU

А.А. Елисеев, Н. Г. Андреади, А.А. Митрофанов

Химический факультет МГУ имени М.В. Ломоносова

Метод *ab initio* молекулярной динамики является мощным инструментом в вычислительной химии и науках о материалах. Особенно часто данный метод применяется в областях, связанных с разработкой новых лекарственных препаратов и функциональных материалов, а также компьютерном моделировании их свойств. Однако при применении данного подхода максимальный размер рассчитываемой системы оказывается жестко ограничен доступными вычислительными ресурсами. Использование графического процессора (GPU) - один из самых мощных способов повышения производительности программного обеспечения. Однако использование GPU связано и с рядом трудностей, наиболее важной из которых, способной значительно снизить производительность или сделать применение данного метода невозможным, является необходимость частого переноса данных между участками кода CPU и GPU. По данной причине, на текущий момент, крайне малая доля квантово-химических пакетов (особенно с открытым кодом) поддерживает работу с GPU. В данной работе проводилось профилирование и переработка кода программного пакета с открытым кодом *qball* [1] (модифицированная версия *qbox* [2], способного к обработке задач как молекулярной динамики с использованием DFT, так и задач расчетов типа TDDFT) как для оценки возможности использования GPU, так и для повышения производительности.

Первым этапом работы стало определение текущей производительности пакета при использовании OMP и MPI технологий параллелизма отдельно. Для оценки использовалась система на основе двух процессоров Intel Xeon E5-2680 v2 и графического ядра NVidia GeForce 2080. Пиковая производительность для операций FP64 составляет 200 GFLOPS и 400 GFLOPS соответственно. Для расчета были выбраны системы, состоящие из 50, 100 или 300 молекул воды, для оценки использовалось время выполнения 10 циклов SCF для задач молекулярной динамики и TDDFT.

Таблица 1. Результаты оценки производительности пакета для циклов SCF MD

Условия теста		Относительное время выполнения			
Тип параллелизма	Количество молекул воды	Количество потоков			
		1	5	10	20
OMP	50	1	0,736	0,786	0,918
	100	1	0,422	0,348	0,395
	300	1	0,378	0,280	0,300
MPI	50	1	0,279	0,219	0,186
	100	1	0,288	0,191	0,138
	300	1	0,240	0,154	0,112

Для задач молекулярной динамики, в случае единичного потока, OMP оказывается незначительно быстрее, что, вероятно, связано с затратами на инициализацию MPI. Однако OMP дает прирост производительности только до 10 потоков, в то время как MPI обеспечивает ускорение вплоть до максимально доступных в данном случае 20 потоков.

Таблица 2. Результаты оценки производительности пакета для циклов SCF TDDFT

Условия теста		Относительное время выполнения			
Тип параллелизма	Количество молекул воды	Количество потоков			
		1	5	10	20
OMP	50	1	0,688	0,806	0,728

	100	1	0,340	0,260	0,286
	300	1	0,277	0,188	0,190
MPI	50	1	0,205	0,144	0,114
	100	1	0,237	0,146	0,098
	300	1	0,221	0,122	0,072

Для задач TDDFT наблюдается картина аналогичная предыдущей, однако все времена расчетов увеличиваются примерно в 2 раза. Таким образом, по результатам тестирования было установлено, что наиболее оптимальной технологией параллелизма для циклов SCF и в случае молекулярной динамики, и в случае TDDFT является MPI.

Для переноса кода на GPU в первую очередь необходимо провести ряд расчетов с использованием средств профилирования кода для выявления узких мест и возможных участков кода, пригодных для оптимизации. По результатам профилирования было выявлено две функции (*SlaterDet::compute\_density* и *SlaterDet::rs\_mul\_add*), при выполнении которых тратится наибольшее количество времени, причем данные функции активны во всех типах задач. Соотношение нагрузки функций меняется в зависимости от типа расчета, однако суммарный процент времени, затраченного на выполнение данных участков кода и в то и в другом случае превышает 50%. Данные функции также используют процедуры быстрого преобразования Фурье (FFT), которые могут выполняться как при использовании встроенных в пакет процедур, так и за счет сторонних библиотек. Это дает возможность использования библиотеки *cufft* для данных операций, что теоретически может значительно повысить производительность, особенно в случае больших задач. Более того, в данных функциях содержатся циклы с большим количеством операций, распараллеливаемых по технологии OpenMP, что дает возможность простого переноса кода на GPU при помощи директив OpenACC.

На данный момент реализован перенос на GPU участков функций, выполняющих преобразование Фурье (при помощи библиотеки *cufft*) и участка функции *SlaterDet::compute\_density* с оптимизацией операций переносов памяти. Данная функция наиболее активно используется в задачах молекулярной динамики, соответственно, на для данного типа расчетов было проведено сравнение на примере, содержащем 300 молекул воды. При сравнении с единичным потоком наблюдается ускорение выполнения тестовой задачи на 20%. В рамках продолжения работы планируется реализация многопоточного гибридного решения MPI/GPU, позволяющего достичь наибольшей производительности.

Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова. Работа выполнена при поддержке гранта РФФИ № 19-73-20115.

## Литература

1. Qball software [Electronic resource]. URL: <http://github.com/LLNL/qball>.
2. Gygi F. Architecture of Qbox: A scalable first-principles molecular dynamics code // IBM J. Res. Dev. 2008. Vol. 52, № 1.2. P. 137–144.

## Разработка распределенных алгоритмов управления системами роевого интеллекта с использованием методов встроенной эволюции\*

А.Г. Николашкин, Н.М. Ершов

Московский государственный университет им. М.В. Ломоносова, факультет вычислительной математики и кибернетики

Групповая робототехника представляет собой перспективное направление в области роевого интеллекта, заключающееся в построении робототехнических систем, состоящих из большого числа одинаковых и относительно просто устроенных роботов. Актуальной задачей в этой области является разработка распределенных алгоритмов управления такого рода системами. Проблема заключается в том, что целью создания алгоритма является некоторое желаемое *коллективное* поведение роя роботов, в то время как реально должно «программироваться» *одинаковое индивидуальное* поведение отдельных роботов роя.

В настоящей работе предполагается, что в качестве системы управления роботов служит искусственная рекуррентная нейронная сеть, на входы которой подаются сигналы различных датчиков робота, а ее выходы определяют параметры команд управления действиями этого робота. Рекуррентность сети позволяет запоминать ее внутреннее состояние и использовать его для более эффективного управления роботом. При такой организации системы управления под программированием робота понимается настройка топологии и весов связей соответствующей нейронной сети, т.е. задача построения алгоритма управления сводится к задаче обучения некоторой рекуррентной нейронной сети.

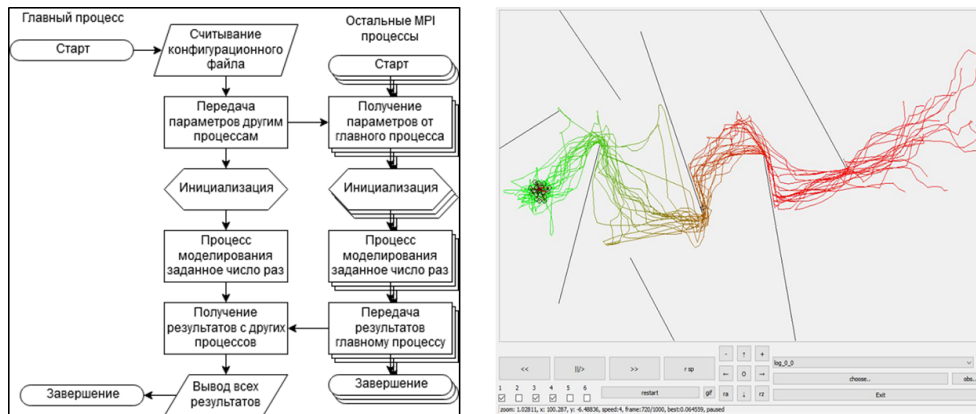


Рис. 1. Организация и интерфейс моделирующей системы

В настоящей работе предлагается использовать для обучения нейронных сетей эволюционный подход на основе методов встроенной эволюции (embodied evolution, [1]). Данный подход означает, что эволюция программ (т.е. нейронных сетей, управляющих роботами) происходит в режиме онлайн, т.е. непосредственно при решении роем роботов поставленной перед ним задачи. Соответственно, каждый робот действует по своему собственному алгоритму, который постепенно эволюционирует за счет взаимодействия с другими роботами в рое. Преимуществом такого подхода по сравнению с традиционным эволюционным подходом помимо его *универсальности* является то, что поведение роя, во-первых, оказывается *адаптивным*, т.е. при относительно плавном изменении внешней среды (в частности, при переходе от виртуальной модели к реальному миру) рой будет успевать менять свое поведение. Во-вторых, такая организация эволюции теоретически может приводить к более

\*Работа выполнена при финансовой поддержке РФФИ (грант № 20-07-01053 А).

сложно устроенным системам управления, например, к построению *гетерогенных* систем, в которых разными роботами решаются разные локальные задачи (как это происходит, например, в колониях муравьев или пчел).

Для сравнения и тестирования различных алгоритмов и методов встроенной эволюции в работе использовались три классические модельные задачи роевой робототехники: задача поиска источника сигнала (например, источника задымления), задача мониторинга и задача развертывания. Компьютерные эксперименты проводились на вычислительной системе IBM Polus [2] с помощью специально разработанной моделирующей системы (рис. 1), реализованной на языке программирования C++ с использованием параллельных вычислений на основе технологий MPI и OpenMP.

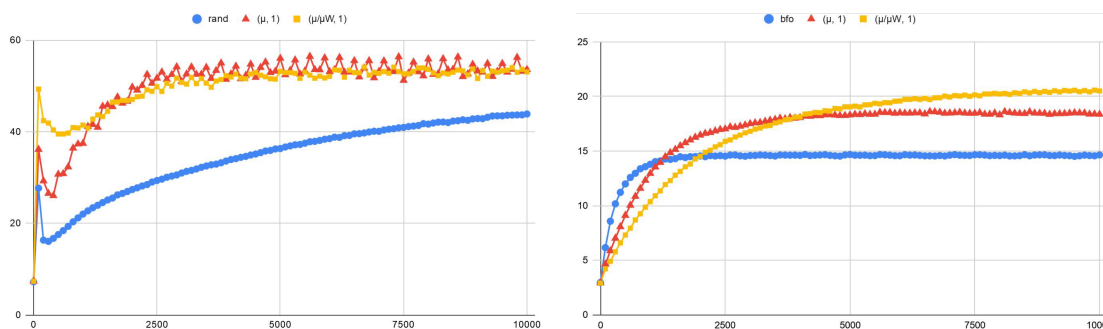


Рис. 2. Сравнение работы методов встроенной эволюции и адаптированных методов роевой оптимизации на примере решения задач мониторинга и развертывания

Был проведен анализ нескольких алгоритмов embodied evolution, исходя из результатов этого анализа для дальнейших исследований был выбран алгоритм  $(\mu/\mu W, 1)$ -On-line EEA [3], для которого было предложено расширение, позволяющее настраивать не только веса связей в нейронной сети, но и их топологию с использованием технологии odNEAT [4]. Сравнение эффективности предложенных вариаций методов встроенной эволюции (рис. 2), в том числе с реализованными ранее методами роевой оптимизации, адаптированными для решения задач роевой робототехники [5], показало работоспособность и перспективность предложенной технологии разработки распределенных алгоритмов управления роевыми робототехническими системами.

## Литература

1. Bredeche N., Haasdijk E., Prieto A. Embodied Evolution in Collective Robotics: A Review // *Frontiers in Robotics and AI*. 2018, Vol. 5, P. 1–15. DOI: 10.3389/frobt.2018.00012.
2. Вычислительный комплекс IBM Polus [Электронный ресурс] // Электронный ресурс. URL: <http://hpc.cs.msu.ru/polus> (дата обращения: 11.04.2021).
3. Boumaza A. Introducing Weighted Intermediate Recombination in On-Line Collective Robotics, the  $(\mu/\mu W, 1)$ -On-line EEA // In: Kaufmann P., Castillo P. (eds) *Applications of Evolutionary Computation*. Lecture Notes in Computer Science, 2019, Vol. 11454. DOI: 10.1007/978-3-030-16692-2\_42.
4. Silva F., Urbano P., Oliveira S., Christensen A. odNEAT: An Algorithm for Distributed Online, Onboard Evolution of Robot Behaviours // *Proceedings of the ALIFE 2012, East Lansing, Michigan*. 2012, P. 251–258. DOI: 10.7551/978-0-262-31050-5-ch034.
5. Николашкин А.Г., Ершов Н.М. Адаптация алгоритмов роевой оптимизации к решению поисковых задач роевого интеллекта // *Системный анализ в науке и образовании*. 2019. № 3. С. 50–55.

## Распараллеливание задачи распространения динамических волновых возмущений в трещиноватых геологических средах

*В.С. Саган, Н.И. Хохлов*

Московский физико-технический институт

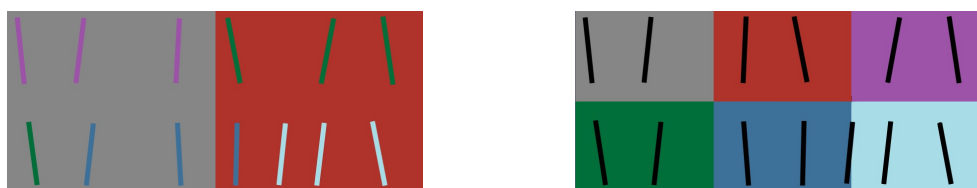
Задача распространения динамических волновых возмущений, в геологических средах с наличием трещиноватых неоднородностей, является одной из актуальных задач в прямом сейсмическом моделировании при проведении сейсморазведочных работ при поиске углеводородов [1]. Для моделирования сейсмических волн в трещиноватых средах разработаны ряд численных методов, в том числе [1,3] и моделей, например модель Шойнберга [2]. При расчёте такого рода сред требуется учёт влияния трещин на сейсмический отклик. Размер единичной трещины намного меньше размеров расчётной области. Размеры трещин могут достигать единиц и десятков метров, а расчётной области единицы километров. Для моделирования таких разномасштабных задач применяются как методы на неструктурных [4], так и на структурных сетках [3]. Структурные расчётные сетки имеют ряд преимуществ перед неструктурными, в том числе простота алгоритмов, отсутствие необходимости построения сетки и скорость расчёта [5]. Последнее очень важно ввиду разномасштабности и большой вычислительной сложности задачи. Однако они имеют и недостатки. Основной из них – невозможность задания сложных конфигураций трещин. Ранее уже был предложен ряд алгоритмов расчета разнонаправленных и субвертикальных трещин на структурных сетках [3, 6]. В работе [6] рассматривается подход на «Химерных» или наложенных сетках. Вокруг трещины строится небольшая структурная сетка, сонаправленная с ориентацией трещины. Затем она накладывается на основную сетку, которая описывает геологический массив. Между сетками используется специальное условие сшивки, для расчёта применялся сеточно-характеристический метод [1,3]. Предложенный подход позволяет достаточно просто задавать сложные конфигурации трещин на структурных сетках. В данной работе рассматривается вопрос распараллеливания метода, использующего наложенные сетки на высокопроизводительных вычислительных системах с распределённой памятью.

Вопрос распараллеливания усложняется наличием множества структурных сеток и связями между ними. При этом размеры сеток сильно отличаются. В работе рассматриваются алгоритмы декомпозиции сеток между процессами в двумерной среде с несколькими трещиноватыми слоями. В рассматриваемой модели 1000 трещин, каждой соответствует наложенная сетка. Отклонение трещин от вертикали составляет не более 5 градусов. У поверхности среды размещается источник возмущения.



Волновой фронт после прохождения волны от источника через среду

Были разработаны два алгоритма, пример работы которых видно на картинке. Каждому процессу соответствует один цвет.

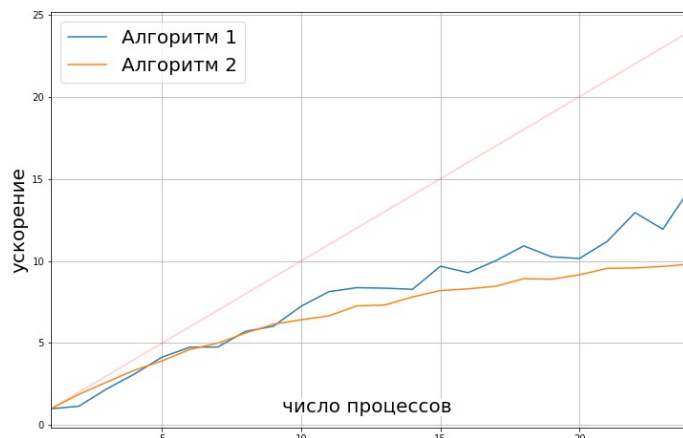


Пример декомпозиции сеток двумя алгоритмами

Первый алгоритм является «жадным» алгоритмом, сначала считается общее количество узлов во всех сетках, причём наложенные сетки учитывают с коэффициентом, который равен отношению времён просчёта одного узла наложенной сетки и одного узла основной сетки и подбирается эмпирически. Затем алгоритм распределяет работу между процессами так, чтобы каждый процесс считал примерно равное число узлов. Этот алгоритм является доработкой алгоритма, предложенного в работе [7].

Второй алгоритм разбивает основную сетку на области по всем процессам равномерно, а каждую сетку трещины отдаёт тому процессу, в области которого она лежит. В основе алгоритма лежит принцип геометрического параллелизма [8].

Для сравнения алгоритмов были выполнены расчёты на числе процессов от 1 до 24.



Зависимость ускорения от числа процессов для рассматриваемых алгоритмов

В перспективе планируется исследование поведения алгоритмов декомпозиции в трёхмерных трещиноватых средах и исследование на большем числе потоков.

## Литература

1. Петров И. Б., Муратов М. В. Применение сеточно-характеристического метода в решении прямых задач сейсморазведки трещиноватых пластов (обзорная статья) //Математическое моделирование. – 2019. – Т. 31. – №. 4. – С. 33-56.
2. Schoenberg M. Elastic wave behavior across linear slip interfaces //The Journal of the Acoustical Society of America. – 1980. – Т. 68. – №. 5. – С. 1516-1521.
3. Khokhlov N, Stognii P. Novel Approach to Modeling the Seismic Waves in the Areas with Complex Fractured Geological Structures. *Minerals*. 2020; 10(2):122. DOI:[10.3390/min10020122](https://doi.org/10.3390/min10020122)
4. Bosma S. et al. Multiscale finite volume method for discrete fracture modeling on unstructured grids (MS-DFM) //Journal of Computational Physics. – 2017. – Т. 351. – С. 145-164.
5. Бирюков В. А. и др. Моделирование распространения упругих волн в геологической среде: сравнение результатов трех численных методов //Журнал вычислительной математики и математической физики. – 2016. – Т. 56. – №. 6. – С. 1104-1114.
6. Ruzhanskaya A., Khokhlov N. Modelling of fractures using the Chimera grid approach //2nd Conference on Geophysics for Mineral Exploration and Mining. – European Association of Geoscientists & Engineers, 2018. – Т. 2018. – №. 1. – С. 1-5. DOI: [10.3997/2214-4609.201802730](https://doi.org/10.3997/2214-4609.201802730)
7. Fofanov V., Khokhlov N. Optimization of Load Balancing Algorithms in Parallel Modeling of Objects Using a Large Number of Grids //Russian Supercomputing Days. – Springer, Cham, 2020. – С. 63-73. DOI:[10.1007/978-3-030-64616-5\\_6](https://doi.org/10.1007/978-3-030-64616-5_6)

8. Яковлевский М.В. Введение в параллельные методы решения задач: Учебное пособие / Предисл.: В. А. Садовничий. – М.: Издательство Московского университета, 2013. – 328 с., илл. – (Серия «Суперкомпьютерное образование»)

## Расчет поляризационных матриц рассеяния фрактальных кластерных частиц атмосферного аэрозоля Титана

М. П. Черешенков<sup>1</sup>, Я. А. Илюшин<sup>1,2</sup>

<sup>1</sup>Московский государственный университет имени М. В. Ломоносова, <sup>2</sup>Институт радиотехники и электроники им. В. А. Котельникова РАН

Аэрозоль атмосферы Титана представляет собой фрактальные агрегаты, образованные за счёт взаимодействия друг с другом продуктов разложения метана и азота под действием фотонов солнца и энергетических электронов от магнитосферы Сатурна. В результате получается множество кластеров, каждый из которых достаточно уникален по своей природе. Такая уникальность может вызывать некоторые неточности в анализе данных. Если стоит цель рассчитать оптические характеристики на агрегатах определённого размера, то могут быть получены результаты, не определяющие всю совокупность таких агрегатов, а лишь частный случай. Поэтому возникает вопрос: "Представляют ли усреднённые характеристики рассеяния агрегатов, одного размера и образованных одним и тем же методом, свойства ансамбля случайных агрегатов?". Этим же вопросом задаются и в работе [1, 2], что даёт этому вопросу актуальность. Главным фактором, требующего учёта, является отличие результатов усреднения для различных методов. На данный момент наиболее известные методы формирования фрактальных частиц делятся на три вида: ВССА [3] (с англ. Баллистическая кластер-кластер агрегация), ВРСА (с англ. Баллистическая частица-кластер агрегация), DLA [4] (с англ. Агрегация ограниченная диффузией). Все три метода имеют уникальные алгоритмы, а также каждая агрегация характерна для разных высот атмосферы. ВРСА и DLA являются более правдоподобными для низких слоёв атмосферы, так как являются более плотными по своей структуре, а вот ВССА наоборот, является подходящим для верхних слоёв, что обусловлено её пористой структурой. Для получения координат таких агрегатов был написан параллельно-вычислительный код на языке C++, что позволило ускорить процесс генерации в несколько раз, относительно последовательного вычисления.

Также нужно отметить, что для расчёта поляризационных матриц требуется информация о комплексных показателях преломления. Экспериментальные данные, описывающие вещество, схожего по химическому составу с толинами, взяты из работы [5].

В настоящей работе проводится усреднение двух первых элементов матрицы рассеяния,

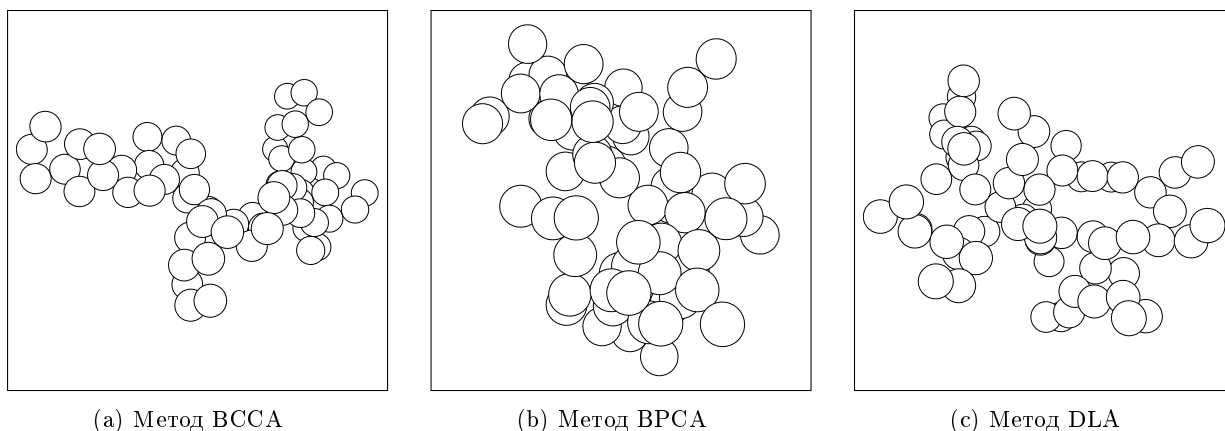


Рис. 1: Фрактальные частицы, полученные различными методами агрегации: (а) Баллистическая кластер-кластер агрегация; (б) Баллистическая частица-кластер агрегация; (с) Агрегация ограниченная диффузией.



так как они представляют наибольший интерес ( $S_{11}$  - интенсивность рассеянного света,  $-S_{12}/S_{11}$  - линейная поляризация). Усреднение проходит по фиксированному количеству агрегатов для каждого количества частиц. Вместе с усреднёнными значениями находится стандартное отклонение для оценки точности усреднённых данных.

Для расчёта поляризационных матриц рассеяния используется реализация метода T-матриц, созданная Мищенко и Маковски [6]. Этот код написан как для серийных, так и для параллельных вычислений. В данной работе используются результаты параллельного вычисления на серийном устройстве с многоядерным процессором.

Анализ данных показал, что наблюдается увеличение точности усреднённых значений с ростом количества частиц для двух определённых методов, DLA и ВРСА. Такое поведение связано с нарушением симметрии агрегатов с ростом частиц для метода ВССА. Также выявлен "пик неточности" для интенсивности, расположение которого зависит от количества частиц и структуры исследуемого агрегата.

Что касается линейной поляризации, то полученные результаты не совпали с данными, полученными в работе Колоколовой и др. [7]. Расхождение данных объясняется тем, что в настоящей работе было проведено усреднение агрегатов, имеющих разные фрактальные размерности. Помимо этого, было выявлено, что точность линейной поляризации растёт с ростом частиц, причём для агрегатов большего размера отклонение линейной поляризации стремится быть одинаковым на всех углах рассеяния.

## Литература

1. Yu.V. Skorov, H.U. Keller, A.V. Rodin, Optical properties of aerosols in Titan's atmosphere, *Planetary and Space Science*, Volume 56, Issue 5, 2008, Pages 660-668, DOI: 10.1016/j.pss.2007.11.013.
2. Yu.V. Skorov, H.U. Keller, A.V. Rodin, Optical properties of aerosols in Titan's atmosphere: Large fluffy aggregates, *Planetary and Space Science*, Volume 58, Issues 14-15, 2010, Pages 1802-1810, DOI: 10.1016/j.pss.2010.08.002.
3. Sota Arakawa, Masaki Takemoto, Taishi Nakamoto, Geometrical structure and thermal conductivity of dust aggregates formed via ballistic cluster-cluster aggregation, *Progress of Theoretical and Experimental Physics*, Volume 2019, Issue 9, September 2019, 093E02, DOI: 10.1093/ptep/ptz102
4. Halsey, Thomas, Diffusion-Limited Aggregation: A Model for Pattern Formation, *Physics Today*, 53, 2000, DOI: 10.1063/1.1333284.
5. S.I. Ramirez, P. Coll, A. da Silva, R. Navarro-González, J. Lafait, F. Raulin, Complex Refractive Index of Titan's Aerosol Analogues in the 200-900 nm Domain, *Icarus*, Volume 156, Issue 2, 2002, Pages 515-529, DOI: 10.1006/icar.2001.6783.
6. Mackowski, Daniel, Calculation of the T matrix and the scattering matrix for ensembles of spheres, *Journal of The Optical Society of America A-optics Image Science and Vision*, 13, 1996, DOI: 10.1364/JOSAA.13.002266.
7. Ludmilla Kolokolova, Hiroshi Kimura, Klaus Ziegler, Ingrid Mann, Light-scattering properties of random-oriented aggregates: Do they represent the properties of an ensemble of aggregates?, *Journal of Quantitative Spectroscopy and Radiative Transfer*, Volume 100, Issues 1-3, Pages 199-206, 2006, DOI: 10.1016/j.jqsrt.2005.11.038.

## Реализация метода FDTD с граничными условиями PML с использованием вычислений в смешанной точности<sup>1</sup>

А. Н. Арисова<sup>1</sup>, В. Д. Волокитин<sup>1</sup>, Е. С. Ефименко<sup>2</sup>, И. Б. Мееров<sup>1</sup>

Нижегородский государственный университет им. Н. И. Лобачевского<sup>1</sup>  
Институт прикладной физики РАН<sup>2</sup>

Производители микропроцессоров выпускают на рынок новые устройства, аппаратно поддерживающие вычисления с плавающей запятой в половинной точности. Несмотря на то, что такая поддержка на данный момент уступает одинарной и двойной точности, соответствующие наборы команд развиваются, а в средствах программирования появляются реализации новых типов данных. Побуждающим фактором для развития этого направления является возможность сокращения затрат памяти и времени счета при тренировке и последующем использовании искусственных нейронных сетей [1], однако вопрос организации вычислений в половинной точности для классического численного моделирования также представляет большой интерес.

Ограничения на диапазоны чисел и число значащих цифр после запятой в числах половинной точности на первый взгляд выглядят непреодолимым препятствием для использования в численном моделировании. Однако широко известные приемы [2, 3], связанные с масштабированием, переупорядочением операций, аккуратными способами суммирования, а также выполнением «чувствительных» этапов расчета в стандартной точности дают надежду на получение приемлемых для использования результатов моделирования с меньшими затратами [4, 5].

В данной работе этот вопрос изучается в контексте решения уравнений Максвелла методом конечных разностей во временной области (FDTD) с поглощающими граничными условиями в форме идеально согласованного слоя (PML) [6]. PML-слой представляет собой граничную область толщиной 6-8 ячеек с плавно нарастающими вдоль направления распространения магнитной и электрической проводимостями  $\sigma(x) = \sigma_0(x/\delta)^n$ , где  $\sigma_0$  – максимальное значение проводимости,  $\delta$  – толщина слоя,  $n > 1$  – показатель степени зависимости. Использование подобного профиля проводимости вместе с модификацией уравнений Максвелла в слое [6], позволяет получить затухание энергии импульса  $< 10^{-6}-10^{-8}$  и является стандартным методом вычислительной электродинамики. Ранее [7] мы реализовали метод FDTD [6] и провели анализ ошибок в задаче с известным аналитическим решением, сравнивая расчеты в разных режимах точности. В данной работе мы реализовали PML, допуская согласованное использование разных режимов точности во внутренней части расчетной области и в PML-слое. Нас интересовал вопрос о том, какого минимального уровня отражения удастся добиться, настраивая параметры PML  $\sigma_0$  и  $n$  при фиксированной толщине слоя  $\delta$  и комбинируя разные режимы точности.

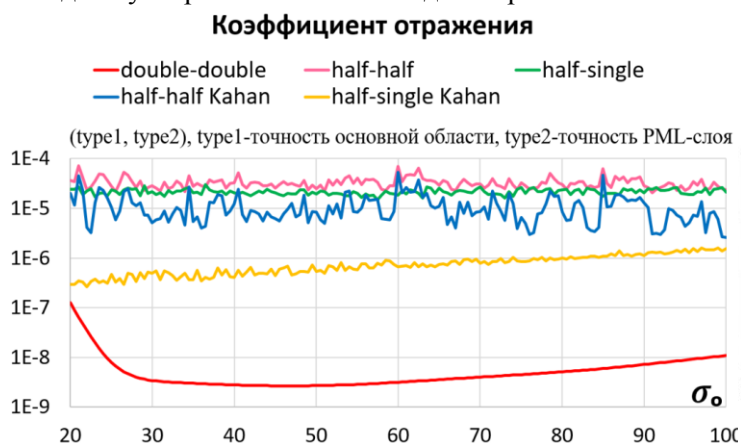
В качестве тестовой задачи рассматривалось нормальное падение гауссова лазерного импульса на границу PML. В силу линейности задачи для оценки качества работы PML-слоя использовался коэффициент отражения импульса по энергии. Были проведены эксперименты с использованием разных типов точности, а также в смешанном режиме. Результаты показали, что в данной задаче вычисления в одинарной точности приводят к приемлемым результатам, тогда как при переходе к половинной точности возникает проблема потери значащих цифр мантииссы при суммировании. Для преодоления этого эффекта, как и ранее [7], был задействован алгоритм компенсации ошибок суммирования Кэхэна [2, 3]. Обнаружено, что коэффициенты обновления полей в слое PML необходимо вычислять в более высоком режиме точности, а порядок операций может оказывать влияние на результаты расчетов в пониженной точности.

На рис. 1 представлены достигнутые значения коэффициента отражения при интегрировании на двумерной сетке при разных значениях параметра PML  $\sigma_0$ . В связи с тем, что результаты, полученные в двойной точности и в одинарной точности, почти неотличимы, последние опущены, чтобы не загромождать график. При вычислениях в половинной точности использование описанных выше модификаций позволяет получить значения коэффициента отражения  $\sim 10^{-5}-10^{-6}$ , что значительно хуже, чем в двойной точности ( $\sim 10^{-9}$ ), однако, все еще приемлемо

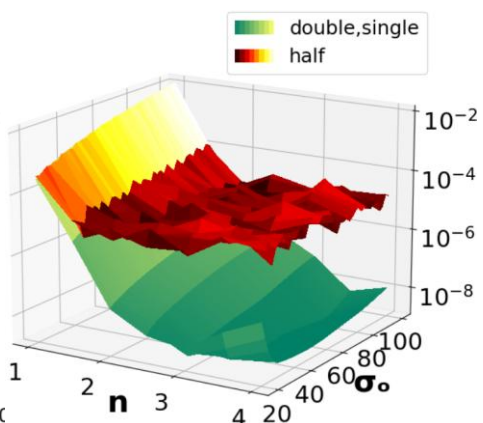
<sup>1</sup> Работа выполнена при поддержке гранта РФФИ и Правительства Нижегородской обл. №18-47-520001.

для широкого класса задач. Наиболее перспективные для практики результаты получены при вычислениях в *смешанной точности*, когда расчеты во внутренней области ведутся в половинной, а в области PML – в одинарной точности. В этом случае удается достичь коэффициента отражения  $\sim 10^{-6}$ - $10^{-7}$ . При этом метод чувствителен к значениям параметров  $n$  и  $\sigma_0$  (рис. 2).

Замеры производительности производились на компьютере Intel Endeavor (2x Intel Xeon Platinum 8260L, 2.40 GHz, 48 ядер, 192 ГБ оперативной памяти). Эффективность распараллеливания в рамках одного сокета во всех реализациях составляет порядка 85%, при задействовании второго сокета эффективность падает до 70%. При этом время работы в двойной и одинарной точности является сопоставимым, поскольку пока не проводилось оптимизаций, направленных на векторизацию кода. При использовании схемы Кэхэна происходит замедление в 1.7 раз относительно варианта без компенсации суммирования. Половинная точность проигрывает в производительности в 2.5 раза, так как на данный момент отсутствует полная аппаратная поддержка данного типа в CPU и для ее эмуляции используется сторонняя библиотека Half [8], которая реализует половинную точность, используя для вычислений одинарную. Отметим, что мы не производили дополнительные оптимизации, направленные на ускорение кода, так как основной целью было проверить саму возможность использования расчетов в пониженной точности. Далее мы планируем проверить работоспособность рассмотренного подхода при решении нелинейных задач электродинамики, а также провести дополнительные оптимизации кода для высокопроизводительных вычислений и портировать код на язык DPC++, чтобы провести эксперименты на платформах с аппаратной поддержкой половинной точности. Учитывая, что на таких платформах ускорение вычислений в половинной точности доходит до 10 раз, мы ожидаем ускорения вычислений даже при использовании схемы суммирования Кэхэна.



**Рис. 1.** Коэффициент отражения системы при численном интегрировании уравнений Максвелла методом FDTD с ГУ PML. Задача о распространении лазерного импульса в вакууме. Сетка 128 x 16.



**Рис. 2.** Зависимость коэффициента отражения системы от параметров PML  $n$  и  $\sigma_0$ .

## Литература

1. Courbariaux M., Bengio Y., David J. P. Training deep neural networks with low precision multiplications // arXiv preprint arXiv:1412.7024. – 2014.
2. Muller J. M. et al. Handbook of floating-point arithmetic. – Boston, MA : Birkhäuser, 2018.
3. Higham N. J. Accuracy and stability of numerical algorithms. – SIAM, 2002.
4. Haidar A. et al. Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers // SC18. – IEEE, 2018. – С. 603-613.
5. Henry G., Tang P. T. P., Heinecke A. Leveraging the Bfloat16 artificial intelligence datatype for higher-precision computations // 2019 ARITH. – IEEE, 2019. – С. 69-76.
6. Taflove A., Hagness S.C. Computational electrodynamics: the finite-difference time-domain method, 3rd ed. – Norwood, MA: Artech House, 2005.
7. Арисова А. Н., Волокитин В. Д., Мееров И. Б. Реализация метода конечных разностей во временной области с использованием вычислений в двойной, одинарной и половинной точности // Суперкомпьютерные дни в России. – 2020. – С. 137-139.

8. Half-precision floating-point library. URL: <http://half.sourceforge.net>

## Реализация параллельных вычислений в рамках программной системы поддержки эволюционных и роевых методов оптимизации\*

А.А. Тимачев, Н.М. Ершов

Московский государственный университет им. М.В. Ломоносова, факультет вычислительной математики и кибернетики

Настоящая работа посвящена вопросам автоматического распараллеливания вычислений в рамках программной библиотеки *Insectae*, предназначенной для решения задач непрерывной и дискретной оптимизации с использованием эволюционных и роевых алгоритмов. Данная библиотека в настоящее время поддерживает несколько наиболее популярных методов оптимизации [1], в том числе метод имитации отжига, генетические алгоритмы, алгоритм дифференциальной эволюции, метод роя частиц и т.д. Помимо базовых версий указанных алгоритмов библиотека поддерживает большое число их вариаций, например, за счет использования альтернативных реализаций операторов данных алгоритмов, а также за счет настройки их числовых параметров, большинство из которых являются вычисляемыми, что позволяет использовать различные *адаптивные* схемы [2] выполнения реализованных в библиотеке алгоритмов.

Все алгоритмы в библиотеке *Insectae* реализованы по одной общей схеме: поиск глобального экстремума целевой функции в них выполняется с помощью популяции особей, каждая из которых представляет собой некоторое пробное решение рассматриваемой задачи; на каждой итерации основного цикла алгоритма происходит последовательное применение различных операторов к текущей популяции. На основе предварительно проведенного анализа были выделены семь различных типов (паттернов) возможных схем выполнения указанных операторов:

- **evaluate** — вычисление целевой функции всеми особями популяции;
- **foreach** — применение заданного оператора ко всем особям;
- **neighbors** — применение заданного оператора ко всем смежным парам особей;
- **pairs** — применение заданного оператора ко всем смежным парам в двух популяциях;
- **pop2ind** — редуцирование заданной популяции в параметры заданной особи;
- **signals** — вычисление сигналов между особями на основе матрицы расстояний;
- **reduce** — редуцирование популяции в глобальные значения (параметры среды).

Особенностью данных паттернов является то, что все они просто и эффективно распараллеливаются. С учетом того факта, что практически вся вычислительная работа рассматриваемых алгоритмов сосредоточена именно в перечисленных паттернах, у нас появляется возможность распараллеливания этих алгоритмов в полностью автоматическом режиме с хорошей итоговой эффективностью. В табл. 1 приведен процент времени работы каждого из перечисленных паттернов для стандартных версий семи реализованных в библиотеке алгоритмов с одинаковыми размером популяции и числом поколений, примененных к одной и той же задаче непрерывной оптимизации. В последнем столбце таблицы в качестве примера показано предполагаемое ускорение, которое можно получить при распараллеливании всех используемых заданным алгоритмом паттернов (за исключением паттерна **reduce**) с использованием  $p = 10$  параллельных процессов (поток).

С учетом проведенного анализа описанных паттернов предполагается реализовать пять сценариев автоматического распараллеливания вычислений в библиотеке *Insectae*. Самым простым подходом является использование многопоточных вычислений для реализации

---

\*Работа выполнена при финансовой поддержке РФФИ (грант № 20-07-01053 А).

всех рассматриваемых паттернов. Во многих важных практических задачах оптимизации наиболее тяжелой вычислительной частью является вычисление целевой функции, примерами таких задач являются, например, задачи машинного обучения, а также задачи структурной биоинформатики [3]. В таких задачах можно достичь существенного ускорения вычислений за счет распараллеливания только паттерна `evaluate` с использованием технологии MPI. Для алгоритмов, использующих интенсивные матрично-векторные вычисления, возможно ускорение этих вычислений с использованием графических процессоров. Примером такого алгоритма является алгоритм бактериального поиска, в котором на каждом шаге должна вычисляться матрица расстояний между всеми особями популяции (паттерн `signals`). Еще одним способом повышения быстродействия является использование островных (или гибридных) моделей, которые также относительно безболезненно распараллеливаются с помощью технологии MPI. Наконец, эффективная параллельная реализация на основе технологии MPI возможна и для клеточных моделей [4] с локальным взаимодействием между особями популяции.

Табл. 1. Время работы различных паттернов

	<code>evaluate</code>	<code>foreach</code>	<code>neighbors</code>	<code>pairs</code>	<code>pop2ind</code>	<code>signals</code>	<code>reduce</code>	$S(10)$
Bees	11%	79%					8%	<b>5.2</b>
BFOA	9%	24%	3%			62%		<b>8.4</b>
CSO	28%		60%				9%	<b>4.8</b>
DE	12%			25%	62%			<b>9.1</b>
GA	24%	42%	31%					<b>7.8</b>
PSO	27%	69%					1%	<b>7.3</b>
SA	63%	35%						<b>8.4</b>

Полученные предварительные результаты по реализации автоматического распараллеливания генетического алгоритма и алгоритма бактериального поиска в двух упомянутых выше сценариях (с использованием технологий MPI и CUDA) показали работоспособность и перспективность предлагаемого подхода. Важной особенностью этого подхода является то, что применение параллельных вычислений при любом способе их реализации является полностью прозрачным для пользователя, единственным действием которого должно быть указание желаемой модели параллельного выполнения выбранного им алгоритма оптимизации.

## Литература

1. Карпенко А.П. Современные алгоритмы поисковой оптимизации. Москва: Изд-во МГТУ им. М.В Баумана, 2014. 446 с.
2. Полуян С.В., Рейнгард Н.М., Ершов Н.М. Самоадаптация в алгоритмах роевой оптимизации // Вестник Российского университета дружбы народов: Серия Математика, информатика, физика. 2014. № 2. С. 415–418.
3. Полуян С.В., Ершов Н.М. Применение параллельных эволюционных алгоритмов оптимизации в задачах структурной биоинформатики // Вестник УГАТУ. 2017. Т. 21, № 4. С. 143–152.
4. Ершов Н.М. Неоднородные клеточные генетические алгоритмы // Компьютерные исследования и моделирование. 2015. Т. 7, № 3. С. 775–780.

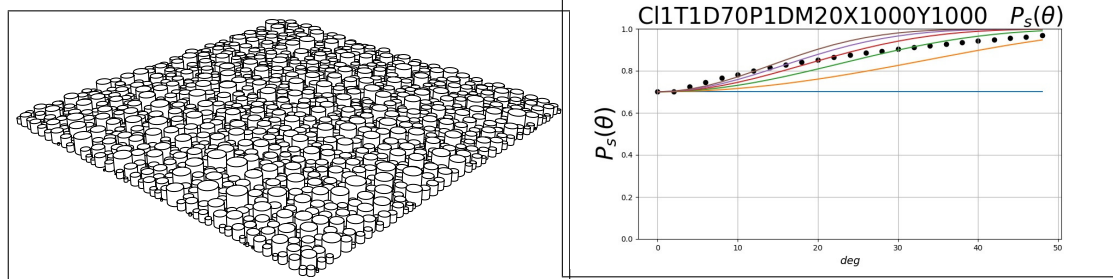
# Численное моделирование статистических свойств теплового радиоизлучения планковских полей разорванной облачности в микроволновом диапазоне

Я.В. Кошцов<sup>1</sup>, Я.А. Илюшин<sup>1,2</sup>

<sup>1</sup>Московский государственный университет имени М. В. Ломоносова, <sup>2</sup>Институт радиотехники и электроники имени В. А. Котельникова РАН

Контроль за характеристиками атмосферы является важной задачей при дистанционном зондировании окружающей среды. Одним из методов зондирования атмосферы является СВЧ-радиометрическое зондирование с искусственных спутников Земли. Метод основан на измерениях уходящего теплового излучения системы атмосфера – морская поверхность в радиодиапазоне. Наблюдения радиотеплового излучения системы «атмосфера- поверхность океана» позволяют получить информацию о состоянии облачных полей в атмосфере и количественно определить многие метеопараметры, например, массу водяного пара, водозапас и эффективную температуру облаков. Недостатком значительной части большинства предыдущих исследований является использование однородной плоскостлой модели облачного слоя, игнорирующей его неоднородную структуру.

Для учёта структуры облачных полей в данном исследовании рассчитываются статистические свойства моделей разорванных облачных полей. Для этого проводится численное моделирование случайных полей с распределениями облаков по размерам предложенными в работе Планка [7] и в сборнике под редакцией Мазина [3], полученное на основе самолетных измерений на Украине. Применяется допущение о цилиндрической форме моделируемых облаков при расчетах лучевых характеристик полей в СВЧ-диапазоне на основании статьи Кутузы и Смирнова [2]. Для реализованных случайных разорванных облачных полей вычисляется, связанная с наблюдаемой радиояркостной температурой  $T_b$  соотношением установленным в работе Гагарина и Кутузы [1], вероятность перекрытия небосвода в направлении визирования под зенитным углом  $\theta$  и сравнивалась с эмпирической зависимостью из статье Охвирля и Эпика [4]. Исследуется соответствие свойств теплового радиоизлучения реализуемых полей с эмпирическими и возможности применения предложенных моделей при СВЧ-радиометрическое зондировании.



(а) Пример проекции численной реализации случайного поля разорванной облачности.

(б) Пример сравнения вероятности перекрытия неба облаками с теоретическими зависимостями

Рис. 1:  $S = 0.7, \alpha = 1, D_m = 20, \beta = 0, \eta = 1$

Численный расчёт проводился на языке программирования C++. Распределение облаков по размерам контролировалось по критерию статистического согласия Колмогорова. После рассчитывались видимые проекции облачного поля в заданном направлении визи-

рования (рис. 1а) и процент перекрытия неба вычислялся пиксельным методом, допускающим параллельный расчет на прямоугольных сегментах изображения, и усреднялся по азимутальному углу. Полученные таким образом расчетные зависимости от зенитного угла визирования сравнивались с семейством эмпирических зависимостей (рис. 1б). Для применимости вычисляемых значений для учёта структуры облачности в атмосфере необходимо строить поля сопоставимые по размерам с реальными облачными полями в тропосфере Земли, моделирование и анализ которых требует значительных вычислительных затрат. Для расчета вероятности перекрытия неба необходимо, во-первых, чтобы средний размер облаков в поле был больше размера, соответствующего размеру пикселя, вследствие чего приходится использовать размер изображения пропорциональный размеру поля и, во-вторых, многократно повторить расчет с различными углами визирования. Численное моделирование статистических свойств теплового радиоизлучения моделей полей разорванной облачности подразумевает потребность произвести расчёта с ансамблем смоделированных случайных полей достаточно большого размера, который должен достигать нескольких сотен для каждого набора параметров, и использование проекций с хорошим разрешением, что соответствует изображению 20 - 50 тыс. пикселей, поэтому оптимально использование для вычислений высокопроизводительных ЭВМ, позволяющим производить вычисления в 400 - 2000 раз быстрее.

На данном этапе вычислен ансамбль квадратных полей размером 1000 максимальных размеров облака для набора всех управляющих параметров, два для каждого распределения. Вычисления произведены на многоядерном персональном компьютере (процессор - 6 ядер, 6 потоков, частота 3.6 ГГц; оперативную память - 32 Гб), при этом время расчета одного поля и его характеристик при указанных размерах составляет в среднем 8 - 12 минут, для изображений 1000 на 1000 пикселей. Методы моделирования и расчета опробовались на суперкомпьютере «Ломоносов».

## Литература

1. Гагарин С.П., Кутуза Б.Г. Влияние морского волнения и неоднородностей атмосферы на СВЧ излучение системы атмосфера-морская поверхность. // Исследование Земли из космоса. 1983 №3. С.88-99.
2. Кутуза Б.Г, Смирнов М.Т. Влияние облачности на усредненное радиотепловое излучение системы "атмосфера-поверхность океана". // Исследования Земли из космоса 1980 №3 с. 76-83.
3. Кучевые облака и связанная с ними деформация полей метеоэлементов. Труды ЦАО. Под редакцией Мазина И. П. и Шметера С. М. - Москва : Гидрометеиздат, 1977. . Вып. 134 с. 53 - 164.
4. Охвирль Х.А., Эпик Р.С. Описание изменчивости усредненных по азимуту параметров кучевой облачности и длинноволновой радиации посредством собственных векторов. В кн. Изменчивость облачности и полей радиации. Тарту. АН ЭССР, 1978. с.5-22.
5. Lund I.A., Shanklin M.D. Universal methods for estimating probabilities of cloud-free lines-of-sight through the atmosphere.// Journal of Applied Meteorology and Climatology 1973, V.12(1), PP. 28-35. DOI: 10.1175/1520-0450(1973)012<0028:UMFEP0>2.0.CO;2.
6. Lund I. A., Shanklin M. D. (1972). Photogrammetrically Determined Cloud-Free Lines-Of-Sight Through The Atmosphere// Journal of Applied Meteorology and Climatology, 11(5), 773-782. DOI: 10.1175/1520-0450(1972)011<0773:PDCFLO>2.0.CO;2.
7. Plank V. G. Cumulus clouds in The size distribution of representative Florida populations. // Journal of Applied Meteorology and Climatology. 1969. V. 8. N. 1. P. 46-67.



## Содержание

<b>Полные и короткие статьи.....</b>	<b>3</b>
Approaches, services, and monitoring in a distributed heterogeneous computational environment for the MPD experiment <i>A.A. Moshkin, I.S. Pelevanyuk, D.V. Podgainy, O.V. Rogachevsky, O.I. Streltsova, M.I. Zuev</i> .....	4
Automatic analysis of large-scale CT databases for fatty liver disease screening: the proof of concept <i>N.S. Kulberg, V.A. Gombolevskiy, A.P. Gonchar, R.V. Reshetnikov, S.S. Repin, S.B. Prokudaylo, V.P. Novik, M.R. Kodenko, M.A. Gusev, I.D. Fateev</i> .....	12
HPC TaskMaster - система мониторинга эффективности задач суперкомпьютера <i>П.С. Костенецкий, А.Б. Шамсутдинов, Р.А. Чулкевич, В.И. Козырев</i> .....	18
Improving SL-AV Global Atmosphere Model Computational Efficiency with I/O and Algorithmic Optimizations <i>Mikhail Tolstykh, Rostislav Fadeev, Gordey Goyman, Vladimir Shashkin</i> .....	26
Optimization and regularization of the inverse problem for stochastic differential equation using graphics accelerators <i>O.I. Krivorotko, A.V. Neverov, T. Hohage</i> .....	32
VaLiPro: валидатор решений задач линейного программирования для кластерных вычислительных систем <i>Л.Б. Соколинский, И.М. Соколинская</i> .....	43
Автоматизация процессов преподавания практики на примере курса по параллельному программированию <i>А.Ю. Нестеров</i> .....	52
Валидация пакета программ «ЛОГОС» для решения задач судостроительной отрасли <i>А.С. Козелков, В.В. Курулин, А.Е. Таранов, С.В. Лашкин, К.С. Плыгунова, О.Л. Крутякова</i> .....	62
Дополнительное распараллеливание MPI программ с помощью системы SAPFOR <i>Н.А. Катаев, А.С. Колганов</i> .....	74
Исследование фазовых переходов в модели Поттса методом Ванг-Ландау <i>М.А. Фадеева</i> .....	87
Компетенции и подготовка кадров в области суперкомпьютерных технологий <i>Ю.Я. Болдырев, В.В. Глухов, О.А. Картавенко</i> .....	97
Математика и компьютеринг: опыт покорения космоса и перспективы для «Будущего Земли». К 110-летию со дня рождения М.В.Келдыша и 60-летию полета Ю.А.Гагарина в Год науки и технологий <i>Т.А. Сушкевич, С.А. Стрелков, С.В. Максакова</i> .....	104
Моделирование магнитных свойств двумерной металлической пены <i>О.Д. Клименкова</i> .....	116

Нейросетевой метод сравнительного анализа трехмерных структур белков на основе суперсемейств доменов <i>А.В. Мазеев, Н.Н. Попова, Д.А. Суплатов</i> .....	124
Опыт реализации магистерской программы на базовой кафедре <i>И.И. Левин, И.Ю. Кузнецова, Б.Е. Механцев</i> .....	131
Применение графических ускорителей для поиска оптимального совмещения пар жестких трехмерных структур методом Кабша <i>И.А. Тимохин, Н.Н. Попова, Д.А. Суплатов</i> .....	137
<b>Аннотации стендовых докладов</b> .....	<b>144</b>
Builder-like Design Pattern for effective In-Situ Processing <i>Boris Morose, Alex Margolin</i> .....	145
First Results of Performance Evaluation of Geospatial Raster Data Processing Systems <i>K.V. Bykov, R.A. Rodrigues Zalipynis</i> .....	147
Maximizing the loop tiling ability via static analysis <i>Al. Levchenko</i> .....	149
Monte Carlo simulations of the two-point correlation functions in the six-vertex lattice model <i>P.A. Belov</i> .....	151
Parallel computing of a cavitation bubbles formation in a two-stage reciprocating pump-compressor <i>A.V. Zanin</i> .....	153
Research of explicit numerical methods calculations on CPU with x86 and ARM architecture <i>Egor Elchinov, Vladislav Furgailo, Nikolay Khokhlov</i> .....	155
Генератор пакетных файлов для планировщика задач SLURM <i>И.А. Михайлов, П.С. Костенецкий</i> .....	157
Методы моделирования электронного газа в плоском диоде <i>И.В. Куликова</i> .....	159
Моделирование динамики горения водорода при помощи полносвязной нейронной сети UNET <i>Я.М. Карандашев, Е.В. Михальченко, М.Ю. Мальсагов, В.Ф. Никитин</i> .....	161
Моделирование процесса распространения загрязняющих веществ в Геленджикской бухте на высокопроизводительной вычислительной системе <i>Ю.В. Белова, Е.О. Рахимбаева, А.А. Филина, А.В. Никитина</i> .....	163
О ранговом распределении суперкомпьютеров в списке top500 <i>С.К. Шикота</i> .....	165
Оценка эффективности модифицированных CSR форматов представления матриц при реализации итерационных методов решения систем уравнений <i>Р.М. Куприй, Б.И. Краснопольский</i> .....	169
Параллельный алгоритм численного метода моделирования массопереноса двухфазной жидкости в трещиновато-поровом коллекторе <i>Л.В. Еникеева, Р.М. Узянбаев, А.А. Мазитов, Ю.О. Бобренёва, И.М.Губайдуллин</i> .....	171

Повышение производительности расчетов ab initio молекулярной динамики при помощи GPU <i>А.А. Елисеев, Н.Г. Андреади, А.А. Митрофанов</i> .....	173
Разработка распределенных алгоритмов управления системами роевого интеллекта с использованием методов встроенной эволюции <i>А.Г. Николашкин, Н.М. Ершов</i> .....	175
Распараллеливание задачи распространения динамических волновых возмущений в трещиноватых геологических средах <i>В.С. Саган, Н.И. Хохлов</i> .....	177
Расчет поляризационных матриц рассеяния фрактальных кластерных частиц атмосферного аэрозоля Титана <i>М.П. Черешенков, Я.А. Илюшин</i> .....	180
Реализация метода FDTD с граничными условиями PML с использованием вычислений в смешанной точности <i>А.Н. Арисова, В.Д. Волокитин, Е.С. Ефименко, И.Б. Мееров</i> .....	182
Реализация параллельных вычислений в рамках программной системы поддержки эволюционных и роевых методов оптимизации <i>А.А. Тимачев, Н.М. Ершов</i> .....	185
Численное моделирование статистических свойств теплового радиоизлучения планковских полей разорванной облачности в микроволновом диапазоне <i>Я.В. Копцов, Я.А. Илюшин</i> .....	187
<b>Содержание</b> .....	<b>189</b>

Научное издание

**СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ**

Труды международной конференции

*27–28 сентября 2021 г. Москва*

Издательство «МАКС Пресс»

Главный редактор: *Е. М. Бугачева*

Обложка: *М. А. Еронина*

Напечатано с готового оригинал-макета

Подписано в печать 26.09.2019 г. Формат 60х90 1/8.

Усл.печ.л. 24,0. Тираж 10 экз. Изд. №. 162.

Издательство ООО «МАКС Пресс». Лицензия ИД N 00510 от 01.12.99 г.  
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,  
2-й учебный корпус, 527 к. Тел. 8(495) 939-3890/91. Тел./Факс 8(495) 939-3891.

Отпечатано в полном соответствии с качеством  
предоставленных материалов в ООО «Фотоэксперт»  
115201, г. Москва, ул. Котляковская, д.3, стр. 13.