Московский государственный университет имени М.В.Ломоносова
Суперкомпьютерный консорциум университетов России
Российская академия наук
Московский Центр фундаментальной и прикладной математики

Суперкомпьютерные дни в России 2026

Труды международной конференции

28–29 сентября 2026 г., Москва

В 2026 году конференция посвящена памяти Владимира Валентиновича Воеводина, организатора и руководителя данной серии конференций

Московский государственный университет имени М.В. Ломоносова
Суперкомпьютерный консорциум университетов России
Российская академия наук
Московский Центр фундаментальной и прикладной математики

В 2026 году конференция посвящена памяти безвременно ушедшего главного организатора данной серии конференций — первого председателя программного комитета, председателя организационного комитета, члена-корреспондента РАН, д.ф.-м.н., профессора Владимира Валентиновича Воеводина

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ-2026

Труды международной конференции

28–29 сентября 2026 г., Москва

МАКС ПРЕСС

МОСКВА – 2026

УДК 519.7
ББК 22.18
С89

https://elibrary.ru/hnkaaw

Под редакцией
А.С. Антонова, Вад. В. Воеводина, Д.А. Никитенко

С89 **Суперкомпьютерные дни в России-2026** : Труды международной конференции. 28–29 сентября 2026 г. Москва / Под. ред. А.С. Антонова, Вад. В. Воеводина, Д.А. Никитенко. – Москва : МАКС Пресс, 2026. – 282 с.

ISBN: 978-5-317-07665-8, e-ISBN: 978-5-317-07672-6
https://doi.org/10.29003/m5440.978-5-317-07665-8

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

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

ISBN: 978-5-317-07665-8
e-ISBN: 978-5-317-07672-6
https://doi.org/10.29003/m5440.978-5-317-07665-8

Полные и короткие статьи

End-to-end acceleration of YOLOv10n video annotation in Rust using USLS and ONNX Runtime

G.E. Gorsky

Bauman Moscow State Technical University, Moscow, Russia

This paper presents an end-to-end acceleration of an automated video annotation pipeline that decodes an input video, performs frame-wise object detection, and writes an annotated output stream. The pipeline is implemented in Rust and uses YOLOv10n on an NVIDIA GPU through USLS and ONNX Runtime with the CUDA Execution Provider. Profiling of the baseline implementation reveals two major bottlenecks: low GPU utilization caused by strictly sequential execution with little overlap between CPU and GPU work, and high CPU overhead of rendering in scenes with many detections. To address these issues, the pipeline is redesigned as a staged architecture with bounded queues, a detector worker pool for concurrent frame processing, and a separate annotator pool that removes visualization from the detector critical path. In addition, a lightweight drawing mode reduces the per-frame cost of rendering when the number of detections is large. Experiments on a single dense-scene 1080p video of 330 s at 25 fps reduce end-to-end processing time from 590 s to about 50 s, giving more than a 10x speedup for the evaluated workload.

Keywords: video analytics, object detection, pipeline parallelism, GPU acceleration, ONNX Runtime, Rust

1. Introduction

Object detection is a core building block in video analytics, monitoring, and dataset creation. In offline annotation workflows, the primary user-visible metric is the time-to-result for an entire input file: a video must be fully decoded, processed, annotated, and saved as an output artifact suitable for inspection or downstream labeling. This end-to-end (E2E) requirement differs from reporting model latency in isolation because practical annotation systems include heterogeneous stages: input demuxing and decoding, frame format conversion and resizing, tensor preparation, GPU inference, post-processing and filtering, visualization (overlay drawing), output encoding, disk I/O, and cross-device synchronization. As a consequence, E2E wall-clock time reflects the combined behavior of the model and the surrounding system rather than the detector alone.

Offline annotation is a representative workload where small per-frame overheads quickly become decisive. A 5-minute Full HD clip at 25 fps contains 7-8 thousand frames; even a 2-3 ms inefficiency per stage can translate into tens of seconds. Here, faster-than-real-time means processing an N-second input in less than N seconds while preserving output quality and frame order.

Modern one-stage detectors can achieve high throughput on contemporary GPUs, yet many per-frame pipelines remain slower than expected because they serialize host-side work around inference. When a pipeline processes each frame strictly sequentially, CPU steps such as pixel format conversion, memory copies, tensor packing, and post-processing delay GPU submission and reduce CPU–GPU overlap. The result is intermittent GPU idling even if the detector itself can run faster. Importantly, the limiting stage is content-dependent. For sparse scenes, inference and post-processing often dominate. For dense

scenes, visualization and per-detection bookkeeping may dominate E2E time because drawing hundreds of boxes per frame is CPU- and memory-bandwidth intensive.

This «overlay as a bottleneck» is not a niche observation; it is explicitly reflected in production guidance. For example, DeepStream's performance documentation [1] describes benchmarked results as end-to-end application performance while noting that output rendering is turned off to achieve peak inference performance. Moreover, DeepStream's best-practice notes explain that output rendering, OSD, and tiling consume compute resources and can reduce inference throughput, and that bounding-box generation may be unnecessary when rendering is disabled – except when the output must be streamed or saved to disk. Offline annotation is exactly the case where overlay cannot be removed: the annotated output video is the deliverable. Therefore, E2E optimization must treat visualization as a first-class stage and reduce its impact without sacrificing interpretability.

Typical file-based analytics pipeline also follows a clear stage pattern that makes these interactions visible. Documentation on converting DeepStream pipelines to Intel DL Streamer uses an example that "reads a video from the input file, decodes it, runs inference, overlays the inferences, re-encodes and outputs a new .mp4 file". This pipeline shape is shared across ecosystems because it mirrors the functional requirements of offline annotation. Consequently, performance engineering for annotation is best approached as pipeline engineering: maximizing overlap between decode/prepare, inference, post-processing, overlay, and encode/write, and ensuring that no single stage becomes the long pole.

The detector family used in this work is YOLO, which remains popular for real-time and near-real-time detection due to its balance between accuracy and computational cost. YOLOv10 targets improved latency–accuracy trade-offs and motivates efficiency improvements at the detector and post-processing level [3,4]. However, even with an efficient detector, overall throughput depends heavily on system design choices: how stages are ordered and parallelized, how host threads interact with GPU execution, and how expensive non-inference stages (especially rendering and encoding) become under realistic workloads. Applied literature also reflects sustained demand for YOLO-based real-time detection pipelines in practical systems [5].

At the same time, model-level approaches such as quantization and pruning address a different axis of optimization, often targeting constrained devices. For example, Mamadaev and Minitaeva discuss mathematical modeling of neural network compression techniques for mobile platforms [6]. In contrast, our focus is orthogonal: we keep the detector and inference engine fixed and seek gains purely from reorganizing the surrounding pipeline.

A second key factor is the deployment runtime and its hardware bindings. ONNX Runtime supports heterogeneous execution through Execution Providers [7], enabling GPU acceleration on NVIDIA hardware via the CUDA Execution Provider [8]. In addition, libraries such as USLS provide Rust-friendly access to common vision models and integrate with ONNX Runtime, making it practical to build an end-to-end pipeline in a systems language while keeping the trained model fixed [9]. This matters for engineering teams because model changes may be undesirable due to accuracy constraints, certification, or integration cost, whereas pipeline-level optimization can often deliver large speedups without altering the model.

Systems research on video analytics similarly emphasizes that end-to-end performance is shaped by pipeline-level decisions rather than model speed alone. Microservice-based and modular pipelines must balance resources across stages and adapt to content-dependent load. Magic-Pipe, for example, studies self-optimizing video analytics pipelines and shows that stage imbalance and dynamic video content can materially affect response time,

motivating measurement-driven tuning and resource allocation [10]. While our setting differs (single-process offline annotation rather than microservices), the underlying lesson transfers directly: content variability can shift the bottleneck between inference and non-inference stages, and performance improvements must address the true E2E critical path rather than one isolated component.

This paper focuses on single-node CPU-GPU throughput optimization for a Rust-based offline pipeline that executes YOLOv10n via USLS and ONNX Runtime with the CUDA Execution Provider. The detector and inference backend are unchanged; the contribution is system-level. We profile a sequential baseline, then redesign execution as a staged pipeline with bounded queues and separated worker pools. In particular, we decouple overlay rendering from detection and introduce lightweight drawing for dense scenes.

The paper makes three main contributions:

1. A measurement-driven characterization of bottlenecks in a full "decode → detect→ → draw → encode/write"pipeline, showing how host-side serialization and dense-scene visualization can dominate E2E wall-clock time.

2. A staged redesign using bounded queues, a detector worker pool, and a dedicated annotator pool that increases CPU–GPU overlap and removes rendering from the detector critical path.

3. Practical tuning rules for selecting detector concurrency and estimating safe multi-process scaling under VRAM and RAM constraints, together with a lightweight drawing mode that reduces visualization overhead while preserving output interpretability.

The remainder of the paper reviews related work, formalizes the problem and constraints, describes the baseline and redesigned pipelines together with a simple throughput model, and then reports experimental results and deployment-oriented memory measurements.

2. Related work

This work sits at the intersection of (i) systems for high-throughput video analytics, (ii) pipeline frameworks used in production deployments, and (iii) model-centric optimizations that are orthogonal to pipeline organization.

2.1. Video analytics systems and dataflow execution

Prior systems research demonstrates that high performance for video analysis requires more than running a detector per frame. NoScope introduces techniques to reduce the cost of neural inference over fixed-angle video by exploiting specialization and cascades, illustrating that naive application of DNN inference can be prohibitively expensive at scale and that system-level optimization can yield dramatic gains [11]. Scanner proposes a dataflow-oriented system for efficient video analysis and scheduling across heterogeneous hardware [12], emphasizing that throughput depends on how computation is expressed and executed rather than on individual kernels alone. While these systems target different deployment settings (large-scale querying and cluster execution), they motivate the same core idea used here: pipeline organization and scheduling are central to performance.

2.2. Production pipelines and the role of overlay/output

Industrial toolkits commonly express video analytics as explicit pipelines that combine decoding, inference, overlay, and encoding. Performance documentation [1] for production frameworks notes that overlay and output rendering can consume meaningful compute resources, and benchmark recipes often disable these stages when reporting peak inference

throughput. For offline annotation, however, overlay and output writing are not optional: they define the deliverable. Therefore, E2E optimization must treat visualization and encoding as first-class stages and must manage their interaction with inference, rather than optimizing the detector alone.

2.3. Content-dependent tuning and self-optimization

A recurring theme in video analytics is that workload characteristics vary across time and across videos. Magic-Pipe [10] studies microservice-based analytics pipelines and introduces mechanisms to adapt resource allocation and tuning based on video content, with the explicit goal of reducing end-to-end response time. Although our pipeline is a single offline application, the dense-vs-sparse scene behavior we observe is a concrete example of content-dependent bottlenecks, and it motivates both decoupling visualization and using measurement-driven configuration.

2.4. Model-level efficiency vs. pipeline-level efficiency

Many works improve inference efficiency by changing the model (quantization, pruning, architectural redesign) or by altering post-processing. YOLO-based detectors are frequently discussed in applied contexts for real-time detection, and model-level optimizations are crucial for deployment on constrained devices. In contrast, the present paper keeps the model and inference engine fixed and focuses on improvements obtainable solely by reorganizing the pipeline, increasing overlap between CPU and GPU stages, and reducing visualization overhead under dense detections. This makes the approach applicable when the detector choice is constrained by accuracy, compatibility, or organizational requirements.

In summary, prior work establishes that end-to-end performance in video analytics is governed by pipeline structure, stage imbalance, and content variability. Our contribution is to apply these insights to a practical Rust-based offline annotation pipeline and to provide concrete redesign and tuning steps that deliver large E2E gains while keeping the detector unchanged.

3. Methodology

3.1. Setting of the problem

We consider an offline video annotation task in which an input video file must be fully processed to produce an output video with overlaid detections. The pipeline includes decoding, frame preparation, GPU inference, post-processing, rendering, and output encoding/writing. Let the input contain N frames, and let the primary performance metric be the end-to-end (E2E) wall-clock time T required to complete the whole file. Unlike per-frame model latency, T captures the user-visible time-to-result and accounts for all non-inference stages that may dominate in practice.

The goal is to minimize T while keeping the detector and inference backend fixed: YOLOv10n runs on an NVIDIA GPU via USLS and ONNX Runtime with the CUDA Execution Provider. The system must preserve frame order and produce an interpretable annotated output; however, the visualization strategy may be simplified if it reduces runtime without compromising readability. The optimization is constrained by the available CPU parallelism and memory budgets (RAM and GPU VRAM), which limit how far the pipeline can be parallelized using worker pools and queues.

Formally, given an input video and a target machine, we seek an execution organization and configuration parameters—detector worker count k, annotator thread count a, and a visualization mode (standard vs. lightweight)—that minimize T under resource constraints. The central hypothesis is that the baseline strictly sequential execution underutilizes the GPU because CPU work serializes GPU submission, and that dense-scene workloads can make visualization the effective critical path even when inference is fast. These assumptions motivate a staged redesign with bounded queues and decoupled rendering, followed by a simple throughput model and measurement-driven rules for selecting k and estimating safe multi-process scaling.

3.2. Baseline pipeline

The baseline implementation processes frames in a single sequential loop:

1. Decode frame **i** from the input stream.
2. Convert/resize and prepare input tensor.
3. Run YOLOv10n inference on GPU.
4. Post-process outputs (decode boxes, filter by confidence, class mapping).
5. Draw overlays (boxes/labels) on CPU into the output frame.
6. Encode and write the annotated frame.

This design is simple and correct, but it serializes CPU preparation, GPU inference, and CPU visualization per frame. As a result, the GPU cannot run continuously if CPU stages are slower than expected, and visualization can block progress for the entire pipeline.

3.3. Staged redesign with bounded queues

The redesigned pipeline is organized as a staged system connected by bounded queues, enabling overlap between stages and isolating expensive rendering from detection. The key stages are:

- Decode/prepare stage: reads and decodes frames, performs minimal format conversion, attaches a monotonically increasing frame index, and pushes frames into a bounded queue.

- Detection stage (worker pool, size k): each detector worker pops a frame, performs input preparation, runs GPU inference, executes post-processing, and pushes detection results (frame + boxes) to the next queue.

- Annotation stage (worker pool, size a): annotators pop detection results and render overlays into frames, then forward annotated frames to the encoder/writer stage.

- Encode/write stage: encodes frames and writes the output file, preserving order (using frame indices and a small reordering buffer if necessary).

Bounded queues enforce backpressure and prevent unbounded memory growth when one stage is slower than the others. Separating detection and annotation ensures that rendering does not stall GPU progress: detectors continue processing new frames while annotators draw previous ones.

3.4. Lightweight drawing mode

Dense scenes may yield many detections per frame, making overlay rendering expensive. In the measurements, lightweight drawing kept all detections but rendered thin single-color boxes by direct RGB-buffer writes, without RGBA conversion, translucent fills, shadows, or alpha blending. Labels and confidences used a simpler text path, and no top-K suppression was applied in Table 1. This reduces memory traffic but gives lower visual richness: no class-specific palette or filled transparent regions.

3.5. Throughput model for E2E time

In steady state, the redesigned pipeline behaves like a staged service system limited by the slowest aggregate stage. Let $\mu_{\text{det}}(k)$ denote detection throughput with k workers, μ_{draw} drawing throughput, and μ_{wr} encoding/writing throughput. In (1), μ_{draw} is shorthand for $\mu_{\text{draw}}(a, m)$, where a is the annotator count and m is the drawing mode. Decode/prepare can also become the bottleneck for other videos or codecs. The overall throughput is defined as:

$$\mu = \min\left(\mu_{\text{det}}(k),\ \mu_{\text{draw}},\ \mu_{\text{wr}}\right) \tag{1}$$

For a video with N frames and an approximately constant initialization and teardown overhead T_{init}, the E2E time is approximated by:

$$T \approx T_{\text{init}} + \frac{N}{\mu}. \tag{2}$$

Equations (1) and (2) are a coarse bottleneck model, not a full predictor. They explain diminishing returns when $\mu_{\text{det}}(k)$ exceeds drawing or writing throughput. For another video, codec, drawing mode, or machine, the bottleneck can move to another stage, so configuration must still be selected by measurement.

3.6. Automatic selection of the detector worker count

The worker count k is selected using measurements from the first M frames. Two average per-frame times are estimated. The value t_{cpu} represents worker CPU time excluding GPU inference. It includes input preparation, post-processing, and packaging. The value t_{gpu} represents GPU inference time. GPU busy fraction for a single worker is approximated as:

$$u_1 = \frac{t_{\text{gpu}}}{t_{\text{cpu}} + t_{\text{gpu}}}. \tag{3}$$

A minimal worker count that tends to saturate the GPU is defined by:

$$k_{\text{sat}} = \left\lceil \frac{1}{u_1} \right\rceil = \left\lceil 1 + \frac{t_{\text{cpu}}}{t_{\text{gpu}}} \right\rceil. \tag{4}$$

The final practical choice is limited by CPU resources. Let p be the number of logical CPU threads. Let a be the annotator thread count. Let r be a small reserve for encoding and I/O. Then:

$$\begin{aligned} k_{\text{max}} &= \max(1,\ p - a - r), \\ k_{\text{opt}} &= \min(k_{\text{sat}},\ k_{\text{max}}). \end{aligned} \tag{5}$$

3.7. Estimating the maximum number of parallel detector processes

For multi-process scaling, per-process memory costs are measured. VRAM consumption per process is denoted by v. Resident RAM consumption is denoted by r_{rss}.

Given free budgets V_{free} for VRAM and R_{free} for RAM, and safety factors γ_V and γ_R in the range 0.85–0.90, limits are computed as:

$$m_{\text{VRAM}} = \left\lfloor \gamma_V \cdot \frac{V_{\text{free}}}{v} \right\rfloor, \quad m_{\text{RAM}} = \left\lfloor \gamma_R \cdot \frac{R_{\text{free}}}{r_{\text{rss}}} \right\rfloor. \tag{6}$$

The resulting safe process count is:

$$m_{\text{opt}} = \max\left(1, \min(m_{\text{VRAM}},\ m_{\text{RAM}})\right). \tag{7}$$

Equations (6) and (7) are intended for deployment planning on the target machine. They avoid relying on a single fixed memory footprint estimate.

4. Experiments and results

4.1. Experimental setup

A single dense test video is used as a deliberately selected stress case, not as a benchmark suite. It has resolution 1920×1080, frame rate 25 fps, duration 330 s, bitrate about 1000 kbps, and size approximately 37 MB. Detections are present in most frames. The total number of frames is $N = 330 \times 25 = 8250$.

The main workstation uses an AMD Ryzen 7 7800X3D (8C/16T) and an NVIDIA RTX 4070 Ti Super. Per-process memory footprint is also measured on RTX 4070 Ti Super, RTX 3090, and RTX 5090. VRAM usage is observed with nvtop [13]. Resident RAM usage is measured as process RSS with htop.

The primary metric is E2E wall-clock time in seconds for the whole video. This time includes decoding, inference, visualization, and output writing.

4.2. End-to-end time

Table 1 reports E2E time for different detector worker counts k. In annotator-pool columns, $a = k$ annotator threads are used. The sequential baseline processes the video in 590 s; with $k = 1$ after staging, E2E time improves to 546 s. One wall-clock measurement per configuration is reported, so run-to-run variability is not characterized.

Table 1. E2E processing time for the 1080p (330 s, 25 fps) test video (seconds)

Workers (k)	**CUDA + workers**	**CUDA + workers + annotators**	**CUDA + workers + annotators + light drawing**
1	546	300	115
2	305	160	68
4	161	108	49
8	101	80	49
10	90	75	50
12	85	71	51
16	83	73	51

On the evaluated dense video, Table 1 supports the initial hypotheses. Increasing k from 1 to 4 substantially reduces E2E time, consistent with improved overlap and reduced

GPU idle time. Adding the annotator pool further reduces time; since $a = k$, this reflects both decoupling and more CPU parallelism for drawing. Lightweight drawing gives the best E2E time, 49–51 s. The best value, 49 s for 8250 frames, is about 168 fps, or 6.7 times faster than the 25 fps input. Saturation beyond $k \approx 4$–8 indicates that drawing and output writing then limit throughput.

4.3. Memory footprint and scaling estimate

Table 2 summarizes measured per-process memory usage for YOLOv10n detector processes.

Table 2. Per-process memory footprint for YOLOv10n detector processes

GPU	**VRAM per detector (MB)**	**Total VRAM (MB)**	**Detectors by VRAM (theor.)**	**RSS per detector (GB)**	**Shared (GB)**
RTX 4070 Ti Super	266	16384	61	1.4	0.30–0.33
RTX 3090	378	24576	65	1.8	0.30–0.33
RTX 5090	652	32768	50	2.2	0.30–0.33

The measured footprint varies across GPUs. This motivates the measurement-driven planning approach in equations (6) and (7), which jointly considers VRAM and RAM constraints.

5. Limitations and future work

The evaluation is based on a single dense test video and a primary workstation configuration, so the results should be read as evidence for this workload rather than proof for all offline annotation workloads. Stronger external validity requires videos with varied object density, codec/bitrate, resolution, and camera motion; other inputs may shift the bottleneck toward decoding, pre-processing, or post-processing.

The throughput reasoning in equations (1) and (2) intentionally simplifies the system as a staged pipeline dominated by the slowest stage. This approximation is useful for understanding saturation and diminishing returns, but it does not explicitly model second-order effects such as queue contention, cache behavior, memory bandwidth pressure during rendering, and driver/runtime scheduling effects. The worker-selection procedure also relies on early measurements (first M frames) and assumes that workload characteristics are approximately stationary; videos with strong temporal variation in object density may benefit from adaptive reconfiguration during execution. Reported wall-clock times can further be affected by OS scheduling noise, background load, GPU frequency behavior, and encoding parameters; repeating runs and reporting variability would provide a more robust picture. VRAM and RSS measurements also depend on drivers, allocators, and tools, so the reported values should be interpreted as practical planning estimates rather than exact accounting.

Several extensions are promising. Faster drawing primitives (including SIMD optimization) or alternative rendering paths could reduce CPU visualization costs further in dense scenes.

Richer instrumentation such as per-stage latency distributions, queue occupancy traces, and GPU utilization timelines would improve diagnosis and allow tighter calibration of the model and heuristics. Batching or micro-batching can be revisited under different CPU/GPU balance regimes and decode/encode settings. Finally, an important practical direction is supporting real-time operation on streaming inputs: extending the pipeline from file-based processing to a low-latency, backpressure-aware RTSP ingestion path with bounded buffering, frame dropping policies under overload, and stable end-to-end latency would make the system applicable to live camera analytics and continuous monitoring scenarios.

Conclusion

This paper demonstrates that large E2E speedups in a dense-video annotation workload can be achieved without changing the detector or inference backend by improving CPU-GPU overlap and decoupling visualization from detection. In a Rust pipeline using USLS [9] and ONNX Runtime [7] with the CUDA Execution Provider [8] to execute YOLOv10n [3,4], the redesigned system employs a detector worker pool, a dedicated annotator pool, and lightweight drawing. On the evaluated 1080p video, E2E time is reduced from 590 s to about 50 s, corresponding to more than 10x speedup and faster-than-real-time throughput.

The results show that, in the evaluated high-density scene, E2E throughput can be limited by non-inference stages and visualization can become the dominant cost. Staged execution with bounded queues and measured worker counts scales until the next bottleneck dominates, explaining saturation once drawing and output writing limit throughput. The memory-footprint measurements and VRAM/RAM planning rules support deployment decisions for multiple detector processes. Overall, measurement-driven pipeline organization can deliver substantial gains while leaving the detector unchanged.

Acknowledgement

[1] NVIDIA, "DeepStream Performance,"NVIDIA DeepStream Documentation. [Online]. Available: https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_Performance.html. Accessed: Jan. 4, 2026.

[2] Intel, "Converting NVIDIA DeepStream Pipelines to Intel® Deep Learning Streamer (DL Streamer),"Intel® Open Edge Platform Documentation. [Online]. Available: https://docs.openedgeplatform.intel.com/dev/edge-ai-libraries/dlstreamer/dev_guide/converting_deepstream_to_dlstreamer.html. Accessed: Jan. 4, 2026.

[3] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, "YOLOv10: Real-Time End-to-End Object Detection,"arXiv:2405.14458, 2024. [Online]. Available: https://arxiv.org/abs/2405.14458.

[4] Ultralytics, "YOLOv10,"Ultralytics Documentation. [Online]. Available: https://docs.ultralytics.com/models/yolov10/. Accessed: Jan. 4, 2026.

[5] T. A. Malkina and A. M. Minitaeva, "Obnaruzhenie ob'ektov v real'nom vremeni s YOLO [Real-time object detection with YOLO],"in Proc. II All-Russian Scientific Conference "Iskusstvennyi intellekt v avtomatizirovannykh sistemakh upravleniya i obrabotki dannykh,"Moscow, Russia, 2023, pp. 267–272. (In Russian).

[6] I. Mamadaev and A. Minitaeva, "Mathematical Modeling of Neural Network Compression Techniques for Mobile Platforms,"J. Phys.: Conf. Ser., vol. 3042, no. 1, p. 012003, 2025, doi: 10.1088/1742-6596/3042/1/012003.

[7] ONNX Runtime, "Execution Providers,"ONNX Runtime Documentation. [Online]. Available: https://onnxruntime.ai/docs/execution-providers/. Accessed: Jan. 5, 2026.
[8] ONNX Runtime, "CUDA Execution Provider,"ONNX Runtime Documentation. [Online]. Available: https://onnxruntime.ai/docs/execution-providers/CUDA-Execution Provider.html. Accessed: Jan. 5, 2026.
[9] J. Jamjamjon, "USLS: Rust library powered by ONNX Runtime,"GitHub repository. [Online]. Available: https://github.com/jamjamjon/usls. Accessed: Jan. 5, 2026.
[10] G. Coviello, Y. Yang, K. Rao, and S. Chakradhar, "Magic-Pipe: Self-Optimizing Video Analytics Pipelines,"in Proc. 22nd Int. Middleware Conf. (Middleware '21), 2021, pp. 79–90, doi: 10.1145/3464298.3484504.
[11] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "NoScope: Optimizing Neural Network Queries over Video at Scale,"Proc. VLDB Endow., vol. 10, no. 11, pp. 1586–1597, 2017, doi: 10.14778/3137628.3137664.
[12] A. Poms, W. Crichton, P. Hanrahan, and K. Fatahalian, "Scanner: Efficient Video Analysis at Scale,"ACM Trans. Graph., vol. 37, no. 4, Art. no. 138, 2018, doi: 10.1145/3197517.3201394.
[13] Syllo, "nvtop: GPU & Accelerator process monitoring,"GitHub repository. [Online]. Available: https://github.com/Syllo/nvtop. Accessed: Jan. 6, 2026.

HAlEM: новый гибридный параллельный алгоритм для линейного программирования на кластерных вычислительных системах

А.Э. Жулев, Л.Б. Соколинский

Южно-Уральский государственный университет
(национальный исследовательский университет)

Работа посвящена новому гибридному масштабируемому проекционному алгоритму HAlEM (Hybrid Along Edges Movement) для линейного программирования на кластерных вычислительных системах. Алгоритм начинает свою работу в произвольной вершине многогранника допустимых решений и строит оптимальный путь по ребрам до точки оптимума. Основным преимуществом HAlEM перед предшествующими разработками (AlFaMove, AlEM) является отказ от комбинаторного перебора всех возможных комбинаций гиперплоскостей за счет использования техники обновления матричного базиса, заимствованной из симплекс-метода. Это позволяет избежать экспоненциальной вычислительной сложности. Представлено формализованное описание алгоритма. Описана его параллельная реализация. Исследована масштабируемость параллельной реализации на кластерной вычислительной системе. Приведены результаты вычислительных экспериментов, подтверждающие высокую параллельную эффективность алгоритма HAlEM.

Ключевые слова: линейное программирование, алгоритм HAlEM, проекционный метод, параллельная реализация, MPI, кластерная вычислительная система, исследование масштабируемости, Netlb-LP.

1. Введение

Линейное программирование (ЛП) [1, 2] является одной из самых востребованных математических моделей для решения задач оптимизации в индустрии, экономике, науке и многих других сферах человеческой деятельности [3–6]. Основным методом решения задач ЛП на практике является симплекс-метод [7], позволяющий эффективно решать широкий класс оптимизационных задач с числом неизвестных до 50 000 [8]. Однако симплекс-методу присущ ряд недостатков, среди которых выделим следующие. Во-первых, симплекс-метод в общем случае характеризуется ограниченной масштабируемостью и не допускает эффективного распараллеливания более чем на 32 процессорных узлах [9]. Во-вторых, симплекс-метод может зацикливаться на вырожденных задачах ЛП [10], что не гарантирует его сходимость к решению [11, 12]. Поэтому остается актуальной задача разработки альтернативных эффективных методов ЛП, свободных от указанных недостатков. Одной из таких альтернатив являются методы проекционного типа, строящие на поверхности многогранника допустимых решений путь к оптимальной точке.

Идея проекционного метода чрезвычайно проста. Пусть в евклидовом пространстве $\mathbb{R}^n$ задана задача ЛП стандартного вида

$$\arg\max\{\langle \boldsymbol{c}, \boldsymbol{x}\rangle \mid A\boldsymbol{x} \leqslant \boldsymbol{b},\ \boldsymbol{x} \geqslant \boldsymbol{0}\}$$

с ограниченной областью допустимых решений, представляющей собой замкнутый выпуклый многогранник M. Стартуя из произвольной граничной точки $\boldsymbol{v}_0$ многогранника M, проекционный метод вычисляет точку

$$\boldsymbol{z} = \boldsymbol{v}_0 + \delta \boldsymbol{e}_{\boldsymbol{c}},$$

где $\boldsymbol{e_c}$ — единичный вектор, сонаправленный с вектором $\boldsymbol{c}$, δ — некоторое положительное вещественное число (см. рис. 1). Затем выполняется метрическая проекция

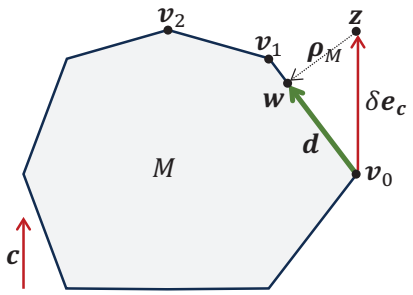


Рис. 1. Проекционный метод

точки $\boldsymbol{z}$, дающая на многограннике M точку $\boldsymbol{w}$:

$$\boldsymbol{w} = \boldsymbol{\rho}_M(\boldsymbol{z}).$$

Выбор числа δ должен гарантировать, что $\boldsymbol{w}$ будет на той же грани, что и $\boldsymbol{u}$. Далее вычисляется направляющий вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}_0$, который позволяет получить следующее приближение $\boldsymbol{v}_1$:

$$\boldsymbol{v}_1 = \arg\max \left\{ \|\boldsymbol{x} - \boldsymbol{v}_0\| \mid \boldsymbol{x} = \lambda \boldsymbol{d} + \boldsymbol{v}_0,\ \lambda \in \mathbb{R}_{>0},\ \boldsymbol{x} \in M \right\}.$$

Выполняя аналогичные действия для $\boldsymbol{v}_1$, получим приближение $\boldsymbol{v}_2$. Процесс заканчивается на k-том приближении, для которого

$$\boldsymbol{\rho}_M(\boldsymbol{v}_k + \delta \boldsymbol{e_c}) = \boldsymbol{v}_k.$$

На рис. 1 этой ситуации соответствует точка $\boldsymbol{v}_2$. Таким образом получаем решение задачи ЛП.

В работе [13] описан апекс-метод решения задачи ЛП, реализующий этот подход в виде итерационного численного алгоритма. Для приближенного вычисления метрической проекции на выпуклый многогранник M апекс-метод использует M-фейеровское отображение $\boldsymbol{\varphi}_M : \mathbb{R}^n \to \mathbb{R}^n$, характеризующееся тем, что образ $\boldsymbol{\varphi}_M(\boldsymbol{x})$ любой точки $\boldsymbol{x} \notin M$ будет ближе к многограннику M, чем $\boldsymbol{x}$:

$$\forall \boldsymbol{x} \notin M, \forall \boldsymbol{y} \in M : \|\boldsymbol{\varphi}(\boldsymbol{x}) - \boldsymbol{y}\| < \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Многократно применяя фейеровское отображение к произвольной внешней точке $\boldsymbol{x}^{(0)} \notin M$, мы получим точку на границе многогранника M, являющуюся некоторым приближением метрической проекции. Указанный фейеровский процесс называется псевдопроектированием, а получившаяся точка — псевдопроекцией. Хотя апекс-метод способен решать некоторые реальные задачи ЛП, он обладает двумя существенными недостатками. Первый недостаток заключается в том, что фейеровский процесс, используемый для приближенного вычисления метрической проекции, имеет низкую линейную скорость сходимости:

$$\left\| \boldsymbol{x}^{(k+1)} - \boldsymbol{\varphi}_M\left(\boldsymbol{x}^{(0)}\right) \right\| \leqslant C q^k,$$

где $0 < C < \infty$ — некоторая константа, а $q \in (0, 1)$ — параметр, зависящий от углов между гиперплоскостями, соответствующими граням многогранника M. Это означает, что расстояние между соседними приближениями с каждой итерацией уменьшается

в геометрической прогрессии со знаменателем, меньшим единицы. Для малых углов скорость сходимости может падать до значений, близких к нулю. Второй недостаток связан с ограничением на длину «выводящего» вектора $\delta \boldsymbol{e_c}$: когда исходная граничная точка $\boldsymbol{v}$ находится близко к противоположному краю грани, значение δ должно быть достаточно малым, чтобы метрическая проекция не оказалась за пределами этой грани. Это существенно ограничивает точность вычисления метрической проекции.

В работах [14, 15] предложен алгоритм AlFaMove (Along Faces Movement), в котором для решения задачи ЛП предлагается вычислять метрическую проекцию не на многогранник, а на аффинное подпространство L, образующее грань допустимого многогранника M, как это показано на рис. 2. В этом случае выводящий вектор $\delta \boldsymbol{e_c}$

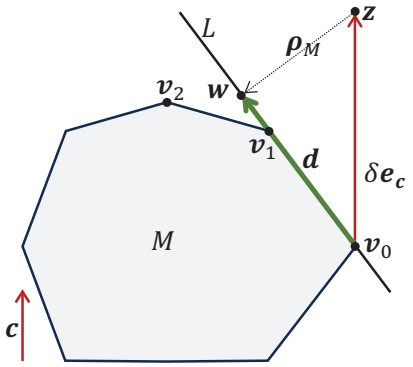


Рис. 2. Алгоритм AlFaMove

может иметь большую длину, что существенно увеличивает точность вычисления направляющего вектора $\boldsymbol{d}$. Для любой грани многогранника M существует комбинация гиперплоскостей из системы ограничений задачи ЛП, пересечением которых будет аффинное подпространство L, образующее эту грань. В соответствии с этим алгоритм AlFaMove перебирает все возможные комбинации гиперплоскостей, проходящих через точку $\boldsymbol{v}_0$. Для каждой комбинации вычисляется свой направляющий вектор $\boldsymbol{d}$. Отбрасываются комбинации, для которых луч $\mathrm{Ray}(\boldsymbol{d}, \boldsymbol{v}_0) = \{\boldsymbol{x} \in \mathbb{R} \mid \boldsymbol{x} = \boldsymbol{v}_0 + \lambda \boldsymbol{d},\ \lambda \in \mathbb{R}_{\geqslant 0}\}$ имеет с многогранником M только одну общую точку:

$$\mathrm{Ray}(\boldsymbol{d}, \boldsymbol{v}_0) \cap M = \boldsymbol{v}_0.$$

Из оставшихся комбинаций выбирается та, у которой следующее приближение $\boldsymbol{v}_1$ дает наибольшее значение целевой функции $f(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle$. Основным недостатком такого подхода является необходимость комбинаторного перебора возможных комбинаций гиперплоскостей, проходящих через точку текущего приближения. Если через точку проходит k гиперплоскостей, то количество комбинаций будет равно 2^k. Таким образом, алгоритм AlFaMove имеет экспоненциальную вычислительную сложность. Это делает затруднительным его применение даже в случае простых многогранников размерности больше 30, у которых из каждой вершины выходит ровно n ребер, где n — размерность пространства решений. Также отметим, что для вычисления метрической проекции алгоритм AlFaMove по-прежнему использует медленные фейеровские процессы.

В недавней работе [16] предложен новый проекционный алгоритм ЛП, получивший название AlEM (Along Edges Movement). В отличие от алгоритма AlFaMove, алгоритм AlEM начинает свою работу в произвольной вершине многогранника допустимых решений и двигается к оптимальной вершине только по ребрам. Для любого ребра существует $n - 1$ гиперплоскостей из системы ограничений задачи ЛП, пересечением которых будет прямая, образующая это ребро. Если через вершину проходит k

гиперплоскостей ($k \geqslant n$), то количество комбинаций может быть вычислено по следующей формуле:

$$C_k^{n-1} = \frac{k!}{(n-1)!(k-n+1)!}.$$

Для вычисления метрической проекции точки на прямую алгоритм AlEM вместо фейеровских отображений использует формулу ортогональной проекции на аффинное подпространство [17], что оказывается значительно более эффективным. Вычислительные эксперименты показали, что быстродействие алгоритма AlEM на задачах ЛП, область допустимых решений которых является простым многогранником, сравнимо с быстродействием симплекс-метода. Однако, в отличии от симплекс-метода, алгоритм AlEM допускает эффективное распараллеливание и исключает возможность зацикливания на вырожденных задачах. К сожалению, в реальных задачах ЛП количество гиперплоскостей, проходящих через вершины многогранника допустимых решений, существенно превышает размерность пространства, что приводит к необходимости комбинаторного перебора.

В этой статье мы предлагаем новый проекционный алгоритм HAlEM (Hybrid Along Edges Movement), который совмещает в себе свойства алгоритма AlEM и симплекс-метода. Принцип работы алгоритма HAlEM заключается в следующем. В начальной вершине $\boldsymbol{v}_0$ формируется матричный базис A_0 из n линейно независимых строк матрицы A, соответствующих гиперплоскостям, проходящим через $\boldsymbol{v}_0$. Эти гиперплоскости образуют n различных ребер. Для каждого ребра с помощью ортогональной проекции вычисляется направляющий вектор. Затем по схеме алгоритма AlEM выбирается ребро, приводящее в вершину $\boldsymbol{v}_1$ с наибольшим значением целевой функции. На основе матричного базиса A_0 формируется новый матричный базис A_1 путем замены строки, не участвовавшей в формировании ребра перехода, на некоторую строку из матрицы A, соответствующую гиперплоскости, проходящей через вершину $\boldsymbol{v}_1$, так, чтобы сохранялась линейная независимость матричного базиса A_1. Процесс продолжается до тех пор, пока не будет достигнута вершина, у которой отсутствуют ребра, ведущие к увеличению целевой функции. Эта вершина будет искомым решением задачи ЛП.

Статья организована следующим образом. Раздел 2 содержит обозначения, определения и утверждения, необходимые для описания алгоритма HAlEM и его численной реализации. В разделе 3 дается формализованное описание алгоритма HAlEM. Раздел 4 посвящен параллельной версии алгоритма HAlEM. В разделе 5 представлены информация о программной реализации алгоритма HAlEM и результаты вычислительных экспериментов. В заключении суммируются полученные результаты и намечаются направления дальнейших исследований.

2. Определения и обозначения

В этом разделе приводятся основные определения и обозначения, используемые для описания алгоритма HAlEM.

В евклидовом пространстве $\mathbb{R}^n$ размерности $n > 1$ имеется задача ЛП общего вида с системой из m ограничений, включающей в себя k линейных уравнений и $m - k$ линейных неравенств:

$$\begin{aligned}
&\langle \boldsymbol{a}_1, \boldsymbol{x} \rangle = b_1, \\
&\cdots\cdots\cdots \\
&\langle \boldsymbol{a}_k, \boldsymbol{x} \rangle = b_k, \\
&\langle \boldsymbol{a}_{k+1}, \boldsymbol{x} \rangle \leqslant b_{k+1}, \\
&\cdots\cdots\cdots \\
&\langle \boldsymbol{a}_m, \boldsymbol{x} \rangle \leqslant b_m,
\end{aligned} \tag{1}$$

Здесь и далее $\langle *, * \rangle$ обозначает скалярное произведение двух векторов. Предполагается, что система (1) включает в себя также неравенства вида

$$\begin{aligned}
&-x_1 \leqslant 0, \\
&\cdots\cdots\cdots \\
&-x_n \leqslant 0.
\end{aligned} \tag{2}$$

Необходимо найти точку в области допустимых решений системы (1), в которой достигается максимум линейной целевой функции

$$F(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x} \rangle .$$

Для $J = \{j_1, \ldots, j_q\} \subseteq \{1, \ldots, m\}$, $|J| = q > 0$ определим матрицу

$$A_J = \begin{bmatrix} \boldsymbol{a}_{j_1} \\ \vdots \\ \boldsymbol{a}_{j_q} \end{bmatrix}$$

и столбец

$$\boldsymbol{b}_J = \begin{bmatrix} b_{j_1} \\ \vdots \\ b_{j_q} \end{bmatrix}.$$

Обозначим через $\bar{I}$ множество индексов уравнений системы (1), и через $\hat{I}$ — множество индексов неравенств системы (1):

$$\begin{aligned}
&\bar{I} = \{1, \ldots, k\}, \\
&\hat{I} = \{k+1, \ldots, m\}.
\end{aligned}$$

Без ограничения общности мы можем считать, что матрица $A_{\bar{I}}$ имеет полный ранг:

$$\operatorname{rank}(A_{\bar{I}}) = k. \tag{3}$$

Мы также будем полагать, что

$$k < n,$$

так как в противном случае область допустимых решений вырождается в точку.

Известно, что область допустимых решений системы (1) представляет собой замкнутый выпуклый многогранник

$$M = \{\boldsymbol{x} \in \mathbb{R}^n | A_{\bar{I}}\boldsymbol{x} = \boldsymbol{b}_{\bar{I}},\ A_{\hat{I}}\boldsymbol{x} \leqslant \boldsymbol{b}_{\hat{I}}\},$$

называемый допустимым. Мы будем предполагать, что допустимый многогранник M является ограниченным множеством. В этом случае задача ЛП всегда имеет решение. Поскольку система (1) включает в себя неравенства (2), то из ограниченности допустимого многогранника M следует, что общее число неравенств должно превышать размерность пространства:

$$m - k > n. \tag{4}$$

Каждому индексу $i \in \bar{I} \cup \hat{I}$ соответствует гиперплоскость

$$H_i = \{\boldsymbol{x} \in \mathbb{R}^n \mid \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle = b_i\}. \tag{5}$$

В совокупности эти гиперплоскости образуют грани допустимого многогранника M.

Базисным индексом вершины $\boldsymbol{v}$ будем называть множество индексов $\mathcal{V}$, удовлетворяющее следующим условиям:

$$\mathcal{V} \subseteq \hat{I}, \tag{6}$$
$$|\mathcal{V}| = n - k, \tag{7}$$
$$\forall i \in \mathcal{V} : \langle \boldsymbol{a}_i, \boldsymbol{v}\rangle = b_i, \tag{8}$$
$$\operatorname{rank}(A_{\bar{I}\cup\mathcal{V}}) = n. \tag{9}$$

Так как $\left|\bar{I}\right| = k$, из (7) и (9) следует, что матрица $A_{\bar{I}\cup\mathcal{V}}$ является квадратной матрицей размера $n \times n$ и имеет полный ранг. Будем такую матрицу называть базисной. С геометрической точки зрения это означает, что пересечение гиперплоскостей, соответствующих строкам базисной матрицы $A_{\bar{I}\cup\mathcal{V}}$, образует вершину $\boldsymbol{v}$:

$$\{\boldsymbol{v}\} = \bigcap_{j\in\bar{I}\cup\mathcal{V}} H_j.$$

Базисная матрица $A_{\bar{I}\cup\mathcal{V}}$ задает $n - k$ различных реберных прямых[1], проходящих через вершину $\boldsymbol{v}$:

$$\forall i \in \mathcal{V} :\ L_i = \bigcap_{j\in\bar{I}\cup\mathcal{E}_i} H_j, \tag{10}$$

где

$$\mathcal{E}_i = \mathcal{V} \setminus \{i\}.$$

Множество индексов $\mathcal{E}_i$, однозначно определяющих i-тую реберную прямую, будем называть реберным индексом. Реберную прямую L_i ($i \in \bar{I} \cup \mathcal{V}$) удобно представлять в параметрическом виде

$$L_i = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \boldsymbol{v} + \lambda\boldsymbol{d},\ \lambda \in \mathbb{R}\}.$$

Направляющий вектор $\boldsymbol{d} \in \mathbb{R}^n$ можно получить следующим образом. Выберем произвольный ненулевой «выводящий» вектор $\boldsymbol{g}$, не являющийся ортогональным по отношению к прямой L_i, и вычислим вторую точку $\boldsymbol{w} \neq \boldsymbol{v}$ на прямой L_i:

$$\boldsymbol{w} = \boldsymbol{\pi}_{\bar{I}\cup\mathcal{E}_i}(\boldsymbol{v} + \boldsymbol{g}).$$

[1]Реберная прямая — это прямая, которой принадлежит некоторое ребро многогранника.

Здесь $\boldsymbol{\pi}_{\bar{I}\cup\mathcal{E}_i}(*)$ обозначает ортогональную проекцию на подпространство[2] $\bigcap\limits_{j\in\bar{I}\cup\mathcal{E}_i} H_j$, образующее реберную прямую L_i в соответствии с формулой (10). Ортогональная проекция на подпространство вычисляется через псевдообратную матрицу по известной формуле [16]:

$$\boldsymbol{\pi}_{\bar{I}\cup\mathcal{E}_i}(\boldsymbol{x}) = \boldsymbol{x} - A^T_{\bar{I}\cup\mathcal{E}_i}\left(A_{\bar{I}\cup\mathcal{E}_i}A^T_{\bar{I}\cup\mathcal{E}_i}\right)^{-1}\left(A_{\bar{I}\cup\mathcal{E}_i}\boldsymbol{x} - \boldsymbol{b}_{\bar{I}\cup\mathcal{E}_i}\right). \tag{11}$$

Заметим, что матрица $A_{\bar{I}\cup\mathcal{E}_i}$ имеет полнострочный ранг в силу того, что базисная матрица $A_{\bar{I}\cup\mathcal{V}}$ имеет полный ранг и $\mathcal{E}_i \subset \mathcal{V}$. Поэтому матрица $A_{\bar{I}\cup\mathcal{E}_i}A^T_{\bar{I}\cup\mathcal{E}_i}$ является обратимой (см. утверждение 3F в [17]). В итоге получаем искомый направляющий вектор

$$\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}.$$

Пусть в контексте системы ограничений (1) заданы точка $\boldsymbol{v} \in M$, произвольный вектор $\boldsymbol{d} \in \mathbb{R}^n$ и полупространство

$$P_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle \leqslant b_i\},$$

ограниченное гиперплоскостью

$$H_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle = b_i\},$$

где $i \in \hat{I}$. Определим луч, исходящий из точки $\boldsymbol{v}$ в направлении вектора $\boldsymbol{d}$:

$$X = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \boldsymbol{v} + \lambda\boldsymbol{d},\ \lambda \in \mathbb{R}_{\geqslant 0}\}.$$

Тогда

$$\text{если} \quad \langle \boldsymbol{a}_i, \boldsymbol{d}\rangle \leqslant 0, \quad \text{то} \quad X \subset P_i, \tag{12}$$

$$\text{если} \quad \langle \boldsymbol{a}_i, \boldsymbol{d}\rangle > 0, \quad \text{то} \quad X \cap H_i = \left\{\boldsymbol{v} + \frac{b_i - \langle \boldsymbol{a}_i, \boldsymbol{v}\rangle}{\langle \boldsymbol{a}_i, \boldsymbol{d}\rangle}\boldsymbol{d}\right\}. \tag{13}$$

Другими словами, луч, исходящий из принадлежащей допустимому многограннику M точки $\boldsymbol{v}$ в направлении $\boldsymbol{d}$, пересечет гиперплоскость H_i в том и только в том случае, когда $\langle \boldsymbol{a}_i, \boldsymbol{d}\rangle > 0$. При этом точка пересечения может быть вычислена по формуле 13.

Пусть в контексте системы ограничений (1) заданы точка $\boldsymbol{v} \in M$ и вектор $\boldsymbol{d} \in \mathbb{R}^n$ такой, что

$$\forall i \in \bar{I}: \ \langle \boldsymbol{a}_i, \boldsymbol{d}\rangle = 0. \tag{14}$$

Определим луч, исходящий из точки $\boldsymbol{v}$ в направлении вектора $\boldsymbol{d}$:

$$X = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \boldsymbol{v} + \lambda\boldsymbol{d},\ \lambda \in \mathbb{R}_{\geqslant 0}\}. \tag{15}$$

Тогда

$$X \cap M = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \lambda\boldsymbol{v} + (1-\lambda)\boldsymbol{q},\ 0 \leqslant \lambda \leqslant 1\},$$

где

$$\boldsymbol{q} = \boldsymbol{v} + \lambda_M\boldsymbol{d}, \tag{16}$$

$$\lambda_M = \min\left\{\frac{b_i - \langle \boldsymbol{a}_i, \boldsymbol{v}\rangle}{\langle \boldsymbol{a}_i, \boldsymbol{d}\rangle}\middle| i \in \hat{I},\ \langle \boldsymbol{a}_i, \boldsymbol{d}\rangle > 0\right\}. \tag{17}$$

[2]Здесь и далее мы опускаем прилагательное «аффинное».

Другими словами, пересечение луча X и допустимого многогранника M представляет собой отрезок между точками $\boldsymbol{v}$ и $\boldsymbol{q}$, где $\boldsymbol{q}$ вычисляется по формулам (16) и (17).

Пусть в контексте системы ограничений (1) имеется вершина $\boldsymbol{v}' \in M$, являющаяся начальной точкой ребра, соответствующего реберной прямой

$$L = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \boldsymbol{v}' + \lambda \boldsymbol{d},\ \lambda \in \mathbb{R}\}.$$

Тогда вершина $\boldsymbol{v}'' \in M$, являющаяся конечной точкой этого ребра, может быть вычислена по формуле

$$\boldsymbol{v}'' = \boldsymbol{v}' + \frac{b_{j^*} - \langle \boldsymbol{a}_{j^*}, \boldsymbol{v} \rangle}{\langle \boldsymbol{a}_{j^*}, \boldsymbol{d} \rangle} \boldsymbol{d}, \tag{18}$$

где

$$j^* \in \arg\min \left\{ \frac{b_j - \langle \boldsymbol{a}_j, \boldsymbol{v} \rangle}{\langle \boldsymbol{a}_j, \boldsymbol{d} \rangle} \;\middle|\; j \in \hat{I},\ \langle \boldsymbol{a}_j, \boldsymbol{d} \rangle > 0 \right\}. \tag{19}$$

Гиперплоскость H_{j^*} в этом случае называется ограничивающей по отношению к реберной прямой L.

Пусть в контексте системы ограничений (1) имеется вершина $\boldsymbol{v}' \in M$ с базисным индексом $\mathcal{V}'$. Зафиксируем некоторый $i^* \in \mathcal{V}'$. Ему соответствует реберная прямая L_{i^*}, вычисляемая по формуле

$$L_{i^*} = \bigcap_{j \in \bar{I} \cup \mathcal{V}' \setminus \{i^*\}} H_j, \tag{20}$$

Пусть также известен направляющий вектор $\boldsymbol{d}$ прямой L_{i^*}:

$$L_{i^*} = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \boldsymbol{v}' + \lambda \boldsymbol{d},\ \lambda \in \mathbb{R}\}. \tag{21}$$

Пусть точка $\boldsymbol{v}''$ является конечной вершиной рассматриваемого ребра. Положим

$$J^* = \arg\min \left\{ \frac{b_j - \langle \boldsymbol{a}_j, \boldsymbol{v} \rangle}{\langle \boldsymbol{a}_j, \boldsymbol{d} \rangle} \middle| j \in \hat{I},\ \langle \boldsymbol{a}_j, \boldsymbol{d} \rangle > 0 \right\}. \tag{22}$$

Тогда для любого $j^* \in J^*$ множество

$$\mathcal{V}'' = (\mathcal{V}' \setminus \{i^*\}) \cup \{j^*\} \tag{23}$$

будет являться базисным индексом для вершины v''.

3. Алгоритм HAlEM

Данный раздел посвящен формализованному описанию алгоритма HAlEM, строящего из произвольной вершины $\boldsymbol{v}$ путь вдоль ребер допустимого многогранника M к вершине, в которой достигается максимум целевой функции задачи ЛП. Псевдокод алгоритма HAlEM представлен в виде алгоритма 1.

Поясним шаги алгоритма 1. На шаге 1 вводятся размерность пространства n, количество ограничений m, количество уравнений k в системе ограничений, матрица коэффициентов A и столбец свободных членов (правых частей) $\boldsymbol{b}$ системы ограничений, градиент $\boldsymbol{c}$ линейной целевой функции и координаты начальной вершины $\boldsymbol{v}_0$ допустимого многогранника M. Эти данные являются глобальными в том смысле, что они доступны всем подпрограммам-функциям, используемым в алгоритме 1.

Предполагается, что ограничения задачи ЛП приведены к виду (1). Также предполагается, что среди уравнений нет избыточных, то есть $\mathrm{rank}(A_{\bar{I}}) = k$, где k — число уравнений. Допускается случай, когда $k = 0$. Шаги 2 и 3 формируют множество индексов уравнений $\bar{I}$ и множество индексов неравенств $\hat{I}$, составляющих систему ограничений (1). Шаг 4 с помощью функции BasicIndex формирует базисный индекс $\mathcal{V}_0 \subset \hat{I}$ для начальной вершины $\boldsymbol{v}_0$. Такой базисный индекс существует в силу (4). Псевдокод функции BasicIndex приведен в приложении в виде алгоритма 4. Шаг 5 устанавливает счетчик итераций t в значение 0. На шаге 6 логической переменной *exit* присваивается значение «ложь».

Основной цикл **repeat/until** (шаги 7–36) выполняет последовательные переходы от вершины к вершине по ребрам допустимого многогранника, пока не будет достигнута оптимальная вершина. Один переход соответствует одной итерации этого цикла.

Вложенный цикл **loop** (шаги 8–35) выполняет переход от вершины $\boldsymbol{v}_t$ к следующей вершине $\boldsymbol{v}_{t+1}$ с бóльшим значением целевой функции. Для этого на шаге 9 точке $\boldsymbol{v}^*$ присваиваются координаты текущей вершины $\boldsymbol{v}_t$.

Следующий за этим цикл **for all** (шаги 10–22) находит для текущей вершины $\boldsymbol{v}_t$ ребро с индексом i^*, ведущее к вершине $\boldsymbol{v}^*$ с наибольшим значением целевой функции среди всех ребер с индексами из $\mathcal{V}_t$. Прокомментируем шаги в теле этого цикла. Шаг 11 с помощью функции DirectionVector вычисляется направляющий вектор $\boldsymbol{d}$ текущей реберной прямой L_i в направлении увеличения целевой функции. Псевдокод функции DirectionVector приведен в приложении в виде алгоритма 5. Оператор **if** на шаге 12 проверяет выполнение условия $\boldsymbol{d} = \boldsymbol{0}$. В этом случае на шаге 13 выполняется оператор **continue**, осуществляющий досрочный переход к следующей итерации цикла **for all**. В противном случае на шаге 15 с помощью функции BoundigHyperplaneIndex вычисляется индекс j гиперплоскости H_j, ограничивающий текущую реберную прямую L_i. Псевдокод функции BoundigHyperplaneIndex приведен в приложении в виде алгоритма 6. Шаг 16 вычисляет конечную вершину $\boldsymbol{v}_{nex}$ текущего ребра в соответствии с формулой (18). Если в операторе **if** на шаге 17 значение целевой функции в вершине $\boldsymbol{v}_{nex}$ оказывается больше значения целевой функции в вершине $\boldsymbol{v}^*$, то вершине $\boldsymbol{v}^*$ присваиваются координаты вершины $\boldsymbol{v}_{nex}$ (шаг 18), индекс i текущей реберной прямой сохраняется в переменной i^* (шаг 19), а индекс соответствующей ограничивающей гиперплоскости H_j сохраняется в переменной j^* (шаг 20). Шаг 21 закрывает оператор **if**. Шаг 22 заканчивает цикл **for all**.

Если в операторе **if** на шаге 23 удалось получить вершину $\boldsymbol{v}^*$ с координатами отличными от текущей вершины $\boldsymbol{v}_t$, то выполняются следующие действия. На шаге 24 эта вершина становится следующим приближением $\boldsymbol{v}_{t+1}$. На шаге 25 для новой вершины $\boldsymbol{v}_{t+1}$ вычисляется базисный индекс $\mathcal{V}_{t+1}$ по формуле (23). Шаг 26 увеличивает на единицу счетчик итераций t. На шаге 27 оператор **break** осуществляет выход из цикла **loop** (шаг 35), после чего выполняется следующая итерация основного цикла **repear/until**.

Если условие оператора **if** на шаге 23 не выполняется, происходит переход к шагу 29. Эта ситуация возникает, когда базисный индекс $\mathcal{V}_t$ оказался «тупиковым»: в нем отсутствуют ребра, ведущие к увеличению целевой функции. В этом случае на шаге 29 выполняется «прокручивание» базисного индекса вершины $\boldsymbol{v}_t$ с помощью функции ScrollBasicIndex, псевдокод которой приведен в приложении в виде алгоритма 7. В результате получается обновленный базисный индекс $\mathcal{V}'_t \neq \mathcal{V}_t$. В этом случае оператор **if** (шаги 30-33) пропускается, и выполняется шаг 34, который заменяет предыдущий базисный индекс на новый. После этого повторяется итерация цикла **loop** для текущей вершины $\boldsymbol{v}_t$, но уже с обновленным базисным индексом.

Алгоритм 1 Алгоритм HAlEM

1: **input** n, m, k, A, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{v}_0$
2: $\bar{I} := \{1, \ldots, k\}$
3: $\hat{I} := \{k+1, \ldots, m\}$
4: $\mathcal{V}_0 := \text{BasicIndex}(\boldsymbol{v}_0)$ // см. алгоритм 4
5: $t := 0$
6: $exit := \textbf{false}$
7: **repeat**
8: **loop**
9: $\boldsymbol{v}^* := \boldsymbol{v}_t$
10: **for all** $i \in \mathcal{V}_t$ **do**
11: $\boldsymbol{d} := \text{DirectionVector}(\boldsymbol{v}_t, \mathcal{V}_t \setminus \{i\})$ // см. алгоритм 5
12: **if** $\boldsymbol{d} = \boldsymbol{0}$ **then**
13: **continue**
14: **end if**
15: $j := \text{BoundigHyperplaneIndex}(\boldsymbol{v}_t, \boldsymbol{d})$ // см. алгоритм 6
16: $\boldsymbol{v}_{nex} := \boldsymbol{v}_t + \frac{b_j - \langle \boldsymbol{a}_j, \boldsymbol{v}_t \rangle}{\langle \boldsymbol{a}_j, \boldsymbol{d} \rangle} \boldsymbol{d}$
17: **if** $\langle \boldsymbol{c}, \boldsymbol{v}_{nex} \rangle > \langle \boldsymbol{c}, \boldsymbol{v}^* \rangle$ **then**
18: $\boldsymbol{v}^* := \boldsymbol{v}_{nex}$
19: $i^* := i$
20: $j^* := j$
21: **end if**
22: **end for**
23: **if** $\boldsymbol{v}^* \neq \boldsymbol{v}_t$ **then**
24: $\boldsymbol{v}_{t+1} := \boldsymbol{v}^*$
25: $\mathcal{V}_{t+1} := (\mathcal{V}_t \setminus \{i^*\}) \cup \{j^*\}$
26: $t := t + 1$
27: **break**
28: **end if**
29: $\mathcal{V}'_t := \text{ScrollBasicIndex}(\boldsymbol{v}_t, \mathcal{V}_t)$ // см. алгоритм 7
30: **if** $\mathcal{V}'_t = \mathcal{V}_t$ **then**
31: $exit := \textbf{true}$
32: **break**
33: **end if**
34: $\mathcal{V}_t := \mathcal{V}'_t$
35: **end loop**
36: **until** *exit*
37: **output** $\boldsymbol{v}_t$
38: **stop**

Если обновить базисный индекс невозможно, функция ScrollBasicIndex возвращает исходный базисный индекс. Это означает, что оптимальная вершина уже достигнута. В этом случае выполняется условие в операторе **if** на шаге 30, что приводит к выполнению шага 31, который устанавливает логическую переменную *exit* в значение «истина». После этого оператор **break** на 32 шаге производит выход из цикла **loop**. Поскольку $exit = \textbf{true}$, оператор **until** на шаге 36 завершает выполнение основного цикла **repeat**/**until**. Шаг 37 выводит координаты оптимальной вершины. Шаг 38 завершает работу алгоритма HAlEM.

4. Параллельная версия алгоритма HAlEM

Алгоритм 1 перебирает в цикле **for all** (шаги 10–22) $n-k$ ребер базисного индекса $\mathcal{V}_t$ с целью нахождения ребра, ведущего к вершине с наибольшим значением целевой функции. Этот цикл поддается эффективному распараллеливанию, что мы реализовали в параллельной версии алгоритма HAlEM.

Параллельная версия алгоритма HAlEM основана на модели параллельных вычислений BSF [18], ориентированной на кластерные вычислительные системы. Модель BSF использует схему распараллеливания «мастер–рабочие» и требует представление алгоритма в виде операций над списками с использованием функций высшего порядка Map и Reduce.

Функция высшего порядка Map(F, *MapList*) выполняет функцию F для всех элементов списка *MapList*, в результате чего формируется список *ReduceList*. Функция высшего порядка Reduce($\oplus$, *ReduceList*) попарно выполняет ассоциативную бинарную операцию $\oplus$ для всех элементов списка *MapList*, в результате чего получается один элемент.

В данном случае список *MapList* имеет длину $n-k$ и включает в себя все элементы базисного индекса $\mathcal{V}_t$, взятые в определенном порядке:

$$MapList = [i_1, \ldots, i_{n-k}], \tag{24}$$

где $\{i_1, \ldots, i_{n-k}\} = \mathcal{V}_t$. Семантика функции F_t определяется алгоритмом 2, шаги которого соответствуют шагам 11– 16 алгоритма 1.

Алгоритм 2 Функция F_t

1: **function** $\mathrm{F}_t(i)$
2: $\quad \boldsymbol{d} := \mathrm{DirectionVector}(\boldsymbol{v}_t, \mathcal{V}_t \setminus \{i\})$ // см. алгоритм 5
3: $\quad$ **if** $\boldsymbol{d} = \boldsymbol{0}$ **then**
4: $\quad\quad$ **return** $(\boldsymbol{v}_t, i, 0)$
5: $\quad$ **end if**
6: $\quad j := \mathrm{BoundigHyperplaneIndex}(\boldsymbol{v}_t, \boldsymbol{d})$ // см. алгоритм 6
7: $\quad \boldsymbol{v}^* := \boldsymbol{v}_t + \frac{b_j - \langle \boldsymbol{a}_j, \boldsymbol{v}_t \rangle}{\langle \boldsymbol{a}_j, \boldsymbol{d} \rangle} \boldsymbol{d}$
8: $\quad$ **return** $(\boldsymbol{v}^*, i, j)$
9: **end function**

Список *ReduceList*, получаемый в результате вызова функции высшего порядка Map(F_t, *MapList*), будет выглядеть следующим образом:

$$ReduceList = \left[(\boldsymbol{v}_1^*, i_1^*, j_1^*), \ldots, (\boldsymbol{v}_{n-k}^*, i_{n-k}^*, j_{n-k}^*)\right],$$

где для любого $l \in \{1, \ldots, n-k\}$ точка $\boldsymbol{v}_l^*$ является смежной вершиной по отношению к вершине $\boldsymbol{v}_t$, i_l^* — индекс соответствующего ребра, j_l^* — индекс гиперплоскости, ограничивающий соответствующую реберную прямую, за исключением случая, когда $j_l^* = 0$. В этом случае $\boldsymbol{v}_l^* = \boldsymbol{v}_t$.

Бинарная операция $\oplus$ над элементами списка *ReduceList* определяется следующей формулой:

$$(\boldsymbol{v}', i', j') \oplus (\boldsymbol{v}'', i'', j'') = \begin{cases} (\boldsymbol{v}', i', j'), & \textbf{if } \langle \boldsymbol{c}, \boldsymbol{v}' \rangle > \langle \boldsymbol{c}, \boldsymbol{v}'' \rangle, \\ (\boldsymbol{v}'', i'', j''), & \textbf{if } \langle \boldsymbol{c}, \boldsymbol{v}' \rangle \leqslant \langle \boldsymbol{c}, \boldsymbol{v}'' \rangle, \end{cases} \tag{25}$$

семантика которой соответствует шагам 17–21 алгоритма 1.

Алгоритм 3 Параллельная версия алгоритма HAlEM

мастер		**l-тый рабочий** $(l = 1, \ldots, L)$	
1:	**input** n, m, k, A, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{v}_0$	1:	**input** n, m, k, A, $\boldsymbol{b}$, $\boldsymbol{c}$
2:	$\bar{I} := \{1, \ldots, k\}$	2:	$L :=$ **NumberOfWorkers**
3:	$\hat{I} := \{k+1, \ldots, m\}$	3:	$l :=$ **MyNumber**
4:	$\mathcal{V}_0 := \mathrm{BasicIndex}(\boldsymbol{v}_0)$	4:	
5:	$t := 0$	5:	
6:	$exit :=$ **false**	6:	
7:	**repeat**	7:	**repeat**
8:	**loop**	8:	
9:	**Broadcast**$(\boldsymbol{v}_t, \mathcal{V}_t)$	9:	**Recv**$(\boldsymbol{v}_t, \mathcal{V}_t)$ // $\mathcal{V}_t = \{i_1, \ldots, i_{n-k}\}$
10:		10:	$K := \frac{n-k}{L}$
11:		11:	$MapList_l := \left[i_{1+(l-1)K}, \ldots, i_{lK}\right]$
12:		12:	$ReduceList_l := \mathrm{Map}(\mathrm{F}_t, MapList_l)$
13:		13:	$(\boldsymbol{v}^*, i^*, j^*) := \mathrm{Reduce}(\oplus, ReduceList_l)$
14:	**Gather**$(ReduceList^*)$	14:	**Send**$(\boldsymbol{v}^*, i^*, j^*)$
15:	$(\boldsymbol{v}^*, i^*, j^*) := \mathrm{Reduce}(\oplus, ReduceList^*)$	15:	
16:	**if** $\boldsymbol{v}^* \neq \boldsymbol{v}_t$ **then**	16:	
17:	$\boldsymbol{v}_{t+1} := \boldsymbol{v}^*$	17:	
18:	$\mathcal{V}_{t+1} := (\mathcal{V}_t \setminus \{i^*\}) \cup \{j^*\}$	18:	
19:	$t := t + 1$	19:	
20:	**break**	20:	
21:	**end if**	21:	
22:	$\mathcal{V}'_t := \mathrm{ScrollBasicIndex}(\boldsymbol{v}_t, \mathcal{V}_t)$	22:	
23:	**if** $\mathcal{V}'_t = \mathcal{V}_t$ **then**	23:	
24:	$exit :=$ **true**	24:	
25:	**break**	25:	
26:	**end if**	26:	
27:	$\mathcal{V}_t := \mathcal{V}'_t$	27:	
28:	**end loop**	28:	
29:	**Broadcast**$(exit)$	29:	**Recv**$(exit)$
30:	**until** $exit$	30:	**until** $exit$
31:	**output** $\boldsymbol{v}_t$	31:	
32:	**stop**	32:	**stop**

Псевдокод параллельной версии алгоритма HAlEM представлен в виде алгоритма 3. Распараллеливанию подвергается цикл **for all** (шаги 10–22) последовательного алгоритма 1. Параллельные вычисления организуются по схеме «мастер–рабочие» и включают в себя $L + 1$ процесс: один процесс–мастер и L процессов–рабочих, где $L \leqslant n - k$. Для простоты мы будем предполагать, что $n - k$ кратно L.

Процесс–мастер (далее — просто «мастер») выполняет последовательную часть алгоритма 3 и организует выполнение параллельной части процессами–рабочими (далее — просто «рабочие»). Шаги 1–8 мастера соответствуют шагам 1–8 последовательного алгоритма 1. На шаге 9 мастер с помощью системной функции **Broadcast** рассылает всем рабочим координаты текущей вершины $\boldsymbol{v}_t$ и соответствующий базисный индекс

$\mathcal{V}_t$. На шаге 14 мастер с помощью системной функции **Gather** собирает результаты всех рабочих, представленных в виде троек вида $(\boldsymbol{v}^*, i^*, j^*)$, и организует их в единый список *ReduceList**. На шаге 15 мастер с помощью функции высшего порядка Reduce редуцирует список *ReduceList** к одной тройке $(\boldsymbol{v}^*, i^*, j^*)$, путем попарного применения бинарной операции $\oplus$ по формуле (25). Далее мастер выполняет шаги 16–28, соответствующие шагам 23–35 последовательного алгоритма 1. На шаге 29 мастер с помощью системной функции **Broadcast** рассылает всем рабочим значение логической переменной *exit*, в зависимости от которого рабочие продолжают или заканчивают свою работу. Финальные шаги 30–32 мастера соответствуют финальным шагам 36–38 последовательного алгоритма 1.

Все рабочие выполняют одну и ту же последовательность шагов, но над разными данными. На шаге 1 каждый рабочий вводит исходные данные задачи ЛП. На шаге 2 с помощью системной функции **NumberOfWorkers** рабочий получает общее число рабочих L, задействованных в вычислениях. На шаге 3 с помощью системной функции **MyNumber** рабочий получает свой номер l. Далее рабочий входит в вычислительный цикл **repeat**/**until** (шаги 7–30). На шаге 9 рабочий с помощью системной функции **Recv** ожидает получения от мастера координат текущей вершины $\boldsymbol{v}_t$ и элементов текущего базисного индекса $\mathcal{V}_t$. На шаге 10 рабочий вычисляет длину K той части списка *MapList*, которую ему необходимо обработать. На шаге 11 рабочий строит свою часть списка $MapList_l$. На шаге 12 с помощью функции высшего порядка Map рабочий вычисляет список $ReduceList_l$. На шаге 13 рабочий с помощью функции высшего порядка Reduce редуцирует список $ReduceList_l$ к одной тройке $(\boldsymbol{v}^*, i^*, j^*)$, путем попарного применения бинарной операции $\oplus$ по формуле (25). На шаге 14 с помощью системной функции **Send** рабочий отсылает полученную тройку мастеру. После этого рабочий на шаге 29 с помощью системной функции **Recv** ожидает получения от мастера значения логической переменной *exit*. Если на шаге 30 переменная *exit* принимает значение **false**, рабочий переходит к следующей итерации вычислительного цикла **repeat**/**until**. В противном случает рабочий заканчивает свою работу.

Отметим, что с помощью системных функций **Recv** и **Gather** осуществляется неявная синхронизация параллельных процессов.

5. Реализация и вычислительные эксперименты

Мы реализовали параллельную версию алгоритма HAlEM на языке C++ с использованием программного BSF-каркаса [20], базирующегося на модели параллельных вычислений BSF [19]. BSF-каркас инкапсулирует все аспекты, связанные с распараллеливанием программы на основе библиотеки MPI. Исходные коды параллельной реализации свободно доступны в репозитории GitVerse по адресу `https://gitverse.ru/sokolinsky/HAlEM`. С использованием этой реализации были проведены вычислительные эксперименты по исследованию масштабируемости параллельной версию алгоритма HAlEM на различных задачах ЛП. Все эксперименты проводились на суперкомпьютере «Торнадо ЮУрГУ» [21], характеристики которого представлены в табл. 1. Для сборки программы использовался компилятор g++, распространяемый в рамках пакета компиляторов GCC 10, и библиотека Intel MPI 5.0. Компиляция выполнялась с опцией оптимизации O3. Задачи запускались на различном количестве процессорных узлов кластера. При этом общее число MPI-процессов было равно $12K$, где K — количество задействованных процессорных узлов. Таким образом, на каждом узле работало 12 MPI-процессов — по числу физических ядер.

Таблица 1. Характеристики суперкомпьютера «Торнадо ЮУрГУ»

Параметр	**Значение**
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 cores, 3.33 GHz)
Число процессоров на узел	2
Память на узел	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

В первой серии экспериментов мы исследовали ускорение и эффективность распараллеливания алгоритма HAlEM на эталонных задачах ЛП из репозитория Netlib-LP [22], доступного по адресу `https://netlib.org/lp/data`. В качестве начальной вершины $\boldsymbol{v}_0$ всегда выбиралась точка $\mathbf{0}$, если она являлась вершиной допустимого многогранника M. В противном случае начальная вершина вычислялась с помощью программы VeRSAl (Vertex Retrieve by Simplex Algorithm), написанной на языке C++, исходные тексты которой свободно доступны в репозитории GitVerse по адресу `https://gitverse.ru/sokolinsky/VeRSAl`. Программа VeRSAl строит на основе исходной задачи ЛП расширенную задачу ЛП в соответствии с методом, описанными в [23] (см. «Общий случай», стр. 199). Точка $\mathbf{0}$ по построению является вершиной допустимого многогранника для расширенной задачи ЛП. Программа VeRSAl решает эту задачу с помощью стандартного симплекс-метода, в результате чего получается вершина допустимого многогранника для исходной задачи ЛП. Результаты экспериментов с эталонными задачами из репозитория Netlib-LP представлены в табл. 2. В столбце 1 перечислены имена задач из репозитория Netlib-LP, использованные для тестирования параллельного алгоритма HAlEM. Файлы с исходными данными этих задач в формате MPS [23] скопированы нами в свободно доступный репозиторий GitVerse по адресу `https://gitverse.ru/sokolinsky/Set-of-LP-Problems/content/master/NetLib-LP`. В этом же репозитории в формате MatrixMarket [24] сохранены координаты начальных вершин для всех указанных задач.

В столбце 2 указано количество ограничений m для соответствующей задачи ЛП, включая неравенства, уравнения (если они присутствуют), и ограничения вида (2). В столбце 3 указано количество переменных n (размерность пространства решений). Столбец 4 содержит размерность d_M допустимого многогранника M, совпадающую с размерностью его аффинной оболочки:

$$d_M = \dim(M) = \dim\left(\mathrm{aff}\left(M\right)\right).$$

Если система ограничений не содержит избыточных уравнений и неявных равенств[3], то размерность допустимого многогранника M может быть вычислена по следующей формуле:

$$\dim(M) = n - k,$$

[3]Неравенство $\langle \boldsymbol{a}, \boldsymbol{x}\rangle \leqslant b$ в системе ограничений $A\boldsymbol{x} \leqslant \boldsymbol{b}$ является неявным равенством, если $\langle \boldsymbol{a}, \boldsymbol{x}\rangle = b$ для всех $\boldsymbol{x}$, удовлетворяющих $A\boldsymbol{x} \leqslant \boldsymbol{b}$.

Таблица 2. Тестирование параллельного алгоритма HAlEM на задачах Netlib-LP

Задача	m	n	d_M	d_M/n	δ	K_{peak}	$K_{\max}$	$T_{K_{\max}}$	T_1	α	ϵ
1	2	3	4	5	6	7	8	9	10	11	12
adlittle	153	97	82	0.85	0	6	5	0.55	2.22	4.04	0.81
afiro	59	32	24	0.75	2.5×10^{-16}	2	2	0.0015	0.0027	1.8	0.90
agg	651	163	127	0.78	0	10	6	2.57	10.40	3.84	0.67
agg2	818	302	242	0.80	7.4×10^{-16}	20	8	93.4	616.2	6.6	0.82
beaconfd	435	262	122	0.47	2.2×10^{-16}	10	6	9.18	31.62	3.44	0.57
blend	157	83	40	0.48	1.2×10^{-16}	3	3	0.35	0.52	1.5	0.51
israel	316	142	142	1	1.2×10^{-15}	11	7	5.46	29.09	5.33	0.76
kb2	93	41	25	0.61	0	2	2	0.022	0.030	1.34	0.67
recipe	366	180	92	0.51	0	7	6	2.07	6.94	3.35	0.56
sc105	207	103	58	0.56	0	4	4	0.13	0.32	2.47	0.62
sc50a	97	48	28	0.58	0	2	2	0.008	0.009	1.2	0.62
sc50b	96	48	28	0.58	2×10^{-16}	2	2	0.005	0.007	1.3	0.65
scagr7	269	140	56	0.4	0	4	4	0.81	2.13	2.6	0.66
share2b	175	79	66	0.83	1.5×10^{-15}	5	3	0.18	0.43	2.34	0.78
stocfor1	228	111	48	0.43	1.1×10^{-14}	4	4	0.18	0.43	2.32	0.58

где k — количество уравнений в системе ограничений. В соответствии с этим количество уравнений k в системе ограничений задачи ЛП из табл. 2 может быть вычислено по формуле

$$k = n - d_M. \tag{26}$$

Столбец 5 содержит отношение размерности d_M допустимого многогранника к размерности n пространства решений. В столбце 6 указана относительная погрешность максимума целевой функции, вычисленного с помощью алгоритма HAlEM, в сравнении с «точным» значением, приведенным в работе Коха [25]:

$$\delta = \left| \frac{F_{Koch} - F_{HAlEM}}{F_{Koch}} \right|.$$

Здесь F_{HAlEM} — значение вычисленное алгоритмом HAlEM, F_{Koch} — значение, указанное в работе [26]. Практически во всех случаях относительная погрешность оказалась в пределах машинного эпсилон [27], равного 2.2×10^{-16} для типа double (64 бита). Столь высокая точность вычислений была достигнута благодаря использованию метода двух-факторной проекции в алгоритме 5.

В столбце 7 приведена верхняя теоретическая граница K_{peak} масштабируемости алгоритма HAlEM, которая указывает максимальное число процессорных узлов, доступных для распараллеливания. Величина K_{peak} вычисляется следующим образом. Алгоритм 3 организует параллельные вычисления путем деления списка *MapList* на равные части по числу рабочих. Обработка этих подсписков выполняется рабочими на

шаге 12 независимо друг от друга. Согласно (24) список *MapList* имеет длину $n-k$. Из (26) следует, что длина списка *MapList* равна d_M. Это означает, что для параллельной обработки списка *MapList* нельзя использовать более d_M рабочих. Учитывая, что при прогоне задачи на каждом процессорном узле запускалось 12 рабочих MPI-процессов, приходим к выводу, что

$$K_{\text{peak}} = \lfloor d_M/12 \rfloor .$$

Столбец 8 содержит реальную границу масштабируемости $K_{\max}$, определенную для каждой задачи ЛП в ходе вычислительных экспериментов. Под реальной границей масштабируемости параллельного алгоритма понимается максимальное число процессорных узлов вычислительного кластера, сверх которого время решения задачи перестает сокращаться. В соответствии с характеристиками вычислительной платформы, использованной при проведении экспериментов (см. табл. 1), число процессорных ядер, задействованных для решения задачи на $K_{\max}$ узлах будет равно $K_{\max} \times 12$. В 9 столбце указано время $T_{K_{\max}}$ решения задачи ЛП на $K_{\max}$ узлах[4]. В 10 столбце указано время T_1 решения этой же задачи на одном процессорном узле. Значения ускорения, достигнутые на границе масштабируемости, представлены в 11 столбце. Ускорение вычислялось по формуле

$$\alpha = \frac{T_{K_{max}}}{T_1}.$$

В 12 столбце содержится параллельная эффективность, достигаемая на границе масштабируемости. Параллельная эффективность вычислялась по формуле

$$\epsilon = \frac{T_1}{K_{max} \cdot T_{K_{max}}}.$$

Анализ полученных результатов позволяет сделать несколько выводов. Первый вывод касается границы масштабируемости $K_{\max}$. Для допустимых многогранников малой размерности ($d_M < 70$) теоретическая граница масштабируемости совпадает с реальной:

$$K_{\text{peak}} = K_{\max}.$$

Для допустимых многогранников средней размерности ($70 \leqslant d_M \leqslant 100$) теоретическая и реальная границы масштабируемости различаются не более чем на 1:

$$K_{\text{peak}} - K_{\max} \leqslant 1.$$

В случае допустимых многогранников большой размерности ($d_M \geqslant 70$), реальная граница масштабируемости может быть существенно меньше теоретической:

$$K_{\max} \ll K_{\text{peak}}.$$

Это связано с тем, что при использовании большого количества MPI-процессов для решения задачи ЛП методом HAlEM, накладные расходы на организацию параллельного выполнения могут полностью нивелировать ускорение, получаемое при добавлении новых процессорных узлов.

Второй вывод связан с корреляцией между величиной d_M/n и эффективностью распараллеливания ϵ. Столбчатая диаграмма, представленная на рис. 3 показывает, что большому значению d_M/n соответствует высокая параллельная эффективность ϵ, и наоборот, низкая параллельная эффективность ϵ характерна для задач ЛП с небольшим значением d_M/n. Из этого ряда выпадает задача israel, которая характерна тем,

[4]Время везде указано в секундах.

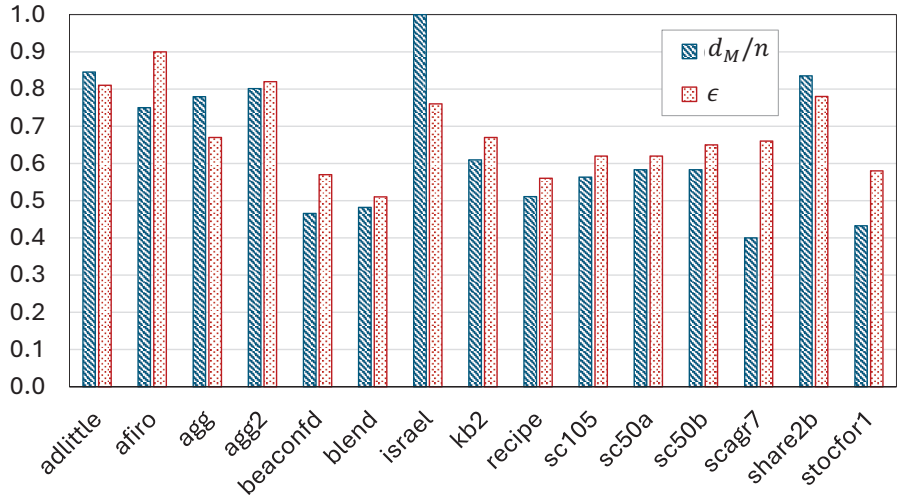


Рис. 3. Корреляция между d_M/n и параллельной эффективностью ϵ

что у нее единственной в системе ограничений отсутствуют уравнения, то есть размерность пространства решений совпадает с размерностью допустимого многогранника. Также отметим, что во всех случаях параллельная эффективность не опускалась ниже 51%, что является неплохим показателем для численного алгоритма.

Для сравнения разработанной параллельной версии алгоритма HAlEM с симплекс-методом мы использовали программу Simplex, исходные тексты которой на языке C++ свободно доступны по адресу `https://gitverse.ru/sokolinsky/Simplex`. Программа Simplex реализует параллельную версию классического симплекс-метода, описанного в [23] (стр. 197–198). Используя эту реализацию, мы решили все упомянутые задачи из репозитория Netlib-LP на той же вычислительной платформе, на которой исследовали HAlEM. Для всех задач ЛП программа Simplex запускалась на двух процессорных узлах по 12 MPI-процессов на узел. Использование большего количества процессорных узлов приводило к деградации ускорения программы Simplex. Результаты сравнения приведены в табл. 3. Эксперименты показали, что метод HAlEM обладает существенно большим ресурсом параллелизма, чем симплекс-метод. По количеству итераций оба метода имеют близкие показатели, однако симплекс-метод превосходит HAlEM по быстродействию на всех исследованных задачах ЛП. При этом разница в быстродействии находится в пределах одного порядка, что не является катастрофическим. Оба метода демонстрируют высокую точность вычислений на границе машинного нуля для типа double (64 бит). Однако существуют классы задач ЛП, на которых метод HAlEM показывает более высокую эффективность, чем симплекс-метод. В качестве примера можно привести циклические многогранники. Используя пример циклического многогранника $C_4(8)$ из [28] (стр. 29–30), мы сконструировали задачу ЛП zieglerC4_8, исходные данные которой в формате MPS можно найти по адресу `https://gitverse.ru/sokolinsky/Set-of-LP-Problems/content/master/Miscellaneous-LP`. Алгоритм HAlEM решает эту задачу за одну итерацию, в то время как программе Simplex для этого требуется 11 итераций.

В финальной серии экспериментов с параллельной версией алгоритма HAlEM мы исследовали зависимость ускорения и параллельной эффективности от размера решаемой задачи. С этой целью мы сконструировали специальную параметризованную задачу ЛП «гиперкуб с отсеченной вершиной», для которой размерность n пространства решений является параметром. Ограничения этой задачи содержат

Таблица 3. Сравнение метода HAlEM с симплекс-методом на задачах Netlib-LP

Задача	Процессорных узлов		Число итераций		Время (с)		Относительная погрешность	
	HAlEM	Simplex	HAlEM	Simplex	HAlEM	Simplex	HAlEM	Simplex
adlittle	5	2	67	59	0.55	0.12	0	3.9×10^{-15}
afiro	2	2	3	4	0.0015	0.0009	2.5×10^{-16}	1.2×10^{-16}
agg	6	2	23	63	2.57	1.06	0	0
agg2	8	2	99	132	93.4	19.3	7.4×10^{-16}	0
beaconfd	6	2	15	23	9.18	2.07	2.2×10^{-16}	2.2×10^{-16}
blend	3	2	35	70	0.35	0.09	1.2×10^{-16}	4.7×10^{-14}
israel	7	2	146	160	5.46	1.26	1.2×10^{-15}	3.9×10^{-15}
kb2	2	2	23	76	0.02	0.02	0	4×10^{-15}
recipe	7	2	12	17	2.07	0.41	0	0
sc105	4	2	13	13	0.13	0.03	0	0
sc50a	2	2	7	7	0.008	0.002	0	0
sc50b	2	2	5	5	0.005	0.002	2×10^{-16}	0
scagr7	4	2	30	30	0.81	0.21	0	1.8×10^{-15}
share2b	3	2	27	33	0.18	0.05	1.5×10^{-15}	2.3×10^{-15}
stocfor1	4	2	12	22	0.18	0.07	1.1×10^{-14}	1.1×10^{-14}

$2n + 1$ неравенств следующего вида:

$$\begin{aligned}
x_1 \qquad\qquad\qquad\qquad &\leqslant 200,\\
x_2 \qquad\qquad\qquad\qquad &\leqslant 200,\\
&\vdots\\
x_n \qquad\qquad &\leqslant 200,\\
x_1 + x_2 + \cdots + x_n &\leqslant 200(n-1) + 100,\\
x_1 \geqslant 0,\quad x_2 \geqslant 0,\quad \cdots\quad , x_n \geqslant 0.&
\end{aligned} \tag{27}$$

Градиент целевой функции задается вектором

$$\boldsymbol{c} = (1, 2, \ldots, n)\,.$$

Задача предполагает нахождение максимума целевой функции и имеет единственное решение в точке $(100, 200, \ldots, 200)$ со значением целевой функции, равным

$$F_{\max}(n) = 100(n^2 + n - 1). \tag{28}$$

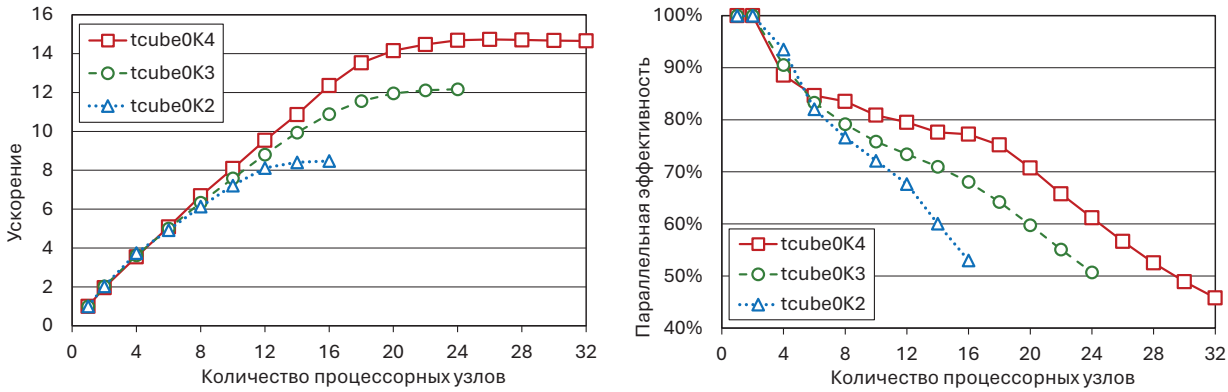


Рис. 4. Ускорение и параллельная эффективность алгоритма HAlEM

Очевидно, что точка **0** является вершиной допустимого многогранника системы ограничений (27). Для произвольного n эта задача может быть получена в формате MatrixMarket [25] с помощью генератора FRaGenLP [29], если в качестве количества случайных неравенств задать 0. Исходные коды генератора FRaGenLP на языке C++ свободно доступны по адресу `https://gitverse.ru/sokolinsky/FRaGenLP`. Сгенерированные задачи для различных n доступны по адресу `https://gitverse.ru/sokolinsky/Set-of-LP-Problems/content/master/Tcube` под именами `lp_tcube<s>K<h>`, где в качестве `<s>` и `<h>` указаны две цифры, задающие размерность задачи: $n =$ sh00. В экспериментах мы использовали три задачи: tcube0K2, tcube0K3 и tcube0K4, информация о которых приведена в табл 4. Семантика столбцов

Таблица 4. Тестирование HAlEM на задачах Tcube

Задача	m	n	$F_{\max}$	K_{peak}	$K_{\max}$	$T_{K_{\max}}$	T_1	α	ϵ
tcube0K2	401	200	4019900	16	16	16.1	136	8.5	0.53
tcube0K3	601	300	9029900	24	24	92.5	1126	12.2	0.51
tcube0K4	801	400	16039900	33	24	387	5681	14.7	0.61

такая же, как в табл. 2, за исключением столбца $F_{\max}$, содержащего максимальное значение целевой функции. Относительную погрешность указывать не стали, так как для всех трех задач она была равна нулю. Соответствующие графики ускорения и параллельной эффективности приведены на рис. 4. Отметим, что, как и в предыдущих экспериментах, на каждом процессорном узле запускалось 12 MPI-процессов. В качестве начальной вершины всегда выбиралась точка **0**. Эксперименты показали, что для задач размерностей 200 и 300 теоретическая граница масштабируемости K_{peak} совпала с реальной $K_{\max}$. Однако при увеличении размерности задачи до 400, начиная с 25 узлов, накладные расходы на распараллеливание стали превалировать над ускорением. В этом случае реальная граница масштабируемости оказалась заметно меньше теоретической. Отметим, что и в этой серии экспериментов параллельная эффективность на реальной границе масштабируемости не опускалась ниже 51%, что можно считать хорошим результатом.

Заключение

В данной работе представлен новый проекционный алгоритм HAlEM для решения задач линейного программирования, который объединяет сильные стороны разработанного ранее авторами алгоритма AlEM (проекционный подход) и классического симплекс-метода (обновление матричного базиса). Основная идея HAlEM заключается в построении пути к оптимальной вершине путем последовательного перехода по ребрам допустимого многогранника, причем на каждом шаге для определения направления движения используется эффективная процедура двух-факторной ортогональной проекции, а для переключения на новое ребро применяется механизм ротации базисных индексов, аналогичный симплекс-методу. Такой гибридный подход позволил преодолеть главные недостатки предшествующих алгоритмов: избежать экспоненциального комбинаторного перебора (в отличие от AlFaMove и AlEM) и проблем, связанных с низкой скоростью сходимости фейеровских процессов при вычислении проекций (в отличие от апекс-метода).

Для предложенного алгоритма разработана параллельная версия, ориентированная на кластерные вычислительные системы. Распараллеливание, основанное на BSF-модели и схеме «мастер-рабочие», заключается в одновременной обработке всех ребер текущей вершины, что позволяет значительно сократить время выполнения итерации.

Проведенные вычислительные эксперименты на задачах из репозитория Netlib-LP и на серии специально сконструированных задач подтвердили работоспособность и эффективность алгоритма HAlEM. Основные выводы по результатам исследования можно сформулировать следующим образом. Благодаря использованию метода двух-факторной проекции, алгоритм HAlEM демонстрирует высокую точность вычисления оптимального значения целевой функции, находящуюся в пределах машинного нуля для типа double (64 бита). HAlEM имеет ресурс параллелизма, существенно превосходящий симплекс-метод, что позволяет эффективно задействовать значительно большее количество процессорных узлов. Хотя по абсолютному времени счета на малом числе узлов HAlEM уступает симплекс-методу (разница в пределах одного порядка), на некоторых задачах (например, на циклических многогранниках) HAlEM может находить решение за меньшее число итераций.

Исследование масштабируемости показало, что граница эффективного распараллеливания алгоритма HAlEM напрямую зависит от размерности допустимого многогранника. Для задач с небольшой и средней размерностью теоретическая граница масштабируемости совпадает с реальной. При увеличении размерности накладные расходы на коммуникацию могут ограничивать рост ускорения, однако параллельная эффективность на реальной границе масштабируемости остается выше 50% во всех случаях, что является хорошим показателем для численного алгоритма.

Дальнейшие исследования планируется направить на разработку искусственной нейронной сети, которая позволит идентифицировать ребра допустимого многогранника быстрее, чем это делает алгоритм HAlEM с помощью двух-факторной проекции. Подобная модернизация позволит алгоритму HAlEM уверенно обогнать симплекс-метод по быстродействию на реальных задачах.

Список литературы

[1] Pan, P.-Q.: Linear Programming Computation. Springer, Berlin, Heidelberg (2014).
[2] Bixby R.E. Solving Real-World Linear Programs: A Decade and More of Progress // Operations Research. 2002. Vol. 50, no. 1. P. 315. DOI: 10.1287/opre.50.1.3.17780.

[3] Rasmussen, S.: Production Economics: The Basic Theory of Production Optimisation. 2nd edn. Springer, Berlin, Heidelberg (2013).
[4] Xu, Y.: Solving Large Scale Optimization Problems in the Transportation Industry and Beyond Through Column Generation. In: Fathi, M., Khakifirooz, M., Pardalos, P.M. (eds.) Optimization in Large Scale Problems: Industry 4.0 and Society 5.0 Applications, pp. 269–292. Springer, Cham (2019).
[5] Chung, W.: Applying large-scale linear programming in business analytics. In: 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 1860–1864. IEEE (2015).
[6] Gondzio, J., Gruca, J.A., Hall, J.A.J., Laskowski, W., Zukowski, M.: Solving large-scale optimization problems related to Bell's Theorem. Journal of Computational and Applied Mathematics 263, 392–404 (2014).
[7] Dantzig, G.B., Thapa, M.N.: Linear Programming 2: Theory and Extensions. Springer, New York (2003).
[8] Tolla, P.: A Survey of Some Linear Programming Methods. In: Paschos, V.T. (ed.) Concepts of Combinatorial Optimization, 2nd edn., pp. 157–188. John Wiley and Sons, Hoboken (2014).
[9] Mamalis, B., Pantziou, G.: Advances in the Parallelization of the Simplex Method. In: Zaroliagis, C., Pantziou, G., Kontogiannis, S. (eds.) Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295, pp. 281–307. Springer, Cham (2015).
[10] Im, H., Wolkowicz, H.: Revisiting degeneracy, strict feasibility, stability, in linear programming. European Journal of Operational Research 310(2), 495–510 (2023).
[11] Gass, S.I., Vinjamuri, S.: Cycling in linear programming problems. Computers and Operations Research 31(2), 303–311 (2004).
[12] Hall, J.A.J., McKinnon, K.I.M.: The simplest examples where the simplex method cycles and conditions where EXPAND fails to prevent cycling. Mathematical Programming, Series B 100(1), 133–150 (2004).
[13] Соколинский, Л.Б., Соколинская, И.М.: О новой версии апекс-метода для решения задач линейного программирования. Вестник ЮУрГУ. Серия: Вычислительная математика и информатика 12(2), 5–46 (2023).
[14] Olkhovsky, N.A., Sokolinsky, L.B.: О новом методе линейного программирования. Вычислительные методы и программирование 24(4), 408–429 (2023).
[15] Соколинский, Л.Б., Ольховский, Н.А., Соколинская, И.М.: Численная реализация метода поверхностного движения для решения задач линейного программирования. Вестник ЮУрГУ. Серия: Вычислительная математика и информатика 13(3), 5–31 (2024).
[16] Жулев, А.Э., Соколинский, Л.Б.: О поиске оптимальной вершины на многограннике допустимых решений задачи линейного программирования. Вычислительные методы и программирование 27(1), 1–18 (2026).
[17] Murty, K.G.: Computational and Algorithmic Linear Algebra and n-Dimensional Geometry. World Scientific Publishing Company, Singapore (2014).
[18] Стренг, Г.: Линейная алгебра и ее применения. Мир, Москва (1980).
[19] 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 149, 193–206 (2021).
[20] Sokolinsky, L.B.: BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems. MethodsX 8, Article number 101437 (2021).
[21] Dolganina, N., Ivanova, E., Bilenko, R., Rekachinsky, A.: HPC Resources of South

Ural State University. In: Sokolinsky, L., Zymbler, M. (eds.) Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618, pp. 43–55. Springer, Cham (2022).
[22] Gay, D.M.: Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Bulletin 13, 10–12 (1985).
[23] Схрейвер, А.: Теория линейного и целочисленного программирования: в 2 т. Т. 1. Мир, Москва (1991).
[24] Муртаф, Б.: Современное линейное программирование. Мир, Москва (1984).
[25] Boisvert, R.F., Pozo, R., Remington, K.A.: The Matrix Market Exchange Formats: Initial Design. NISTIR 5935. National Institute of Standards and Technology, Gaithersburg (1996).
[26] Koch, T.: The final NETLIB-LP results. Operations Research Letters 32(2), 138–142 (2004).
[27] Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics. 2nd edn. Texts in Applied Mathematics, vol. 37. Springer, Berlin (2007).
[28] Циглер, Г.М.: Теория многогранников. МЦНМО, Москва (2014).
[29] Соколинский, Л.Б., Соколинская, И.М.: О генерации случайных задач линейного программирования на кластерных вычислительных системах. Вестник ЮУрГУ. Серия: Вычислительная математика и информатика 10(2), 38–52 (2021).

Приложение. Функции, используемые в алгоритме HAlEM

Функция BasicIndex

Функция BasicIndex используется алгоритмом 1 на шаге 4 для построения базисного индекса начальной вершины. Псевдокод этой функции представлен в виде алгоритма 4.

Алгоритм 4 Построение базисного индекса для начальной вершины

```
1:  function BasicIndex(v)
2:      V := ∅
3:      for j = k + 1, ..., n do                    // k — число уравнений в системе (1)
4:          for all i ∈ Î do
5:              if ⟨a_i, v⟩ = b_i then
6:                  if rank(A_{Ī∪V∪{i}}) = j then
7:                      V := V ∪ {i}
8:                      break
9:                  end if
10:             end if
11:         end for
12:     end for
13:     return V
14: end function
```

В силу предположения (3) система ограничений (1) не содержит избыточных уравнений [5]. Поскольку любая точка допустимого многогранника M должна удовлетворять

[5] Если система ограничений содержит избыточные уравнения, их можно элиминировать, включая в первоначально пустую матрицу A строки коэффициентов уравнений, увеличивающих ее ранг,

всем уравнениям из системы (1), любой матричный базис в любой вершине $\boldsymbol{v}$ допустимого многогранника M должен включать в себя коэффициенты всех уравнений. Для того, чтобы получился полный матричный базис в вершине $\boldsymbol{v}$, к матрице $A_{\bar{I}}$ необходимо добавить $n-k$ строк коэффициентов неравенств, соответствующих ограничивающим гиперплоскостям, проходящим через $\boldsymbol{v}$, так, чтобы ранг получившейся квадратной матрицы был равен размерности пространства n. Алгоритм 4 вычисляет базисный индекс, содержащий индексы соответствующих неравенств. На шаге 2 создается пустой базисный индекс $\mathcal{V}$. Внешний цикл **for** выполняет $n-k$ итераций, добавляя каждый раз в $\mathcal{V}$ новый индекс i из $\hat{I}$ так, чтобы ранг матрицы $A_{\bar{I}\cup\mathcal{V}\cup\{i\}}$ оставался полным. Это обеспечивается путем выполнения вложенного цикла **for all** (шаги 4–9), который для каждого i-того неравенства на шаге 5 проверяет условие $\langle \boldsymbol{a}_i, \boldsymbol{v}\rangle = b_i$. Если это условие выполняется, то граничная гиперплоскость, соответствующая i-тому неравенству, проходит через вершину $\boldsymbol{v}$. В этом случае на шаге 6 проверяется условие $\operatorname{rank}(A_{\bar{I}\cup\mathcal{V}\cup\{i\}}) = j$. Выполнение этого условия означает, что матрица $A_{\bar{I}\cup\mathcal{V}\cup\{i\}}$ имеет полностроочный ранг. В этом случае шаг 7 добавляет индекс i в $\mathcal{V}$, после чего оператор **break** на шаге 8 выполняет досрочный выход из вложенного цикла **for all**. Процесс продолжается, пока не будут выполнены все итерации внешнего цикла **for**. В итоге получается полный базисный индекс $\mathcal{V}$, который на шаге 13 возвращается в качестве результата функции BasicIndex.

Функция DirectionVector

Функция DirectionVector используется алгоритмом 1 на шаге 11 для вычисления направляющего вектора реберной прямой с реберным индексом $\mathcal{E}$. Псевдокод этой функции представлен в виде алгоритма 5.

Алгоритм 5 Вычисление направляющего вектора

1: **function** DirectionVector($\boldsymbol{v}, \mathcal{E}$)
2: $\quad \boldsymbol{e_c} := \boldsymbol{c} / \|\boldsymbol{c}\|$
3: $\quad \boldsymbol{z} := \boldsymbol{v} + \delta \boldsymbol{e_c}$
4: $\quad \boldsymbol{w} := \boldsymbol{\pi}_{\bar{I}\cup\mathcal{E}}(\boldsymbol{z})$
5: $\quad$ **if** $\|\boldsymbol{w} - \boldsymbol{v}\| = 0$ **then**
6: $\quad\quad$ **return 0**
7: $\quad$ **end if**
8: $\quad \boldsymbol{d} := \boldsymbol{w} - \boldsymbol{v}$
9: $\quad \boldsymbol{e_d} := \boldsymbol{d} / \|\boldsymbol{d}\|$
10: $\quad \boldsymbol{z}' := \boldsymbol{v} + \delta \boldsymbol{e_d}$
11: $\quad \boldsymbol{w}' := \boldsymbol{\pi}_{\bar{I}\cup\mathcal{E}}(\boldsymbol{z}')$
12: $\quad \boldsymbol{d}' := \boldsymbol{w}' - \boldsymbol{v}$
13: $\quad$ **return** $\boldsymbol{d}'$
14: **end function**

Реализация этого алгоритма основывается на предложенном нами вычислительном методе, названным двух-факторной проекцией. Суть этого метода заключается в следующем. В контексте задачи ЛП (1) с помощью реберного индекса $\mathcal{E}$ задана реберная прямая $L_{\mathcal{E}}$, проходящая через вершину $\boldsymbol{v} \in M$. Необходимо найти направляющий вектор этой прямой, указывающий в сторону увеличения значения линейной целевой

и игнорируя строки коэффициентов уравнений, при добавлении которых ранг матрицы A остается неизменным.

функции с градиентом $\boldsymbol{c}$. Для этого на шаге 2 алгоритма 4 вычисляется единичный вектор $\boldsymbol{e_c} = \boldsymbol{c}/\|\boldsymbol{c}\|$, сонаправленный с вектором $\boldsymbol{c}$. Шаг 3 находит точку $\boldsymbol{z} = \boldsymbol{v} + \delta\boldsymbol{e_c}$. Здесь δ — положительная вещественная константа, являющаяся параметром алгоритма. Шаг 4 с помощью формулы (11) вычисляет точку $\boldsymbol{w} = \boldsymbol{\pi}_{\bar{I}\cup\mathcal{E}}(\boldsymbol{z})$, являющуюся ортогональной проекцией точки $\boldsymbol{z}$ на прямую $L_{\mathcal{E}}$ (см. рис. 5). Если на шаге 5 выясня-

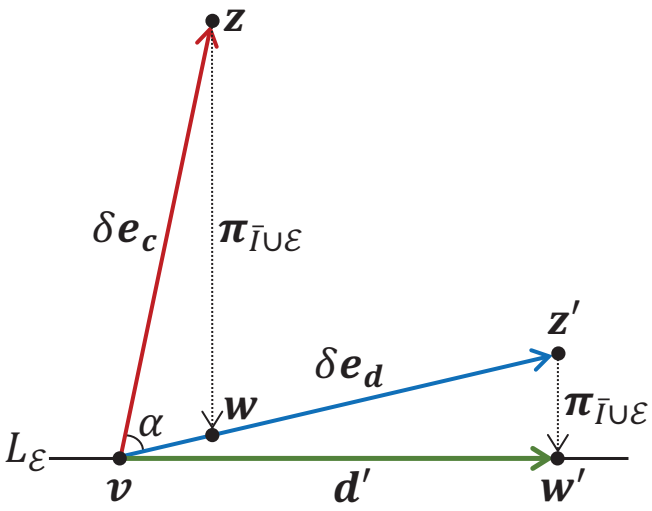


Рис. 5. Двух-факторная проекция

ется, что расстояние между точками $\boldsymbol{w}$ и $\boldsymbol{v}$ равно нулю, это означает, что градиент $\boldsymbol{c}$ целевой функции перпендикулярен прямой $L_{\mathcal{E}}$, и она не может вести к увеличению целевой функции. В этом случае на шаге 6 функция DirectionVector в качестве результата возвращает нулевой вектор. В противном случае на шаге 8 получаем ненулевой вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}$, направленный в сторону увеличения целевой функции. Точность вычисления координат точки $\boldsymbol{w}$ критически зависит от двух факторов: величины δ и угла α между векторами $\boldsymbol{c}$ и $\boldsymbol{d}$. Вычислительные эксперименты показали, что для достижения высокой точности на реальных задачах ЛП значение параметра δ должно находится в следующих пределах: $10^5 \leqslant \delta \leqslant 10^9$. Но этого оказывается недостаточно. Если величина угла α, оказывается близкой к $90°$, то наблюдается катастрофическая потеря точности для любых значений параметра δ. На рис. 5 такая потеря точности выражается в том, что точка $\boldsymbol{w}$ оказывается на некотором расстоянии от прямой $L_{\mathcal{E}}$. Для нейтрализации этого эффекта выполняется вторая фаза проектирования. На шаге 9 вычисляется единичный вектор $\boldsymbol{e_d} = \boldsymbol{d}/\|\boldsymbol{d}\|$, сонаправленный с вектором $\boldsymbol{d}$. Шаг 10 находит точку $\boldsymbol{z}' = \boldsymbol{v} + \delta\boldsymbol{e_c}$. Шаг 10 с помощью формулы (11) вычисляет точку $\boldsymbol{w}' = \boldsymbol{\pi}_{\bar{I}\cup\mathcal{E}}(\boldsymbol{z}')$, являющуюся ортогональной проекцией точки $\boldsymbol{z}'$ на прямую $L_{\mathcal{E}}$. Шаг 12 с высокой точностью вычисляет направляющий вектор $\boldsymbol{d}' = \boldsymbol{w}' - \boldsymbol{v}$ реберной прямой $L_{\mathcal{E}}$, указывающий в сторону увеличения целевой функции. Шаг 13 возвращает вектор $\boldsymbol{d}'$ в качестве результата функции DirectionVector.

Функция BoundigHyperplaneIndex

Функция BoundigHyperplaneIndex используется алгоритмом 1 на шаге 15 для вычисления индекс j гиперплоскости H_j, ограничивающей реберную прямую L_i. Псевдокод этой функции представлен в виде алгоритма 6.

Алгоритм 6 Вычисление индекса ограничивающей гиперплоскости

```
1: function BoundigHyperplaneIndex(v, d)
2:     β_min := +∞
3:     for all j ∈ Î do
4:         if ⟨a_j, d⟩ > 0 then
5:             β := (b_j − ⟨a_j, v⟩) / ⟨a_j, d⟩
6:             if β < β_min then
7:                 β_min := β
8:                 j* := j
9:             end if
10:        end if
11:    end for
12:    return j*
13: end function
```

Здесь $\boldsymbol{v}$ — вершина, через которую проходит реберная прямая L_i, $\boldsymbol{d}$ — ее направляющий вектор. Алгоритм 6 в точности реализует вычисления по формуле (22) и не нуждается в пояснении. Заметим только, что ограничивающая гиперплоскость всегда существует в силу ограниченности допустимого многогранника M.

Функция ScrollBasicIndex

В некоторых случаях алгоритм HAlEM, находясь в вершине $\boldsymbol{v}_t$, может попасть на шаге 29 в «тупиковую» ситуацию, когда в базисном индексе $\mathcal{V}_t$ отсутствуют ребра, ведущие к увеличению целевой функции. С помощью комбинаторного перебора всевозможных сочетаний ограничивающих гиперплоскостей, проходящих через $\boldsymbol{v}_t$, всегда можно найти ребро, ведущее к увеличению целевой функции (если такое ребро не существует, мы уже находимся в оптимальной вершине). Этот метод был нами реализован в алгоритме AlEM [16]. Однако алгоритм AlEM имеет неприемлемо высокую экспоненциальную вычислительную сложность для многих практических задач ЛП. Вместо этого в алгоритме HAlEM на шаге 29 мы используем функцию ScrollBasicIndex, которая «прокручивает» в вершине $\boldsymbol{v}_t$ различные варианты базисного индекса $\mathcal{V}_t$ до тех пор, пока не будет найдена комбинация, в которой есть ребро, ведущее к увеличению целевой функции. Псевдокод этой функции представлен в виде алгоритма 7, который для этой цели использует подход, применяемый в симплекс-методе и детально описанный в разделе 11.1 монографии [23].

При написании алгоритма 7 мы сохранили обозначения, использованные в [23], за исключением следующих: вершине x_0 соответствует $\boldsymbol{v}_t$; матрице A_0 соответствует матрица $A_{\bar{I}\cup\mathcal{V}}$; столбцу b_0 соответствует $\boldsymbol{b}_{\bar{I}\cup\mathcal{V}}$. Для своей работы алгоритм 7 использует три вспомогательных вектора: $\boldsymbol{g}, \boldsymbol{y} \in \mathbb{R}^n$ и $\boldsymbol{u} \in \mathbb{R}^m$. Основной цикл **repeat**/**until** (шаги 2–32) осуществляет «прокрутку» базисных индексов, пока не будет найден вариант с ребром, ведущим к увеличению целевой функции. Шаг 3 вычисляет вспомогательный вектор $\boldsymbol{g}$. Шаги 4–7 заполняют координаты вектора $\boldsymbol{u}$ из двойственной задачи. В отличии от [23] мы заполняем нулями координаты $\boldsymbol{u}$, соответствующие уравнениям. На шагах 8–14 находим индекс i^*, соответствующий первой положительной координате вектора $\boldsymbol{u}$. Если в $\boldsymbol{u}$ таких координат нет, то мы находимся в оптимальной вершине. В этом случае функция ScrollBasicIndex на шаге 16 возвращает исходный базисный индекс. В противном случае на шагах 18–20 вычисляются координаты направляющего вектора $\boldsymbol{y}$ для i^*-го ребра. Шаги 21–30 вычисляют индекс j^* ограничивающей гиперплоскости подобно тому, как это делалось в алгоритме 6.

Алгоритм 7 Прокручивание базисного индекса для выхода из тупика

Require: $\boldsymbol{g} = (g_1, \ldots, g_n)$; $\boldsymbol{y} = (y_1, \ldots, y_n)$; $\boldsymbol{u} = (u_1, \ldots, u_m)$

1: **function** ScrollBasicIndex($\boldsymbol{v}, \mathcal{V}$)
2: **repeat**
3: $\boldsymbol{g} := \boldsymbol{c}A^{-1}_{\bar{I}\cup\mathcal{V}}$
4: $\boldsymbol{u} := \boldsymbol{0}$
5: **for** $i = 1, \ldots, n-k$ **do**
6: $u_{\nu_i} := g_{k+i}$
7: **end for**
8: $i^* := 0$
9: **for** $i = 1, \ldots, m$ **do**
10: **if** $u_i < 0$ **then**
11: $i^* := i$
12: **break**
13: **end if**
14: **end for**
15: **if** $i^* = 0$ **then**
16: **return** $\mathcal{V}$
17: **end if**
18: **for** $j = 1, \ldots, n$ **do**
19: $y_j := -A^{-1}_{\bar{I}\cup\mathcal{V}}[j, k + i^*]$
20: **end for**
21: $\lambda_{min} := 0$
22: **for all** $j \in \hat{I}$ **do**
23: **if** $\langle \boldsymbol{a}_j, \boldsymbol{y}\rangle > 0$ **then**
24: $\lambda := \frac{b_j - \langle \boldsymbol{a}_j, \boldsymbol{v}\rangle}{\langle \boldsymbol{a}_j, \boldsymbol{y}\rangle}$
25: **if** $\lambda < \lambda_{min}$ **then**
26: $\lambda_{min} := \lambda$
27: $j^* := j$
28: **end if**
29: **end if**
30: **end for**
31: $\mathcal{V} := (\mathcal{V} \setminus \{i^*\}) \cup \{j^*\}$
32: **until** $\lambda_{min} > 0$
33: **return** $\mathcal{V}$
34: **end function**

Шаг 31 обновляет («прокручивает») базисный индекс $\mathcal{V}$, удаляя из него i^* и добавляя j^*. Процесс продолжается до тех пор, пока не будет получен вариант с $\lambda_{min} > 0$, означающий, что j^*-тое ребро ведет к увеличению целевой функции.

Следует отметить, что в редких случаях могут встретиться вырожденные задачи ЛП, для которых через определенное число итераций цикла **repeat/until** мы можем получить базисный индекс, который уже встречался ранее. В этом случае произойдет зацикливание алгоритма 7. Это характерно и для оригинального симплекс-метода, разработанного Данцигом. Для таких случаев предложены различные техники [11], позволяющие выйти из зацикливания. Однако обсуждение этих вопросов выходит за рамки настоящей статьи. Для невырожденных задач ЛП алгоритм HAlEM за конечное число шагов гарантированно сходится к оптимальной вершине.

Real-Time Detection of Memory Consumption Anomalies on the cHARISMa HPC Cluster

M. Salikova[1], P. Kostenetskiy[1]

[1]HSE University

Memory anomalies such as leaks pose a serious threat to the stability of large-scale high-performance computing infrastructure. This paper describes the design of a real-time memory anomaly detection module integrated into the monitoring ecosystem of the cHARISMa HPC cluster at HSE University. The system uses a cascaded pipeline that combines statistical feature engineering with a supervised Random Forest classifier to identify anomalous memory behavior at the per-job level.

Keywords: HPC monitoring, memory leak detection, time-series analysis, anomaly detection, hybrid models.

1. Introduction

High-performance computing clusters (HPC) are built to run computationally intensive tasks across distributed infrastructures. They are broadly applied in scientific simulations, machine learning, engineering, and large-scale data analysis. Reliable operation of HPC systems depends heavily on predictable resource utilization. Memory anomalies can noticeably degrade performance. In a large cluster like cHARISMa at HSE University [1], where many jobs run concurrently, even moderate inefficiencies tend to compound over time, causing poor scheduling decisions and elevated job failure rates. What makes this particularly challenging is that such anomalies often remain invisible for a long time. Memory consumption in university workloads is rarely static. Jobs frequently exhibit complex dynamics driven by a variety of factors. Among the problematic behaviors, memory leaks stand out as both the most operationally dangerous and the hardest to catch reliably. A leak manifests as a gradual rise in memory usage over time, which left unaddressed can exhaust available resources and destabilize the node or the cluster as a whole. This work addresses the problem of detecting such anomalies in real time using a combination of statistical analysis and machine learning applied to time-series telemetry. It is carried out as part of a bachelor research project.

2. Related works

Manual monitoring of memory usage across an HPC cluster is simply not feasible: the scale and variability of workloads make it impossible for administrators to review per-job time-series behavior in real time, especially when anomalies are subtle and context-dependent. Automated detection is therefore a necessity rather than a convenience. The earliest approaches to resource anomaly detection in HPC were based on statistical process control [2]. Practical implementations of cluster-wide behavioral analysis are successfully deployed at other prominent computing centers, for instance, the JobDigest and Octoshell platforms developed at Lomonosov Moscow State University (MSU) [5], which provide deep historical analysis of job efficiency and highlight abnormal resource consumption. In the machine learning literature, Random Forest and related supervised methods have demonstrated good results in anomaly classification tasks on operational data [3].

3. The cHARISMa HPC Cluster

HSE University operates the cHARISMa high-performance computing cluster. The system comprises 51 compute nodes equipped with modern GPU accelerators: NVIDIA H200 141GB, H100 80GB, A100 80GB, and V100 32GB, alongside high-performance CPUs. The heterogeneous node configuration is designed to support a wide range of research workloads. To date, the cluster has executed over 3,500,000 jobs and contributed to more than 250 studies in mathematics, physics, and related fields, with results published in Q1 Scopus and CORE A venues.

3.1. Position of the Proposed Module

Cluster operations are managed through HPC Taskmaster [4], a Task Efficiency Monitoring System developed by the HSE Supercomputer Modeling Unit. It provides centralized control over SLURM job management, databases, visualization, and reporting. The anomaly detection module described here augments this infrastructure with an analytical layer capable of intelligent job-level memory analysis.

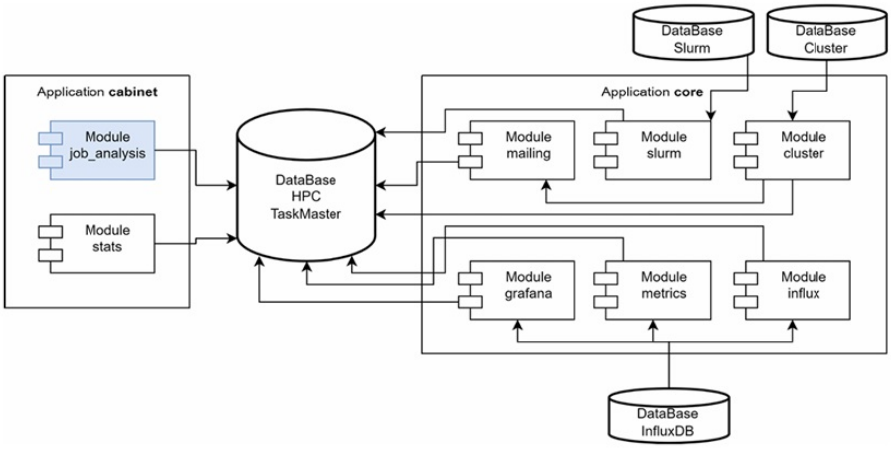


Fig. 1. Current layers of HPC Taskmaster

4. The Problem

Consider a job running on an HPC node. Its memory usage can be represented as a discrete time series:

$$X = \{x_1, x_2, \ldots, x_n\}$$

where x_i is the memory consumption at time t_i. The goal is to construct a cascaded decision pipeline ending in a function:

$$f \colon X \to \{0, 1\}$$

where 0 denotes a negatively classified normal execution (which includes stable plateaus and safe volatility), and 1 represents a positively classified anomalous memory behavior, specifically a leak. Unlike standard classification problems, this setting presents several complicating factors: ground-truth labels are largely unavailable, signals are non-stationary

and noisy, and inference must complete in real time. The approach therefore relies on a hybrid framework that evaluates multiple behavioral facets before making a final binary decision.

5. Methodology

A memory leak is not treated as a strict Boolean condition. Instead, it is interpreted as a *combination of tendencies*: a sustained upward trend over the analysis window, infrequent large negative drops, approximately linear growth, and sufficient total accumulation to warrant operational concern.

5.1. Hybrid detection pipeline

The pipeline combines signal processing, domain-specific heuristic rules, and machine learning in a sequential cascaded structure. Early rule-based stages filter out clearly normal behavior at low computational cost, and later stages apply the ML model only to the remaining suspicious candidates. Analysis is performed on a sliding window of 6 hours with 5-minute aggregation.

5.1.1. Feature engineering

Given memory usage V, normalization is applied to account for scale differences:

$$N_i = \frac{V_i - \min(V)}{\max(V) - \min(V)}$$

First-order differences are computed to encode local dynamics: $\Delta N_i = N_i - N_{i-1}$. Three key derived features are extracted: the coefficient of determination R^2 of a linear fit, the volatility $\sigma_{\Delta N}$, and the maximum drop $\min(\Delta N)$.

5.1.2. Heuristic filtering

Three domain-specific filters suppress false positives representing legitimate normal behaviors. First, jobs loading large datasets exhibit a strong startup trend forming a plateau (Fig 2). If the local regression on the series tail yields $\beta_{\text{tail}} \leq 0.0005$, the process is classified as a stable plateau and excluded from further analysis.

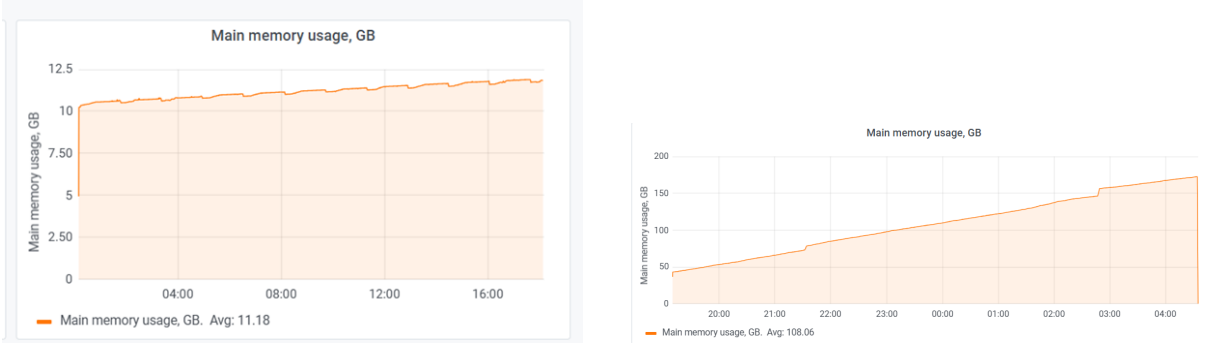


Fig. 2. Normal plateau behavior (left) and memory leak anomaly (right)

Second, to suppress insignificant anomalies, a two-part criticality threshold is applied: jobs with total growth below 300 MB or a growth rate below 100 MB/hour are ignored.

Third, a highly unstable, non-leaking memory management pattern is isolated. Jobs satisfying $\sigma_{\Delta N} > 0.04 \wedge \min(\Delta N) < -0.15$ are filtered out as simply volatile.

5.1.3. Machine Learning Layer

In the absence of labeled data, a weak supervision strategy is utilized. While the heuristic rules above systematically assign *negative* labels to normal jobs and filter them out, *positive* labels (leaks) for the training dataset are synthetically generated. This is done by isolating historical jobs that bypassed the initial normal filters, performed continuous growth thresholds and terminated with an error SLURM status. These feature vectors and synthetically assigned labels are manually reviewed and verified to correct mislabeled instances. A Random Forest classifier is then trained on this dataset using balanced class weights.

6. Statistical Modeling

6.1. Trend Estimation

To quantify the global trend of the memory series, a linear regression model is fitted over the window: $\tilde{x}_i = \beta_0 + \beta_1 i + \epsilon_i$. A classical memory leak caused by unreleased memory allocations within an iterative workflow (such as a program's main execution loop) typically manifests as a monotone accumulation over time. Therefore, assuming that loop iterations take roughly equal time, the resulting memory leak profile is approximately linear. A high coefficient of determination (R^2) effectively captures this constant-rate allocation pattern.

6.2. Derived Statistical Features

Other extracted scalar features include total growth G, maximum drop $D_{\min}$, volatility σ_d, fraction of negative steps p_{neg}, and the mean positive increment μ_+. These define the multi-dimensional feature space for the classification logic.

7. Machine Learning Approach

We define the binary classifier $f_\theta : \mathbb{R}^k \to \{0, 1\}$, where k is the number of extracted statistical features. The model is a Random Forest classifier:

$$f(x) = \frac{1}{T} \sum_{t=1}^{T} h_t(x)$$

Random Forest was selected because it naturally captures non-linear variable interactions, is highly robust to the label noise inherent to our weak supervision strategy, and offers transparent feature importance estimates critical for system administrators.

8. Final Decision

The ML model serves as the final stage for candidates passing the statistical filters. The composite anomaly decision is:

$$\text{Leak} \iff \text{Passed Heuristic Filters} \wedge P_{\text{ML}} > \theta$$

where P_{ML} is the ML confidence score. The system must operate under real-time constraints, avoiding full recomputations by maintaining running statistics updated in $O(1)$ per new observation. A full analysis cycle for all active jobs completes well within the 5-minute sampling interval.

9. Integration with HPC Monitoring Infrastructure

The module is integrated into the HPC Taskmaster telemetry framework. It queries active memory metrics from InfluxDB using time-bucket aggregation, processes each job independently (allowing horizontal scaling across active nodes), and records generated anomalies for immediate alert distribution and subsequent post-hoc inspection.

10. Results and Observations

To evaluate the proposed approach, a historical dataset consisting of 2,500 job time-series (up to 72 points each) was collected. The data was split chronologically to maintain practical validity: roughly 80% (2,000 jobs) for training and 20% (500 jobs) for the test set. Because HPC anomalies are rare, the test set contained a naturally imbalanced distribution with approximately 5.6% positive instances (28 verified leaks). To benchmark the module, we defined a standard *rule-based baseline*. This baseline purely relies on rigid static thresholds to identify anomalies, specifically flagging any job meeting both $G > 300\,\mathrm{MB}$ and $R^2 > 0.75$. During the offline evaluation on the test set, the pure rule-based baseline achieved a Recall of 69.1% but suffered from heavy false positives. This yielded a low Precision of 41% and an F1-score of 0.51. The proposed hybrid pipeline improved detection. The ML layer successfully eliminated many false positives. It achieved Precision of 47% (an absolute improvement of 6% over the baseline) and a Recall of 74%, yielding a combined F1-score of 0.58 alongside an overall Accuracy of 96.2% (due to the imbalance). While the precision indicates the continued presence of some false alerts, this represents a practical improvement in the context of severely imbalanced telemetry data.

Feature importance analysis confirmed a clear dominance of three features by the ML model: total growth G, which captures accumulation magnitude; the linear slope β_1, encoding the growth rate; and the coefficient of determination R^2, which measures growth consistency.

Conclusion

The paper has presented a practical approach to detecting memory anomalies in HPC systems, targeted for testing and deployment on the cHARISMa cluster. By combining statistical pre-filters with an ensemble machine learning model, the framework achieves an F1-score of 0.58, outperforming naive rule-based tracking. The immediate next step is a controlled shadow-mode deployment on the live cluster: the module will log anomalies without triggering automated intervention, allowing administrators to further build a ground-truth dataset in real operational conditions. Following successful shadow evaluation, a graduated rollout is planned in which verified anomalies will trigger automated notifications to job owners, ultimately serving as a foundation for a broader resource efficiency analytics platform.

Acknowledgement

[1] Kostenetskiy P.S., Chulkevich R.A., Kozyrev V.I. HPC Resources of the Higher School of Economics. Journal of Physics: Conference Series, 2021, Vol. 1740, Art. 012050. DOI: 10.1088/1742-6596/1740/1/012050.

[2] Nikitenko D. A., Voevodin V. V., Zhumatiy S. A. Deep Analysis of Job State Statistics on Lomonosov-2 Supercomputer. Supercomputing Frontiers and Innovations, 2018, Vol. 5, No. 2, pp. 4–10. DOI: 10.14529/jsfi180201.

[3] Lu Y., Zhang W., Yang L., et al. A Survey of Anomaly Detection in HPC Systems Using Machine Learning. CCF Transactions on High Performance Computing, 2026. DOI: 10.1007/s42514-025-00266-7.

[4] Kostenetskiy P., Kozyrev V., Chulkevich R., Raimova A. Enhancement of the Data Analysis Subsystem in the Task-Efficiency Monitoring System HPC TaskMaster for the cHARISMa Supercomputer Complex at HSE University. International Conference on Parallel Computational Technologies, 2025. DOI: 10.1007/978-3-031-73372-7_4.

[5] Nikitenko D., Voevodin V., Antonov A., et al. JobDigest – Detailed System Monitoring-Based Supercomputer Application Behavior Analysis. In: *RuSCDays 2017. Communications in Computer and Information Science*, Vol. 793. Springer, Cham, 2017. DOI: 10.1007/978-3-319-71255-0_42.

Анализ параллельных алгоритмов для симметризованных треугольных методов решения сеточных эллиптических уравнений с переменными коэффициентами*

А.И. Сухинов[1], В.Н. Литвинов[1], Д.А. Соломаха[1]

[1]Донской государственный технический университет, г. Ростов-на-Дону

Рассматривается параллельная реализация методов последовательной верхней релаксации (SOR) и его симметричной модификации (SSOR) для решения трёхмерных сеточных уравнений эллиптического типа с переменными коэффициентами. Предложена модель иерархической декомпозиции расчётной области единичного куба $\Omega = [0,1]^3$ вдоль оси аппликат, обеспечивающая математическую эквивалентность параллельного алгоритма последовательному эталону при конвейерной обработке с асинхронной синхронизацией теневых плоскостей и многопоточной обработке поперечных сечений средствами OpenMP. Теоретически обоснованы условия сходимости итерационных операторов в диапазоне параметра релаксации $\omega \in (0,2)$ и доказано свойство положительной определённости предобуславливателя M_{SSOR}. Экспериментальные исследования на кластерной системе с процессорами Intel Xeon продемонстрировали сокращение числа итераций методом SSOR на 45–55% по сравнению с SOR при сохранении устойчивости схемы. Достигнутое ускорение $S_{16} \approx 7.1$–7.6 при эффективности $E_{16} \approx 0.44$–0.47 подтверждает целесообразность применения разработанных алгоритмов в составе высокопроизводительных вычислительных комплексов для задач гидрофизики и цифровой физики сплошных сред.

Ключевые слова: сеточные уравнения, уравнение Пуассона, итерационные методы, метод верхней релаксации, параллельные алгоритмы, декомпозиция области, OpenMP, ускорение.

1. Введение

Численное моделирование многокомпонентных гидрофизических, геофизических и термомеханических процессов в сплошных средах традиционно опирается на решение систем дифференциальных уравнений в частных производных эллиптического и параболического типов. В частности, при математическом описании динамики прибрежных акваторий, фильтрации в пористых средах, теплопереносе в анизотропных материалах и коррекции давления в схемах расщепления уравнений Навье–Стокса возникает необходимость решения трёхмерного уравнения Пуассона или его обобщённой формы с пространственно-переменными коэффициентами. Дискретизация указанных уравнений на структурированных сетках высокого разрешения приводит к формированию систем линейных алгебраических уравнений (СЛАУ) порядка 10^7, матрицы которых обладают разреженной структурой, свойством строгого диагонального преобладания и положительной определённостью.

*Работа выполнена при финансовой поддержке Российского научного фонда (проект 22-11-00295-П)

Прямые методы решения обладают вычислительной сложностью $O(N^3)$ по памяти и $O(N^{2.5})$–$O(N^3)$ по времени для трёхмерных задач, что делает их неприменимыми для расчётов промышленного масштаба на современных суперкомпьютерных архитектурах. В связи с этим итерационные методы релаксации, в частности метод последовательной верхней релаксации (Successive Over-Relaxation, SOR) и его симметричная модификация (Symmetric SOR, SSOR), сохраняют высокую актуальность благодаря асимптотически линейной вычислительной сложности $O(N^3)$ и минимальным требованиям к оперативной памяти. Ключевым ограничением данных методов является лексикографический обход узлов, генерирующий строгую последовательную зависимость обновлений и препятствующий естественному распараллеливанию по данным.

К числу существенных преимуществ симметричной модификации метода последовательной верхней релаксации (SSOR) относится устранение направленной асимметрии, присущей классическому алгоритму SOR при лексикографическом обходе узлов. За счёт последовательного применения прямого и обратного полушагов итерационный оператор SSOR приобретает симметричную структуру, что математически выражается в свойстве положительной определённости соответствующего предобуславливателя $M_{\mathrm{SSOR}}(\omega)$ для любого параметра релаксации $\omega \in (0, 2)$. Данное свойство гарантирует вещественность и положительность спектра преобразованной матрицы системы, что позволяет использовать SSOR в качестве эффективного симметричного предобуславливателя в методах сопряжённых градиентов (PCG). В рамках данного подхода обеспечивается квадратичная сходимость и строгая монотонность убывания энергетической нормы ошибки, чего невозможно достичь при использовании несимметричных итерационных схем.

Кроме того, симметричный обход снижает чувствительность алгоритма к точному значению параметра ω, расширяя интервал устойчивой сходимости и повышая робастность метода при решении уравнений с сильно неоднородными пространственно-переменными коэффициентами. В контексте трёхмерных сеточных задач SSOR также демонстрирует улучшенные свойства сглаживания высокочастотных компонент погрешности, что обуславливает его широкое применение в качестве сглаживающего оператора в геометрических и алгебраических многосеточных алгоритмах. Указанные теоретические преимущества реализуются при умеренных вычислительных затратах, поскольку каждый полушаг выполняется через стандартное ядро SOR, что сохраняет асимптотическую сложность $O(N^3)$ и минимальные требования к дополнительной памяти, критически важные при моделировании задач высокого разрешения на распределённых вычислительных архитектурах.

Разработка эффективных параллельных алгоритмов SOR/SSOR, согласованных с иерархической архитектурой распределённых вычислительных сред, представляет собой фундаментальную задачу вычислительной математики. Обеспечение скрытия латентности межпроцессорных коммуникаций, балансировки нагрузки и векторизации вычислительных ядер при сохранении математической эквивалентности последовательному алгоритму остаётся приоритетным направлением при проектировании высокопроизводительных вычислительных комплексов для гидрофизики и цифровой физики сплошных сред. В настоящей работе предлагается модель декомпозиции расчётной области единичного куба $\Omega = [0, 1]^3$ вдоль оси аппликат (Oz), реализующая асинхронную синхронизацию теневых слоёв и многопоточную обработку поперечных сечений, что позволяет преодолеть структурные ограничения лексикографической зависимости и достичь высокой вычислительной эффективности на многоядерных системах.

2. Математическая постановка и итерационные схемы SOR/SSOR

Рассматривается краевая задача Дирихле для обобщённого уравнения Пуассона в единичном кубе $\Omega = [0,1]^3$:

$$-\nabla \cdot (a(\mathbf{x})\nabla u(\mathbf{x})) = \varphi(\mathbf{x}), \quad x \in \Omega, \quad u|_{\partial\Omega} = 0, \tag{1}$$

где коэффициент диффузии моделирует пространственно-неоднородную среду и задаётся гладкой функцией

$$a(\mathbf{x}) = 1 + c\|\mathbf{x} - \mathbf{x}_0\|^2, \quad c = 101.0, \quad \mathbf{x}_0 = (0.5, 0.5, 0.5)^\top.$$

Правая часть $\varphi(\mathbf{x})$ предполагается известной и принадлежащей $L_2(\Omega)$. На равномерной сетке

$$\Omega_h = \{(x_i, y_j, z_k) \mid x_i = ih, y_j = jh, z_k = kh; i, j, k = 0, \ldots, N-1\},$$

с шагом $h = (N-1)^{-1}$ вводится стандартный шеститочечный разностный шаблон. Дискретизация интегро-интерполяционным методом даёт для каждого внутреннего узла (i, j, k) разностное уравнение

$$\begin{aligned} -\frac{1}{h^2}\Big[&\kappa_{i+\frac{1}{2},j,k}(u_{i+1,j,k} - u_{i,j,k}) - \kappa_{i-\frac{1}{2},j,k}(u_{i,j,k} - u_{i-1,j,k}) \\ &+\kappa_{i,j+\frac{1}{2},k}(u_{i,j+1,k} - u_{i,j,k}) - \kappa_{i,j-\frac{1}{2},k}(u_{i,j,k} - u_{i,j-1,k}) \\ &+\kappa_{i,j,k+\frac{1}{2}}(u_{i,j,k+1} - u_{i,j,k}) - \kappa_{i,j,k-\frac{1}{2}}(u_{i,j,k} - u_{i,j,k-1})\Big] = \varphi_{i,j,k}, \end{aligned} \tag{2}$$

где $\kappa_{i\pm\frac{1}{2},j,k} = a(x_i \pm h/2, y_j, z_k)$ и аналогично для других направлений – значения коэффициента диффузии в полуцелых точках. Для единообразия записи уравнение приводят к виду

$$a^{(d)}_{i,j,k} u_{i,j,k} - \sum_{\nu} \kappa_\nu u_\nu = h^2 \varphi_{i,j,k}, \tag{3}$$

где индекс ν пробегает шесть соседей узла, κ_ν – соответствующий коэффициент на ребре, а диагональный элемент

$$a^{(d)}_{i,j,k} = \sum_{\nu} \kappa_\nu.$$

В матричной форме система принимает вид

$$A\mathbf{u} = \mathbf{f}, \qquad A \in \mathbb{R}^{N^3 \times N^3}, \tag{4}$$

с вектором неизвестных u, упорядоченных лексикографически, и правой частью $\mathbf{f} = h^2\boldsymbol{\varphi}$. Матрица A наследует свойства оператора задачи: она симметрична, положительно определена (СПО) и обладает M-свойством – все внедиагональные элементы неположительны, диагональные положительны, и имеет место строгое диагональное преобладание. Эти свойства гарантируют корректность применяемых итерационных методов.

Представим A в стандартном расщеплении

$$A = D - L - U, \tag{5}$$

где $D = \mathrm{diag}(A)$ – диагональная часть, L – строго нижняя треугольная, U – строго верхняя треугольная. Поскольку нумерация лексикографическая, L и U являются разреженными блочно-трёхдиагональными матрицами.

2.1. Метод последовательной верхней релаксации (SOR)

Итерационный процесс SOR определяется рекуррентным соотношением

$$(D - \omega L)\mathbf{u}^{(m+1)} = [(1-\omega)D + \omega U]\mathbf{u}^{(m)} + \omega\mathbf{f}, \tag{6}$$

где $\omega \in (0, 2)$ – релаксационный параметр, m – номер итерации. В координатной форме для узла (i, j, k) это даёт известную формулу

$$u_{i,j,k}^{(m+1)} = (1-\omega)u_{i,j,k}^{(m)} + \frac{\omega}{a_{i,j,k}^{(d)}}\left(h^2\varphi_{i,j,k} + \sum_{\nu}\kappa_\nu u_\nu^{\text{curr/prev}}\right). \tag{7}$$

Здесь u_ν^{curr} – значения соседей, уже обновлённые на текущей итерации (при лексикографическом порядке это соседи с меньшими индексами), а u_ν^{prev} – значения с предыдущей итерации (ещё не обновлённые). При $\omega = 1$ метод превращается в классический метод Гаусса–Зейделя.

Сходимость метода SOR гарантируется классическим результатом Островского–Рейча: если A – симметричная положительно определённая матрица, то метод SOR сходится при любом $\omega \in (0, 2)$. При этом спектральный радиус итерационной матрицы $\mathcal{L}_{\text{SOR}} = (D - \omega L)^{-1}[(1-\omega)D + \omega U]$ строго меньше единицы, а оптимальное значение ω_{opt}, минимизирующее спектральный радиус, определяется соотношением

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \mu_{\max}^2}},$$

где $\mu_{\max}$ – максимальное по модулю собственное значение матрицы $D^{-1}(L + U)$, соответствующей якобиеву итерационному оператору.

Для уравнения Пуассона с постоянным коэффициентом $a(x) \equiv 1$ известно, что $\mu_{\max} = \cos(\pi h)$. Тогда

$$\omega_{\text{SOR}}^* = \frac{2}{1 + \sin(\pi h)} \approx 2 - 2\pi h, \quad \rho_{\text{SOR}}^* = \omega_{\text{SOR}}^* - 1 \approx 1 - 2\pi h.$$

Число итераций $m \approx \ln\varepsilon / \ln\rho$. Отсюда $m_{\text{SOR}} = O(N|\ln\varepsilon|)$, $m_{\text{SSOR}} = O(N|\ln\varepsilon|)$ с вдвое большей константой.

Для рассматриваемой трёхмерной задачи с переменными коэффициентами точное значение ω_{opt} аналитически не выражается, поэтому на практике его подбирают экспериментально в диапазоне $[1.8, 1.95]$.

При гладком переменном коэффициенте $a(\mathbf{x})$ собственные значения матрицы Якоби $D^{-1}(L + U)$ локализованы вблизи значений, отвечающих локально замороженному коэффициенту, поэтому оптимальный параметр ω слабо меняется при изменении c. Вычислительные эксперименты подтверждают, что ω_{opt} для SOR остаётся в диапазоне $[1.88, 1.95]$ при $c \in [0, 200]$, а для SSOR – в диапазоне $[1.78, 1.90]$.

2.2. Симметричный метод последовательной верхней релаксации (SSOR)

Асимметрия прямого лексикографического обхода в SOR приводит к несимметричности итерационного оператора и зависимости от направления обхода. Метод SSOR устраняет этот недостаток, выполняя на каждой итерации два полушага:

1. **Прямой ход:** $k = 0, \ldots, N-1$, формула SOR.

2. **Обратный ход:** $k = N-1, \ldots, 0$, та же формула SOR, применённая к результату прямого хода.

В матричной записи два полушага записываются как

$$(D - \omega L)\mathbf{u}^{(m+1/2)} = [(1-\omega)D + \omega U]\mathbf{u}^{(m)} + \omega\mathbf{f}, \tag{8}$$

$$(D - \omega U)\mathbf{u}^{(m+1)} = [(1-\omega)D + \omega L]\mathbf{u}^{(m+1/2)} + \omega\mathbf{f}. \tag{9}$$

Исключая промежуточный вектор $\mathbf{u}^{(m+1/2)}$, получаем неявную схему

$$M_{\mathrm{SSOR}}(\omega)\mathbf{u}^{(m+1)} = (M_{\mathrm{SSOR}}(\omega) - A)\mathbf{u}^{(m)} + \mathbf{f}, \tag{10}$$

где предобуславливатель $M_{\mathrm{SSOR}}(\omega)$ задаётся выражением

$$M_{\mathrm{SSOR}}(\omega) = \frac{1}{\omega(2-\omega)}(D - \omega L)D^{-1}(D - \omega U). \tag{11}$$

Симметрия и положительная определённость M_{SSOR} при $\omega \in (0,2)$ устанавливается прямыми вычислениями: M_{SSOR} является произведением взаимно сопряжённых треугольных сомножителей, поэтому $(M_{\mathrm{SSOR}})^{\mathrm{T}} = M_{\mathrm{SSOR}}$, и для любого ненулевого $\mathbf{x}$

$$\mathbf{x}^{\mathrm{T}} M_{\mathrm{SSOR}}\mathbf{x} = \frac{1}{\omega(2-\omega)} \left\| D^{-1/2}(D - \omega U)\mathbf{x} \right\|_2^2 > 0.$$

Это свойство делает M_{SSOR} идеальным симметричным предобуславливателем для методов сопряжённых градиентов, обеспечивая монотонное убывание энергетической нормы ошибки.

Следовательно, для решения системы $A\mathbf{u} = \mathbf{f}$ можно применить метод сопряжённых градиентов с предобуславливателем M_{SSOR}, что гарантирует сходимость за число итераций, пропорциональное $\sqrt{\mathrm{cond}(M_{\mathrm{SSOR}}^{-1}A)}$, а не $\mathrm{cond}(A)$, как в случае простых итераций SSOR.

Итерационная матрица метода SSOR имеет вид

$$\mathcal{L}_{\mathrm{SSOR}} = I - M_{\mathrm{SSOR}}^{-1}A = (D - \omega U)^{-1}\left[(1-\omega)D + \omega L\right](D - \omega L)^{-1}\left[(1-\omega)D + \omega U\right],$$

откуда видна её симметричность относительно скалярного произведения, порождённого M_{SSOR}.

Спектральный анализ преобразованной системы $M_{\mathrm{SSOR}}^{-1}A$ показывает, что собственные значения вещественны и положительны, а их распределение существенно лучше обусловлено, чем у исходной матрицы A. Более того, при оптимальном выборе ω число обусловленности может быть значительно снижено, что уменьшает количество итераций.

2.3. Критерий остановки

Сходимость контролируется по относительной норме поправки:

$$\varepsilon^{(m)} = \frac{\|\mathbf{u}^{(m+1)} - \mathbf{u}^{(m)}\|_2}{\|\mathbf{u}^{(m+1)}\|_2 + \delta}, \quad \delta = 10^{-16}, \tag{12}$$

где δ введён для предотвращения деления на ноль. Итерационный процесс прекращается, как только $\varepsilon^{(m)} < \varepsilon_{\mathrm{tol}} = 10^{-6}$. Проверка сходимости выполняется не на каждом шаге, а периодически (через каждые 20 итераций), чтобы уменьшить накладные расходы на вычисление глобальной нормы при параллельной реализации.

3. Модель вычислительной системы и иерархическая декомпозиция расчётной области

Для организации эффективного распараллеливания на многоядерных системах с общей памятью вводится иерархическая модель декомпозиции расчётной области Ω_h, использующая стандарт OpenMP.

Уровни декомпозиции и отображение на ресурсы:

1. Логические разделы (1D, Oz): Глобальная сетка $N_x \times N_y \times N_z$ ($N_x = N_y = N_z = N$) разбивается по аппликате (k-индекс) на P разделов, каждый из которых закрепляется за отдельным потоком OpenMP. Размер локальной подобласти для потока t определяется как:
$$n_z^{(t)} = \left\lfloor \frac{N}{P} \right\rfloor + (t < N \bmod P), \quad t \in \{0, \ldots, P-1\}. \tag{13}$$
2. Уровень вычислительных ядер: Внутри каждого потока локальные плоскости (i, j) дополнительно распараллеливаются с помощью вложенных параллельных регионов или collapse-директив OpenMP.
3. Уровень потоковой обработки: Поперечные сечения обрабатываются параллельно с использованием директивы `#pragma omp parallel for collapse(2)`, что обеспечивает максимальную утилизацию SIMD-конвейеров и кеш-памяти.

Обновление узла (i, j, k) зависит от значений шести соседей. Для обеспечения корректности вычислений на границах разделов локальные массивы расширяются двумя граничными (теневыми) плоскостями:

$$\mathbf{u}^{\text{loc}} \in \mathbb{R}^{N \times N \times (n_z^{(t)}+2)}. \tag{14}$$

Поскольку метод SOR требует строгой последовательности вдоль Oz, параллелизм реализуется через конвейерную обработку фрагментов. Граф зависимостей представляет собой ориентированный ациклический граф, узлы которого соответствуют фрагментам сетки, а дуги задают смежность по координатам релаксации. Число шагов вычислительного процесса определяется как

$$N_r = N + P - 1, \tag{15}$$

а коэффициент полной загрузки вычислителей

$$\eta = \frac{N}{N + P - 1} = \left(1 + \frac{P-1}{N}\right)^{-1}. \tag{16}$$

Для скрытия задержек синхронизации применяется механизм флагов готовности с двойной буферизацией теневых плоскостей. Поток, обновивший граничный слой, устанавливает соответствующий флаг; соседний поток неблокирующим образом проверяет флаг, что позволяет совмещать вычисления внутренних слоёв с ожиданием готовности граничных данных. Это полностью аналогично неблокирующим точечным взаимодействиям и сохраняет низкие накладные расходы.

4. Алгоритмическая архитектура и организация параллельных вычислений

Программная реализация выполнена на языке C++17 с применением OpenMP для внутриузловой многопоточности. Архитектура решателя включает следующие ключевые компоненты:

1. Представление данных: Трёхмерные массивы хранятся в одномерном формате с макросом индексации $\mathrm{IDX3}(i, j, k, N) = iN^2 + jN + k$, что обеспечивает непрерывность доступа к памяти и эффективную векторизацию.
2. Основной итерационный цикл: Для $m = 0, 1, \ldots, m_{\max}$:
   - Прямой лексикографический обход локальных слоёв $k_{\mathrm{loc}} = 0 \ldots n_z^{(r)} - 1$ с применением формулы SOR;
   - Обратный обход $k_{\mathrm{loc}} = n_z^{(r)} - 1 \ldots 0$ (для SSOR);
   - Асинхронная синхронизация теневых плоскостей между соседними потоками (установка/проверка флагов готовности);
   - Локальное вычисление норм невязки и глобальная редукция посредством атомарного накопления в общей переменной с последующей барьерной синхронизацией;
   - Трансляция флага остановки через общую переменную с барьерной синхронизацией, гарантирующей одновременное завершение всех потоков.
3. Сборка результата: Каждый поток копирует свои внутренние узлы (без теневых слоёв) в итоговый массив, используя предвычисленные смещения, что обеспечивает корректную агрегацию данных без дополнительных пересылок.

Критически важной особенностью реализации является сохранение строгой последовательности обновлений вдоль оси Oz, что гарантирует математическую эквивалентность последовательному алгоритму и отсутствие нарушения монотонности схемы. Поперечные направления (x, y) распараллеливаются с применением статического планирования потоков, минимизирующего накладные расходы на синхронизацию.

5. Результаты вычислительных экспериментов и анализ масштабируемости

Вычислительные эксперименты проводились на вычислительной системе со следующими аппаратными характеристиками:

- Центральный процессор: Intel® Xeon® (архитектура x86_64), 16 физических ядер с поддержкой технологии Hyper-Threading (64 логических потока), базовая тактовая частота 2.4 ГГц;
- Оперативная память: 128 Гб DDR4, частота 2666 МГц, четырёхканальный режим доступа;
- Накопитель: твердотельный накопитель NVMe PCIe 3.0 x4 объёмом 1 Тб, обеспечивающий пропускную способность последовательного чтения/записи до 3500/3000 Мб/с;
- ОС Linux (ядро 5.15), компилятор GCC 11.3 с флагами `-O3 -march=native -ffast-math`, библиотека OpenMP 5.0.

Все запуски выполнялись в выделенном режиме с фиксацией процессов на физических ядрах посредством утилиты `numactl` для минимизации эффектов миграции потоков и конфликтов кеш-памяти. Для обеспечения воспроизводимости результатов каждый эксперимент повторялся трижды, в таблицах приведены усреднённые значения времени выполнения.

Экспериментальные исследования проведены на тестовой задаче с известным точным решением $u_{\mathrm{exact}} = x(x-1)y(y-1)z(z-1)$. Параметры расчётной области: единичный куб $\Omega = [0, 1]^3$, параметр неоднородности коэффициента диффузии $c = 101.0$. Точность остановки итерационного процесса: $\varepsilon_{\mathrm{tol}} = 10^{-6}$. Параметр релаксации ω подбирался эмпирически для каждого метода и размера сетки с целью

минимизации числа итераций при сохранении устойчивости схемы ($\omega_{\mathrm{SOR}} \in [1.90, 1.95]$, $\omega_{\mathrm{SSOR}} \in [1.80, 1.90]$).

Параллельная реализация основана на многопоточной модели (OpenMP). Декомпозиция расчётной области выполнена вдоль оси Oz, синхронизация теневых плоскостей реализована через флаги готовности. Поперечные сечения (x, y) распараллелены с помощью `collapse(2)`. Для каждого метода проведена серия запусков при варьировании числа потоков $P \in \{1, 2, 4, 8, 12, 16\}$ и размера сетки $N \in \{128, 256, 512\}$. В таблицах 1 и 2 приведены усреднённые значения времени выполнения T (в секундах).

Таблица 1. Результаты вычислений методом SOR: время выполнения T (с)

vCPU P	$N = 128$	$N = 256$	$N = 512$
1	13	302	4302
2	8	170	2015
4	4.1	112	1298
8	2.8	66	911
12	2.5	46	645
16	1.8	40	568

Таблица 2. Результаты вычислений методом SSOR: время выполнения T (с)

vCPU P	$N = 128$	$N = 256$	$N = 512$
1	19	471	11122
2	13	289	6328
4	11	235	3539
8	8	92	2149
12	7	83	1776
16	4	67	1572

Для количественной оценки эффективности параллельной реализации вычислялись коэффициенты ускорения $S_p = T_1/T_p$ и параллельной эффективности $E_p = S_p/P$, где T_1 — время выполнения на одном процессе, T_p — время при использовании P процессов. Дополнительно проводилась оценка точности: относительная L_2-норма погрешности не превышала $2.1 \cdot 10^{-6}$, что согласуется с порядком аппроксимации $O(h^2)$ и заданным порогом сходимости. Контроль сходимости выполнялся каждые 20 итераций с использованием атомарной редукции и барьера, обеспечивая синхронное завершение всех потоков. Результаты расчёта метрик масштабируемости представлены в таблицах 3–6.

Таблица 3. Ускорение S_p метода SOR для различных размеров сетки

vCPU P	$N = 128$	$N = 256$	$N = 512$
1	1.00	1.00	1.00
2	1.63	1.77	2.13
4	3.17	2.70	3.31
8	4.64	4.56	4.72
12	3.20	3.70	3.12
16	7.22	7.56	7.57

Таблица 4. Эффективность E_p метода SOR для различных размеров сетки

vCPU P	$N = 128$	$N = 256$	$N = 512$
1	1.00	1.00	1.00
2	0.81	0.89	1.07
4	0.79	0.67	0.83
8	0.58	0.57	0.59
12	0.43	0.55	0.56
16	0.45	0.47	0.47

Таблица 5. Ускорение S_P метода SSOR для различных размеров сетки

vCPU P	$N = 128$	$N = 256$	$N = 512$
1	1.00	1.00	1.00
2	1.46	1.63	1.77
4	1.72	2.00	3.17
8	2.38	5.12	5.22
12	2.71	5.67	6.32
16	4.75	7.03	7.14

Анализ масштабируемости выявляет характерные закономерности. Для метода SOR на сетке $N = 512$ при переходе от одного к двум потокам наблюдается суперлинейное ускорение ($S_2 \approx 2.13$, $E_2 > 1$), что объясняется более эффективным использованием кеш-памяти при уменьшении размера локальной подобласти. При

Таблица 6. Эффективность E_P метода SSOR для различных размеров сетки

vCPU P	$N = 128$	$N = 256$	$N = 512$
1	1.00	1.00	1.00
2	0.73	0.81	0.89
4	0.43	0.50	0.79
8	0.30	0.64	0.65
12	0.23	0.47	0.53
16	0.30	0.44	0.45

дальнейшем росте числа потоков эффективность монотонно снижается, достигая $E_{16} \approx 0.47$ для SOR и $E_{16} \approx 0.45$ для SSOR. Согласно закону Амдала, предельное ускорение ограничено долей последовательного участка: если доля последовательных операций (включая синхронизацию теневых плоскостей и редукцию) составляет $f \approx 0.05$, то максимальное ускорение $S_{\max} = 1/f \approx 20$, что согласуется с наблюдаемой тенденцией насыщения. Для SSOR эффективность растёт с увеличением размера сетки (от $E_{16} = 0.30$ при $N = 128$ до $E_{16} = 0.45$ при $N = 512$), что свидетельствует о снижении доли накладных расходов на синхронизацию относительно объёма вычислений. Достигнутые показатели подтверждают эффективность предложенной гибридной схемы распараллеливания.

На рис. 1 представлено сравнение полей абсолютной погрешности для методов SOR и SSOR на центральном сечении $z = 0.5$. Для метода SOR характерна пространственная асимметрия с концентрацией максимальной ошибки в области, смещённой относительно центра, что обусловлено однонаправленностью лексикографического обхода. В случае SSOR, благодаря симметризации оператора и двунаправленному распространению поправок, поле погрешности приобретает центрированную симметричную форму, что свидетельствует о более равномерном сглаживании невязки. На рис. 2 показана зависимость количества итераций, необходимых для достижения точности $\varepsilon_{tol} = 10^{-6}$, от числа потоков P для методов SOR и SSOR при $N = 128$.

6. Заключение

В работе представлены математическая постановка и параллельная реализация методов SOR и SSOR для решения трёхмерных сеточных уравнений эллиптического типа. Разработанная модель иерархической декомпозиции расчётной области вдоль оси аппликат, дополненная асинхронной синхронизацией теневых плоскостей и многопоточной обработкой поперечных сечений, обеспечивает математическую эквивалентность параллельного алгоритма последовательному эталону при эффективной утилизации многоядерных ресурсов. Теоретический анализ спектральных свойств итерационных операторов подтвердил условия сходимости в диапазоне $(0, 2)$ и обосновал применение SSOR в качестве симметрично положительно определённого предобуславливателя. Экспериментальные результаты демонстрируют существенное сокращение числа итераций методом SSOR, однако выявляют фундаментальные ограничения силь-

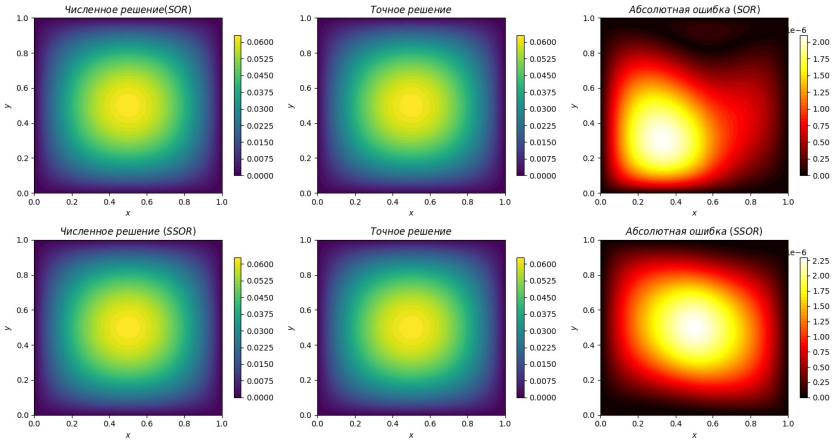


Рис. 1. Сравнение распределения абсолютной погрешности на сечении $z = 0.5$ при $N = 256$

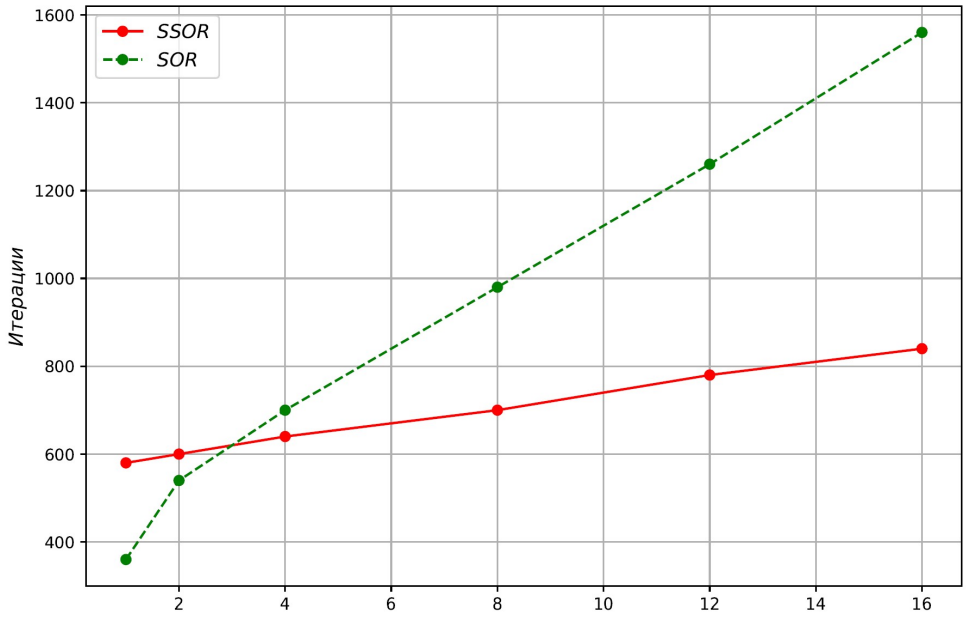


Рис. 2. Зависимость числа итераций от количества процессов P для методов SOR и SSOR при $N = 128$

ной масштабируемости, обусловленные лексикографической зависимостью данных и ростом накладных расходов на синхронизацию.

Перспективным направлением развития является адаптивная стратегия вычислений, основанная на пространственной неоднородности скорости сходимости SSOR. Поскольку при симметричном обходе узлы, расположенные вблизи границ, достигают заданной точности существенно раньше внутренних узлов, целесообразно применение техники динамического сужения активной вычислительной области. На каждом этапе итерационного процесса производится анализ локальной невязки, и стабилизировавшиеся узлы исключаются из дальнейших вычислений до тех пор, пока поправки в соседних узлах не превысят пороговое значение. Данный подход позволяет снизить общий объём арифметических операций, уменьшить интенсивность синхронизации для «замороженных» подобластей и повысить локальность доступа к кеш-памяти за счёт концентрации нагрузки в активной внутренней зоне.

Список литературы

[1] Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений. М.: Изд-во Наука, 1978. 592 с.

[2] Сухинов А.И., Чистяков А.Е., Никитина А.В. [и др.] Построение параллельных алгоритмов для моделирования гидродинамических процессов в Азовском море на основе гибридной технологии MPI+OpenMP // Вычислительная механика сплошных сред. 2023. Т. 16, № 1. С. 17–35. DOI: 10.7242/1999-6691/2023.16.1.2.

[3] Сидорякина В.В., Соломаха Д.А. Симметризованные варианты методов Зейделя и верхней релаксации решения двумерных разностных задач эллиптического типа // Computational Mathematics and Information Technologies. 2023. Т. 7, № 3. С. 12–19. DOI: 10.23947/2587-8999-2023-7-3-12-19.

[4] Сухинов А.И., Чистяков А.Е., Алексеенко Е.В. Численная реализация трехмерной модели гидродинамики для мелководных водоемов на супервычислительной системе // Математическое моделирование. 2011. Т. 23, № 3. С. 3–21. DOI: 10.1134/S2070048212050083.

Высокопроизводительные реализации матричного метода муравьиных колоний для параметрических задач на CPU с AVX2, GPU с CUDA и гибридных MPI+GPU кластерах

Ю.П. Титов[1]

[1]Московский Авиационный Институт (Национальный Исследовательский Университет), Москва, Россия

В работе представлены высокопроизводительные реализации матричной модификации метода муравьиных колоний для задач параметрической оптимизации. Исследованы три архитектурные модели параллелизма: SMP с векторизацией AVX2, SIMT на GPU и гибридная MPI+GPU. Эксперименты проведены на широком спектре оборудования: серверные ускорители NVIDIA Tesla V100, массовые GPU NVIDIA GTX 1050Ti и RTX 4060Ti, встраиваемая платформа NVIDIA Jetson Orin, а также CPU Intel Xeon Gold 6130, Core i5-8300H/9400F/10400F/12450H и AMD Ryzen 5 7500F / Ryzen 9 7900. Показана масштабируемость по данным: увеличение размерности задачи от 42 до 1,38 млн снижает нормированное время на Tesla V100 на 95%, а на Xeon Gold 6130 — на 82%. Среди CPU абсолютным лидером является AMD Ryzen 5 7500F с использованием AVX2, который обгоняет Intel Xeon Gold 6130 на 59%. Применение векторных инструкций AVX2 обеспечивает дополнительное ускорение до 85% относительно скалярной реализации. Гибридная MPI+GPU архитектура на 2–4 ускорителях показывает параллельную эффективность 91–99%. Проведено моделирование добавленной задержки, имитирующей работу суррогатных моделей и дорогостоящих целевых функций.

Ключевые слова: метод муравьиных колоний, параметрическая оптимизация, параллельные вычисления, CUDA, OpenMP, AVX2, MPI

1. Введение

В современной науке и технике все большее распространение получают задачи, в которых требуется найти оптимальный набор значений независимых параметров, определяющих качество функционирования сложной системы. Формально такая задача может быть определена следующим образом. Пусть имеется множество параметров $P = \{p_1, p_2, \ldots, p_n\}$. Каждый параметр p_i может принимать значения из конечного множества допустимых альтернатив $V_i = \{v_{1,i}, v_{2,i}, \ldots, v_{m_i,i}\}$, где $m_i = |V_i| > 0$. Решением является вектор $X_k = \{x_{1,k}, x_{2,k}, \ldots, x_{n_k,k}\}$, где $x_{i,k} \in V_i$. Требуется найти $X^* = \arg\min_{X_k} f(X_k)$, при этом целевая функция f задана в виде сложной аналитической или имитационной модели, вычисление которой требует значительных временных или вычислительных затрат [1–3]. В отличие от классической задачи коммивояжера (Travelling Salesman Problem – TSP), где решение представляет собой упорядоченную последовательность вершин, в параметрической задаче выбор значения каждого параметра $x_{i,k}$ не зависит от выбора других параметров. Это фундаментальное свойство независимости является ключевым для эффективной параллелизации.

Метод муравьиных колоний (Ant Colony Optimization, ACO), предложенный Colorni, Dorigo и Maniezzo в начале 1990-х годов [4], изначально разрабатывался для решения NP-трудных комбинаторных задач, таких как задача коммивояжера. Базовый алгоритм Ant System (AS) [5] использует стохастическое правило перехода, основанное на мультипликативной свертке феромонного следа τ_{ij} и эвристической привлекательности, основанной на обратной длине дуги $\eta_{ij} = 1/d_{ij}$. Дальнейшее развитие метода привело к созданию ряда модификаций, направленных на улучшение баланса между поиском множества решений, исследованием (exploration) и поиском решения около лучшей точки, эксплуатацией (exploitation) пространства поиска. Ant Colony System (ACS) [6] ввел правило псевдослучайного пропорционального выбора и локальное обновление феромона, что позволило повысить скорость сходимости. MAX-MIN Ant System (MMAS) [7] предложил механизм ограничения феромонных значений для предотвращения преждевременной стагнации. Elitist Ant System (EAS) и Rank-Based Ant System (RAS) усилили положительную обратную связь от лучших решений [8], а Best-Worst Ant System (BWAS) [9] ввел механизм «наказания» плохих решений путем вычитания феромона с ребер, входящих в наихудший маршрут.

В последние годы исследования в области ACO сосредоточены на адаптации метода к новым классам задач и повышению его эффективности. Shen и Gao [10] предложили гибридный ACO с адаптивным механизмом испарения феромона для динамических задач маршрутизации, показав улучшение на 18–25% по сравнению с классическими методами. Zhang et al. [11] разработали многоколоночный ACO с элитной миграцией для задачи балансировки сборочных линий. Liao et al. [12] провели обширное сравнение 15 модификаций ACO на 50 тестовых задачах TSP, показав, что гибриды с локальным поиском (ACO+2-opt, ACO+Lin-Kernighan) остаются наиболее эффективными для комбинаторной оптимизации.

Особый интерес представляет применение ACO к параметрическим задачам, особенно в контексте настройки гиперпараметров моделей машинного обучения. В работе [13] применялся ACO для оптимизации архитектуры сетей Long Short-Term Memory (LSTM-сетей), показав, что ACO находит конфигурации с точностью на 5–7% выше, чем случайный поиск, и на 3–4% выше, чем байесовская оптимизация. Hwang, Kang и Kim [14] предложили гибрид ACO и двунаправленной LSTM для распознавания эмоций, достигнув точности 94.2% на наборе данных RAVDESS. В работе [1] впервые предложили модификацию ACO с хэш-таблицей для кэширования результатов вычислений в параметрических задачах, показав сокращение числа вызовов целевой функции. Дальнейшее развитие [2, 3] привело к матричной формализации ACO, которая позволила использовать ускорители Single Instruction, Multiple Data (SIMD-ускорители) и достичь ускорения до 12 раз на центральных процессорах (CPU).

Параллелизация ACO является естественным направлением развития метода, поскольку популяция независимых муравьев-агентов предоставляет широкие возможности для распределения вычислений. Еще в 1998 году Bullheimer, Kotsis и Strauß [15] впервые систематически проанализировали стратегии распараллеливания Ant System, заложив основы для последующих исследований. Randall и Lewis [16] предложили master-slave модель для ACO, показав ускорение до 8 раз на 16 CPU. Zhou et al. [17] исследовали параллельный ACO на многоядерных SIMD CPU, продемонстрировав ускорение до 6.3 раза на процессоре Intel Xeon Phi. Abouelfarag, Aly и Elbaily [18] провели детальный анализ производительности OpenMP-реализации ACO, выявив узкие места в доступе к феромонной матрице и предложив техники локализации обновлений, повысившие эффективность с 45% до 85%. Mansour, Alaya и Tagina [19] разработали новый параллельный гибридный многокритериальный ACO на основе OpenMP

(PHMOACO), который превосходит классические MOACO по скорости сходимости на 30–40%. Тем не менее подобные исследования опирались на задачу коммивояжера и не рассматривали возможности работы метода при решении параметрических задач.

С появлением архитектуры CUDA для графических ускорителей (GPU) и видеокарт NVIDIA значительный импульс получили исследования GPU-реализаций ACO. Bai et al. [20] впервые реализовали MMAS на GPU, достигнув ускорения в 80 раз для TSP с 2000 городов. Skinderowicz [21] предложил эффективную реализацию ACS на GPU с использованием warp-специализации, показав ускорение 37.9 раза по сравнению с CPU-версией. В последующей работе [22] он развил этот подход для MMAS, предложив технику селективной памяти феромонов (SPM), которая улучшила качество решений при ускорении 16.85 раза. Huan et al. [23] предложили высокопроизводительную реализацию ACS на основе warp-специализации со статико-динамическим набором кандидатов. Zhi et al. [24] представили параллельный ACO на CUDA, в котором все операции выполняются одновременно без традиционных барьеров синхронизации, достигнув рекордного времени 0.8 секунды на итерацию с популяцией 10 000 муравьев. Cecilia et al. [25] разработали методологию высокопроизводительной реализации природо-инспирированных алгоритмов на GPU, предложив три ключевых принципа: максимальная параллелизация, минимизация конфликтов памяти и оптимальное использование иерархии памяти.

Для распределенных вычислительных систем, включая кластеры и суперкомпьютеры, наиболее адекватной моделью является Message Passing Interface (MPI). Pedemonte, Nesmachnow и Cancela [26] представили всестороннюю классификацию подходов к параллельному ACO, выделив мелкозернистые, крупнозернистые (островные) и гибридные модели. Tsutsui [27] реализовал ACO на нескольких GPU с CUDA для задачи квадратичного назначения (QAP), достигнув почти линейной масштабируемости с эффективностью 92% на 4 GPU. Stojanov и Fabian [28] сравнили четыре протокола миграции в MPI-реализации ACO, показав, что асинхронная миграция по запросу сокращает общее время решения на 23% по сравнению с синхронной миграцией. Verstraete [29] в обзоре метаэвристик для гетерогенных сред описал адаптивные MPI-реализации ACO с динамическим управлением размером сообщений.

Несмотря на значительный прогресс в области параллельных реализаций ACO, существует ряд нерешенных проблем. Большинство существующих работ ориентированы на классические комбинаторные задачи (TSP, QAP), где структура графа накладывает жесткие ограничения на параллелизацию. Параметрические же задачи, обладающие свойством независимости параметров, исследованы в этом контексте недостаточно. Кроме того, отсутствует систематическое сравнение эффективности различных моделей параллелизма (SMP, NUMA, SIMT на GPU, MPP) на единой методической базе для параметрических задач. Настоящая работа направлена на восполнение этого пробела. В статье представлены высокопроизводительные реализации матричного ACO для параметрических задач на трех архитектурных моделях: Symmetric Multiprocessing (SMP) с использованием OpenMP и векторизации Advanced Vector Extensions (AVX2), Single Instruction, Multiple Thread (SIMT) на GPU с использованием CUDA, а также распределенная Massively Parallel Processing (MPP)-модель с использованием MPI и гибридная MPI+GPU архитектура. Проведен сравнительный анализ решений на различных вычислительных платформах, включая серверы с Intel Xeon Gold 6130, ускорители NVIDIA Tesla V100, платформу NVIDIA Jetson Orin, а также персональные компьютеры и ноутбуки с GPU различных поколений.

2. Модели и методы

2.1. Параметрический граф и его матричное представление

Классическая постановка ACO предполагает, что муравей-агент перемещается по графу, где вершины соответствуют состояниям (например, городам в TSP), а ребра — возможным переходам. Вероятность перехода из вершины i в вершину j на итерации t определяется правилом случайного пропорционального выбора с применением метода обратной функции распределения:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}, \quad j \in J_i^k,$$

где $\tau_{ij}(t)$ — количество феромона на ребре (i, j) на итерации t; $\eta_{ij} = 1/d_{ij}$ — величина обратная длине ребра (i, j); α, β — параметры алгоритма; J_i^k — множество вершин (ребер (i, j)), в которые может попасть муравей-агент k из вершины i. Так как у каждого муравья свой путь, то для избегания посещения в пути коммивояжера одной вершины дважды необходимо хранить список посещенных вершин (taboo-list). После построения решений всеми муравьями-агентами на итерации происходит глобальное обновление феромонов: испарение $\tau_{ij}(t+1) \leftarrow (1-p)\tau_{ij}(t)$ и добавление $\tau_{ij}(t+1) \leftarrow \tau_{ij}(t) + \sum_{k=1}^{m} Q/L_k$ для ребер, входящих в маршрут k-го муравья-агента. p и Q — параметры алгоритма, а L_k — длина пути k-го муравья-агента [6,7].

Для параметрической задачи, в отличие от TSP, граф решений имеет принципиально иную структуру. Каждый параметр p_i образует отдельный слой, вершины которого соответствуют допустимым значениям $v_{i,j}$. В параметрической задаче решение представляет собой вектор $X = \{x_1, x_2, \ldots, x_n\}$, где каждый компонент x_i определяется независимо из своего множества допустимых значений $V_i = \{v_{1,i}, v_{2,i}, \ldots, v_{m_i,j}\}$. Это позволяет использовать слоистый параметрический граф, в котором каждый слой соответствует одному параметру, а вершины в слое соответствуют его допустимым значениям. Поскольку выбор значения одного параметра не зависит от выбора другого, все ребра между слоями являются фиктивными, а феромон ассоциируется не с ребрами, а с вершинами.

Для работы с непрерывными параметрами выполняется их дискретизация. Однако при высокой точности число значений одного параметра становится чрезмерно большим, что существенно снижает эффективность применения модификаций метода муравьиных колоний [3]. Эффективным вариантом является декомпозиция значения параметра на несколько последовательных слоев, где итоговое значение вычисляется как линейная комбинация выбранных компонент. На рис. 1 показаны два варианта такой декомпозиции для представления числа в диапазоне $[0, 10)$ с точностью до 0.1. Первоначально каждый параметр разбивается на слои с десятичными знаками по 10 вершин (со значениями от 0 до 9) для удобства представления и ускорения сходимости алгоритма.

Дальнейшее разбиение каждого слоя зависит от применяемой параллельной модификации. Для большинства SMP и MPP систем оптимальным является дальнейшее разложения слоя цифр от 0 до 9 на два подслоя по 5 и 2 вершины (рис. 1, граф A). Но в полученном графе максимальное количество вершин равно 5 и не кратно 4, что требует сложных механизмов хранения и выравнивания памяти при параллельной работе. На рисунке 1 справа предложен граф B, в котором максимальное количество вершин в одном слое кратно 4, что существенно увеличивает эффективность memory-bound алгоритмов, но увеличивает число возможных значений, добавляя к интервалу от 0

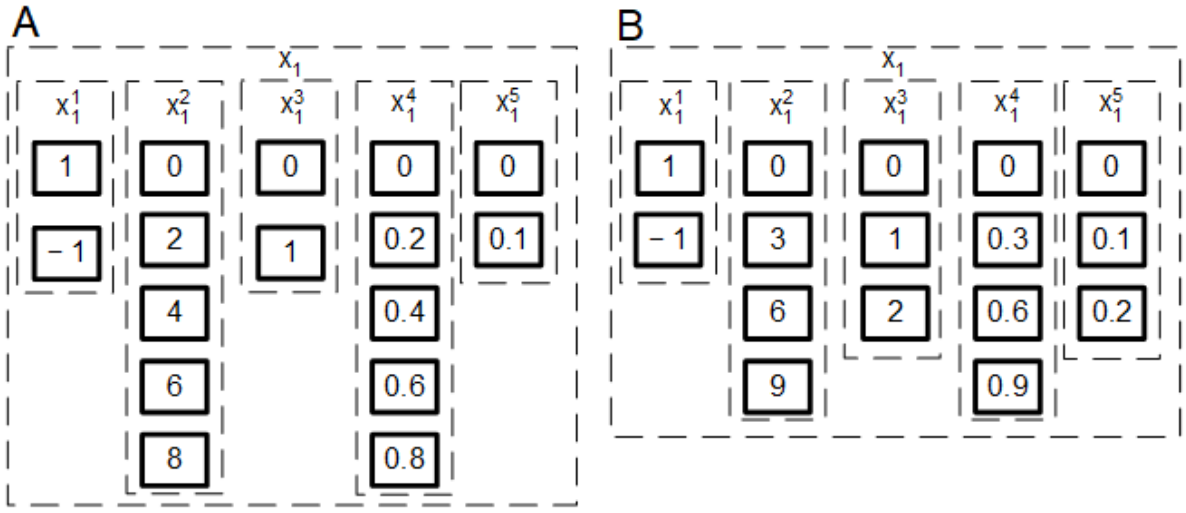


Рис. 1. Примеры декомпозиции непрерывного значения на слоистый граф с максимальным числом вершин в слое равным 5 (граф A) и 4 (граф B).

до 9 еще значения 10 и 11. Тем не менее предложенная структура позволяет убрать дополнительные значения путем добавления условия: Если в текущем слое выбрана четвертая вершина (во всех слоях это число 9 в нужной степени), то в следующем слое обязательно выбирается первая вершина (значение 0).

Для унификации все множества V_i дополняются фиктивными вершинами до одинакового размера $m = \max_i(m_i)$. Тогда состояние метода муравьиных колоний на итерации t полностью описывается следующими матрицами:

- Матрица параметров $V \in \mathbb{R}^{n\times m}$ содержит допустимые значения.
- Матрица феромонов $T(t) \in \mathbb{R}^{n\times m}$ отражает коллективный опыт колонии. Фиктивные вершины имеют значение 0, что не позволяет их выбрать муравьями-агентами.
- Матрица посещений $\Theta(t) \in \mathbb{R}^{n\times m}$ содержит число посещений муравьями-агентами каждой вершины.
- Матрица вероятностей $P(t) \in \mathbb{R}^{n\times m}$ содержит значения вероятности выбора вершин.
- Матрица распределения $F(t) \in \mathbb{R}^{n\times m}$ содержит кумулятивные суммы (функции распределения) для метода обратной функции.
- Матрица путей агентов $N(t) \in \mathbb{N}^{n\times K}$ содержит индексы выбранных вершин для K муравьев-агентов на итерации t.
- Вектор целевых функций $Y(t) \in \mathbb{R}^K$ определяет значения целевой функции $f(X_k)$.

Такое представление позволяет свести итерацию матричной модификации метода муравьиных колоний к последовательности матричных операций, которые эффективно выполняются как на CPU с поддержкой AVX2, так и на GPU с архитектурой CUDA.

Для параметрических задач, где эвристика часто недоступна, это является серьезной проблемой, особенно в условиях где время однократного вычисления целевой функции соизмеримо со временем работы алгоритма поиска, и каждое лишнее обращение к модели ведет к неоправданному росту общего календарного времени решения. Для ее решения в работе [1] предложено семейство модификаций ACO, использующих хэш-таблицу для хранения уже вычисленных решений и изменяющих поведение агента при обнаружении ранее пройденного пути, что позволяет сократить число дорогостоящих вызовов модели за счет повторного использования ранее полученных результатов. Наиболее эффективной для задач с дорогостоящей целевой

функцией оказалась модификация ACOCCyN (ACO Cluster Cycle with N limit). При параллельных вычислениях проверка наличия решения в хэш-таблице хоть и требует дополнительного времени работы вычислителя, но не требует синхронизации, так как операция чтения может быть выполнена параллельно. Но занесение новых значений требует синхронизацию доступа. Использование критических секций неэффективно, так как метод муравьиных колоний большую часть времени исследует новое пространство и постоянно наполняет хэш-таблицу новыми путями. В представленных параллельных алгоритмах применяется подход Compare-and-Swap (CAS) с блокированием отдельных значений. Только если несколько муравьев-агентов на одной итерации нашли новый одинаковый путь требуется повторная проверка значения, вероятность данной ситуации крайне мала.

Эвристика η_{ij} играет ключевую роль, направляя поиск в перспективные области (например, в сторону ближайших городов). Однако в параметрических задачах такая априорная информация часто отсутствует: нет оснований считать одно значение параметра $\tau_{ij}(t)$ «более привлекательным», чем другое, до начала работы алгоритма. В этом случае классическое правило выбора вырождается и приводит к быстрой стагнации: как только какой-либо путь оказывается усилен феромоном, все агенты начинают следовать по нему, игнорируя альтернативные области пространства поиска. Для решения этой проблемы в работах [1, 2] предложено использовать аддитивную свертку, в которой второй член основан на частоте посещений вершин. Модифицированное правило имеет вид:

$$z_{ij}(t) = \lambda_1 \tau_{norm,ij}(t) + \lambda_2 \theta_{ij}(t), \quad p_{ij}(t) = z_{ij}(t) / \sum_{l=1}^{m} z_{il}(t),$$

где $\tau_{norm,ij}(t) = \tau_{ij}(t) / \sum_{l=1}^{m} \tau_{il}(t)$ — нормированное значение феромона для значения i параметра j на итерации t; $\theta_{ij}(t)$ — количество выборов муравьями-агентами значения i параметра j на итерации t; λ_1, λ_2 — весовые коэффициенты аддитивной свертки (в работе $\lambda_1 = \lambda_2 = 1$).

Экспериментальные исследования [2, 3] показали, что предложенное правило позволяет находить глобальный оптимум на многомерных тестовых функциях с вероятностью до 95% при числе итераций, в 2–3 раза меньшем, чем для классического ACO с мультипликативным правилом и случайной инициализацией феромонов.

2.2. Матричные модификации метода муравьиных колоний

Каждая итерация матричной модификации метода муравьиных колони состоит из трех последовательных фаз. На первой фазе по текущим матрицам феромонов и посещений вычисляется кумулятивная матрица вероятностей. Эта операция сводится к поэлементной обработке строк матриц и выполняется параллельно для всех параметров. На второй фазе каждый муравей-агент независимо строит решение: для каждого параметра генерируется случайное число, и по кумулятивной матрице вероятностей методом обратной функции определяется индекс выбранного значения. Сформированный вектор параметров проверяется в хэш-таблице, хранящей уже вычисленные результаты, что позволяет избежать повторных дорогостоящих вызовов модели при повторном посещении агентом уже исследованной комбинации. Если в хэш-таблице значения не найдены, происходит запуск модели и вычисление целевой функции. На третьей фазе выполняется обновление феромонной матрицы: все значения умножаются на коэффициент испарения ρ, после, для каждого агента на вершины, составляющие его решение, добавляется количество феромона, обратно пропорциональное значению целевой функции (для задач минимизации). Одновременно

обновляется матрица посещений. За счет небольшого количества итераций внутренних циклов, рассматривающих все значения одного слоя (5 или 4 итерации цикла в зависимости от структуры параметрического графа) эффективным становится применение директивы компилятора `#pragma unroll`. Алгоритм повторяет описанный цикл до достижения заданного числа итераций или выполнения критерия сходимости [30].

Благодаря матричной формализации алгоритм естественным образом распараллеливается по двум измерениям: по агентам (каждый агент строит решение независимо) и по параметрам (выбор значения для каждого параметра не зависит от других). В CUDA-реализации каждый блок потоков соответствует одному агенту, а каждый поток внутри блока обрабатывает один параметр. Подобная организация обеспечивает коалесцированный доступ к глобальной памяти и позволяет эффективно использовать иерархию памяти GPU. В случае, когда число параметров n превышает максимально допустимое количество потоков в одном блоке (обычно 1024 для современных GPU), используется стратегия циклического распределения параметров между потоками. Каждый поток обрабатывает несколько параметров, последовательно проходя по ним в цикле. При этом сохраняется коалесцированный доступ к памяти, поскольку соседние потоки обрабатывают параметры со сдвигом, равным размеру блока, что обеспечивает обращение к последовательным адресам памяти. Для повышения эффективности применяется техника тайлинга (tiling). Параметры разбиваются на блоки (тайлы) фиксированного размера, кратного размеру варпа (32), что позволяет загружать данные в разделяемую память и повторно использовать их при обработке нескольких параметров. Ключевой оптимизацией является применение warp-редукции при обновлении феромонов. Вместо того чтобы каждый поток выполнял атомарное добавление в глобальную память, 32 потока внутри варпа накапливают дельты в регистрах, затем с помощью shuffle-инструкций выполняют горизонтальное суммирование, и только первый поток варпа осуществляет атомарную запись результата. Это сокращает количество конфликтующих атомарных операций в 32 раза и существенно повышает пропускную способность [31].

В отличие от GPU-реализации, OpenMP-версия, использует крупнозернистый параллелизм на уровне циклов для матричной реализации метода муравьиных колоний, разделенной на три фазы. Потоки выполняются асинхронно и синхронизируются только в критических секциях, что накладывает ограничения на масштабируемость: при увеличении числа потоков свыше 12–16 эффективность начинает снижаться из-за накладных расходов на синхронизацию и конфликтов доступа к общей памяти. Для минимизации этих конфликтов хэш-таблица разделена на независимые сегменты (шарды), число которых равно числу потоков, что позволяет каждому потоку работать только со своим шардом и полностью исключить атомарные операции при поиске и вставке. Обновление феромонов организовано по принципу локального накопления: каждый поток накапливает изменения в приватных векторах, а после завершения цикла по агентам выполняет слияние в глобальные матрицы в критической секции. Генерация случайных чисел в OpenMP-реализации выполняется с использованием генератора `std::mt19937_64`, где каждый поток инициализирует собственный экземпляр с уникальным seed, зависящим от номера потока и номера итерации, что обеспечивает независимость последовательностей и воспроизводимость результатов.

В разработанной OpenMP-реализации матричного ACO ключевым элементом повышения производительности является использование векторных инструкций AVX2 (Advanced Vector Extensions 2), поддерживаемых современными x86-процессорами. AVX2 оперирует 256-битными регистрами YMM, позволяя выполнять одну операцию над четырьмя числами с плавающей точкой двойной точности (double) одновремен-

но. В контексте матричной формализации ACO это свойство оказывается особенно ценным, однако требует применения специальной структуры параметрического графа. В отличие от универсальной конфигурации с произвольным числом вершин в слое, для эффективного использования AVX2 необходимо обеспечить фиксированное число значений параметра в слое, равное четырем ($m = 4$). При $m = 4$ каждая строка матрицы феромонов, посещений или вероятностей идеально соответствует ширине AVX2-регистра, позволяя загружать и обрабатывать ее за одну векторную инструкцию. Векторизация применяется на всех этапах алгоритма: при нормировке феромонов (вычисление суммы строки и поэлементное деление с использованием `mm256_hadd_pd` и `mm256_div_pd`), при расчете вероятностей (аддитивная свертка с инвертированными значениями посещений), при обновлении феромонов (испарение с помощью `mm256_mul_pd`) и при выборе значения параметра агентом, где векторное сравнение с четырьмя кумулятивными вероятностями с использованием `mm256_cmp_pd` и `mm256_movemask_pd` заменяет последовательный поиск. В OpenMP-реализации векторизация достигается за счет использования intrinsic-функций внутри каждого потока, что в сочетании с многопоточностью обеспечивает двухуровневый параллелизм: крупнозернистый (распределение агентов и параметров между ядрами CPU) и мелкозернистый (векторная обработка внутри каждого ядра).

В отличие от CPU, GPU-реализация не использует AVX2-инструкции, поскольку архитектура CUDA оперирует собственной SIMT-моделью, где 32 потока в варпе выполняют одинаковые инструкции над разными данными, что является аппаратной векторизацией на уровне варпа, не требующей явного использования AVX2.

Для MPP систем, представляющих собой кластеры вычислительных узлов, разработана гибридная MPI+GPU реализация матричной модификации метода муравьиных колоний, использующая островную модель параллелизма. В этой архитектуре каждый MPI-процесс имеет полную копию параметрического графа (матрицы феромонов, посещений и параметров) и управляет собственным графическим ускорителем, на котором выполняется вычисление целевой функции для своей подпопуляции муравьев-агентов. После каждой итерации выполняется коллективная операция `MPI_Reduce` (для сбора статистики) или `MPI_Allgather` (для синхронизации феромонных матриц), что позволяет объединить опыт всех колоний и обновить глобальное состояние. Для снижения коммуникационной нагрузки применяется дельта-компрессия: передаются только изменения феромонов, превышающие заданный порог, что особенно эффективно на поздних итерациях, когда феромонная матрица стабилизируется. Каждый MPI-процесс инициализирует собственный GPU с помощью `cudaSetDevice(rank % num_gpus_per_node)`, что обеспечивает равномерное распределение вычислительной нагрузки между ускорителями в многоузловой конфигурации. При этом на отдельных GPU выполняется весь алгоритм с обменом и редукцией матриц посещений и феромонов, полученных муравьями-агентами на итерации, но хэш-таблица на каждом GPU не синхронизируется. При этом исследовалась как централизованная модель, в которой редукция выполняется только на главном процессе (rank=0) с последующей рассылкой итоговой матрицы, так и распределенной модели с пересылкой командами `MPI_Allgather` для синхронного варианта или асинхронный протокол с использованием `MPI_Isend` и `MPI_Irecv` с последующим `MPI_Waitall`.

2.3. Оценка вычислительной сложности матричной формализации

Метод муравьиных колоний для задачи TSP имеет итеративную сложность $O(K \cdot n^2)$, где K – число муравьев-агентов на итерации, а n – количество вершин в графе. Квадрат возникает из-за того, что для построения полного маршрута из n вершин агент выполняет n шагов со сложностью каждого шага $O(n)$. Применение методов вращения рулеточного колеса ослабляет эту зависимость, но не нивелирует. Матричная модификация метода муравьиных колоний выполняет выбор каждой вершины в слое независимо от выбора других вершин $O(K \cdot n \cdot m)$, где m – количество вершин в одном слое. Применение AVX инструкций при $m = 4$ позволяет осуществлять векторные операции за один такт, т.е. $O(K \cdot n)$. переход от квадратичной зависимости $O(n^2)$ к линейной $O(n)$ является фундаментальным преимуществом матричной формализации для параметрических задач. Это позволяет решать задачи с числом параметров $n \geq 10^5$ и более за приемлемое время. К тому же в отличие от TSP, где сложность определяется плотностью графа, в параметрической задаче сложность зависит только от числа параметров и фиксированного m, что делает алгоритм предсказуемо масштабируемым. А использование векторных инструкций (AVX2) и массового параллелизма (CUDA) позволяет дополнительно снизить сложность, обеспечивая практическое ускорение в десятки и сотни раз по сравнению с последовательной реализацией.

3. Эксперимент и результаты

3.1. Экспериментальная установка и методология оценки

Экспериментальное исследование разработанных параллельных реализаций матричной модификации метода муравьиных колоний проводилось на различных вычислительных платформах, охватывающих широкий спектр современных архитектур: от энергоэффективных встраиваемых систем до высокопроизводительных серверных ускорителей, от мобильных процессоров разных поколений до многопроцессорных кластерных конфигураций:

- NVIDIA Tesla V100 — эталонная конфигурация с 5120 ядрами CUDA и 32 ГБ HBM2-памяти.
- NVIDIA GTX 1050Ti (архитектура Pascal) — 768 ядер CUDA, 4 ГБ GDDR5.
- NVIDIA RTX 4060Ti (архитектура Ada Lovelace) — 4352 ядра CUDA, 8 ГБ GDDR6.
- NVIDIA Jetson Orin — 1792 ядра CUDA, энергопотребление не более 15 Вт.

Ключевые оптимизации CUDA-реализации достигаются применением директив компилятора `-use_fast_math` для ускорения математических операций, `-maxrtegcount` для ограничения числа используемых регистров и `-Xptxas -O3,-v` для контроля оптимизаций PTX-кода. Использовался компилятор `nvcc` версии 12.6.

Для оценки масштабируемости по данным гибридной MPI+GPU реализации использовалась многопроцессорная (MPP) система на базе сервера DELL POWEREDGE C4140, обладающая четырьмя графическими ускорителями NVIDIA Tesla V100 16 ГБ SXM2, объединенные через NVLink и установленные на одной плате. Благодаря размещению всех четырех ускорителей на одной плате и использованию высокоскоростной шины NVLink, задержки при обмене данными между GPU в MPP-реализации являются незначительными по сравнению с вычислительной нагрузкой.

Для CPU-реализаций использовались пять различных систем:

- Сервер DELL POWEREDGE C4140 с Intel Xeon Gold 6130 2.1 GHz (Skylake Server,

2017) 16 ядер (2 процессора – 32 ядра, 64 логических потока), 256ГБ RAM DDR4-2666, размеры кэшей одного процессора L3-22.0Мб, L2-16.0Мб (16384/16=1024 Кб на ядро), L1-1024Кб.
- 12th Gen Intel Core i5-12450H 2.0 GHz (Adler Lake с ядрами Golden Cove + Gracemont, 2021) 8 ядер 12 логических потоков, 16Гб RAM DDR5-3200, размеры кэшей L3-12.0Мб, L2-7.0Мб (7168/8=896 Кб на ядро в среднем, а точнее, 1.25 МБ на P-core (4 ядра) и 2 МБ на E-кластер, по 512Кб на E-core), L1-704Кб (для P-core 48Кб и классические 32Кб для E-core).
- 10th Gen Intel Core i5-10400F 2.9 GHz (Comet Lake, 2020) 6 ядер 12 логических потоков, 16Гб RAM DDR4-2600, размеры кэшей L3-12.0Мб, L2-1.5Мб (1536/6=256 Кб на ядро), L1-384Кб.
- 9th Gen Intel Core i5-9400F 2.9 GHz (Coffee Lake Refresh, 2019) 6 ядер, 16Гб RAM DDR4-2933, размеры кэшей L3-9.0Мб, L2-1.5Мб (1536/6=256 Кб на ядро), L1-384Кб.
- 8th Gen Intel Core i5-8300H 2.3 GHz (Coffee Lake, 2018) 4 ядра 8 логических потоков, 8Гб RAM DDR4-2666, размеры кэшей L3-8.0Мб, L2-1.0Мб (1024/4=256 Кб на ядро) L1-256Кб.
- AMD Ryzen 5 7500F 3.7GHz (Zen 4, Raphael, 2023) 6 ядер с SMT, 12 логических потоков, 32Гб RAM DDR5-6000, размеры кэшей L3-32.0Мб, L2-6.0Мб (6144/6=1024 Кб на ядро), L1-384Кб.
- AMD Ryzen 9 7900 3.7GHz (Zen 4, Raphael, 2023) 12 ядер (2 кристалла CCD) с SMT, 24 логических потоков, 32Гб RAM DDR5-6000, размеры кэшей L3-64.0Мб (2 кристалла CCD по 32Мб, объединенных через Infinity Fabric), L2-12.0Мб (12288/12=1024 Кб на ядро), L1-768Кб.

Выбор мобильных процессоров обусловлен не только их широкой распространенностью, но и возможностью оценить влияние тепловых ограничений, турбобуста и гибридной архитектуры (P-cores/E-cores) на масштабируемость по данным и энергоэффективность матричного ACO. Директива компиляции, использованная для сборки программного модуля, имеет следующий вид: `clang++ -target x86_64-pc-windows-msvc -std=c++17 -fopenmp=libomp -O3 -maxv2 -march=native -o <executable_name> <source_file.cpp>`. Применяется компилятор LLVM Clang 21.1.4.

Все тесты проводились для 500 итераций работы метода муравьиных колоний по 500 муравьев-агентов. Результаты приводятся для поиска оптимального решения для функции Растригина. Приведенные показатели являются нормированными к одному муравью-агенту и одному слою, что позволяет корректно сравнивать результаты при варьировании этих параметров в широких диапазонах (число муравьев-агентов от 256 до 16 миллионов, число слоев от 42 до 1.38 миллиона). Для каждого измерения проводилось более 50 повторных запусков с вычислением среднего арифметического и стандартного отклонения. Программное обеспечение и результаты работы для различных модификаций на различных платформах находятся в репозитории `https://github.com/kalengul/ACO_SIMD`.

3.2. Анализ производительности CUDA-реализаций на различных поколениях GPU

Для анализа производительности реализации определим масштабируемость алгоритма по данным изменяя размерность задачи. На серверном ускорителе Tesla V100 наблюдается монотонное уменьшение времени выполнения 500 итераций 500 муравьями-агентами с 2.05 мс при 42 слоях до 0.097 мс при 43008 слоях, после чего наступает

плато в диапазоне 0.10–0.12 мс до 1.38 миллиона слоев. Этот эффект объясняется тем, что увеличение числа слоев требует обработки большего объема данных (матрицы феромонов, вероятностей, посещений), что позволяет полностью загрузить вычислительные ресурсы GPU, минимизируя накладные расходы на запуск ядер, синхронизацию варпов и латентность доступа к глобальной памяти. При малом числе слоев (менее 100–200) GPU оказывается существенно недогружен: количество одновременно активных варпов значительно меньше максимально возможного, многие потоковые мультипроцессоры простаивают, а преобладающими становятся латентности передачи управления и конфликты при доступе к памяти. Начиная с 1000–2000 слоев для V100 достигается режим насыщения, при котором дальнейшее наращивание размерности задачи не приводит к линейному росту абсолютного времени выполнения.

Для массового GPU GTX 1050Ti, относящегося к архитектуре Pascal с пропускной способностью памяти 112 ГБ/с минимальное нормированное время достигается при 2688–5376 слоях (0.447–0.453 мс), а при дальнейшем росте размерности время не возрастает, но и не снижается, оставаясь стабильным на уровне 0.45–0.48 мс вплоть до исчерпания видеопамяти при 344064 слоях. Это свидетельствует о том, что архитектурные ограничения по пропускной способности памяти и количеству одновременно активных варпов на GTX 1050Ti становятся узким местом значительно раньше, чем на V100. При малом числе слоев (менее 672) наблюдается существенно более высокое нормированное время (до 2.43 мс при 42 слоях), что объясняется не только недогрузкой GPU, но и накладными расходами на организацию параллельных циклов при малой размерности. Интересно отметить, что на GTX 1050Ti зависимость времени от числа слоев не является монотонной: при переходе от 42 к 84 слоям время уменьшается более чем вдвое (с 2.43 до 1.65 мс), а затем продолжает снижаться, но уже с меньшим темпом. Это связано с тем, что при очень малом числе слоев доминирующий вклад вносят накладные расходы на запуск CUDA-ядер и передачу данных между хостом и устройством, которые практически не зависят от размерности задачи и размазываются по большему числу обрабатываемых элементов при росте числа слоев.

Высокую эффективность демонстрирует современный ускоритель RTX 4060Ti, построенный на архитектуре Ada Lovelace с пропускной способностью памяти 288 ГБ/с и поддержкой технологии Shader Execution Reordering (SER). При сверхмалом числе слоев (42–84) этот GPU обеспечивает время 0.83–0.60 мс, что в 2.5–2.8 раза быстрее Tesla V100 (2.05–1.12 мс) и в 2.9–2.7 раза быстрее GTX 1050Ti (2.43–1.65 мс). Однако уже при 168 слоях соотношение меняется: Tesla V100 (0.584 мс) оказывается быстрее RTX 4060Ti (0.736 мс) примерно на 21% и быстрее GTX 1050Ti (1.50 мс) в 2.6 раза. В диапазоне 336–672 слоев Tesla V100 сохраняет лидерство (0.309–0.182 мс), тогда как RTX 4060Ti демонстрирует 0.467–0.280 мс, а GTX 1050Ti — 1.08–0.667 мс.

При дальнейшем росте размерности (10752–86016 слоев) Tesla V100 вновь выходит вперед, обеспечивая преимущество в 15–40% перед RTX 4060Ti благодаря более высокой пропускной способности HBM2-памяти (1555 ГБ/с против 288 ГБ/с у GDDR6). RTX 4060Ti демонстрирует рост времени до 0.169–0.173 мс, тогда как V100 остается в пределах 0.100–0.107 мс.

При 172032 слоях RTX 4060Ti показывает 0.154 мс, что лишь незначительно уступает V100 (0.112 мс) с учетом близости к границе видеопамяти ускорителя (8 ГБ). Начиная с 344064 слоев RTX 4060Ti, как и GTX 1050Ti, испытывает нехватку видеопамяти, тогда как Tesla V100 с 40 ГБ HBM2 продолжает уверенно работать вплоть до 1.38 миллиона слоев, сохраняя время на уровне 0.12–0.13 мс, что подтверждает масштабируемость по данных для серверных ускорителей.

Платформа Jetson Orin, ориентированная на мобильные применения и работающая в режиме низкого энергопотребления 7–15 Вт, демонстрирует стабильное время около 1.48–1.51 мс при числе слоев от 336 до 10752, что примерно в 12–15 раз медленнее Tesla V100, но при энергопотреблении на один-два порядка меньшем. Особенно показательно, что на Jetson Orin отсутствует ярко выраженный режим насыщения: нормированное время остается практически постоянным (1.48–1.62 мс) в широком диапазоне слоев от 336 до 43008. Этот результат имеет важное практическое значение, показывая, что предложенный матричный ACO может эффективно работать на встраиваемых GPU-ускорителях, сохраняя приемлемую производительность при кардинально более низком энергопотреблении.

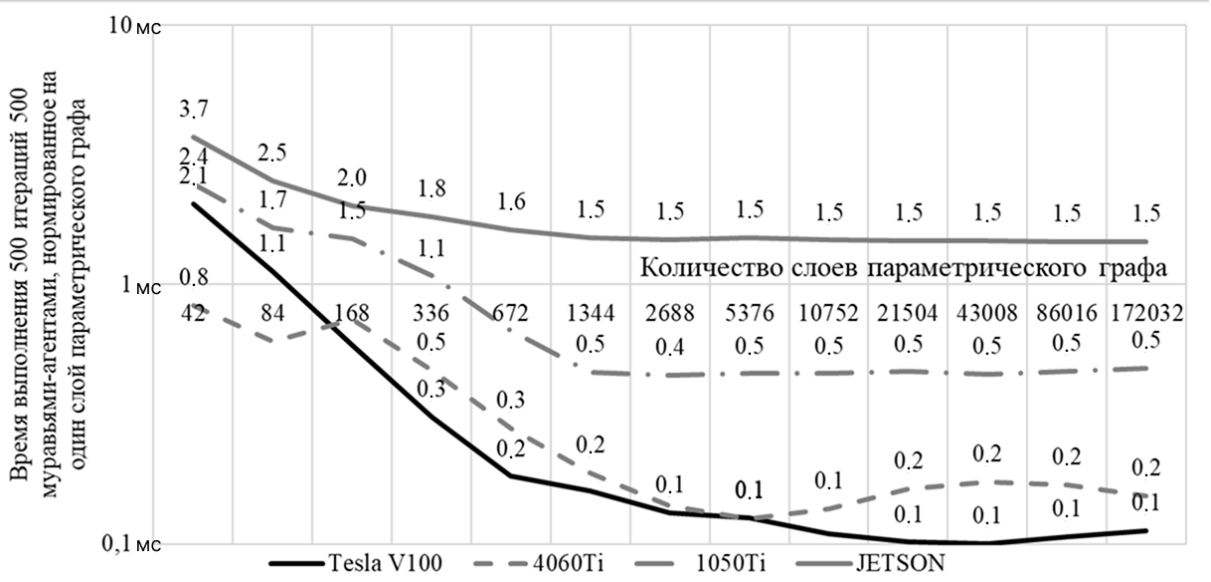


Рис. 2. Оценка времени (в мс) работы матричной реализации метода муравьиных колоний на различных ускорителях.

Наиболее детальный анализ масштабируемости по данным для CUDA-реализации при увеличении числа муравьев-агентов представлен в таблице 1 для ускорителя NVIDIA GTX 1050Ti. Эксперимент охватывал числа агентов от 256 до 2 миллионов и числа слоев параметрического графа от 42 до 344064, причем для каждой конфигурации измерялось нормированное время выполнения 500 итераций в пересчете на одного агента и один слой.

Область недогрузки GPU наблюдается при числе агентов менее 1024 для всех размерностей слоев. В этой области нормированное время монотонно снижается с ростом числа агентов: например, для графа с 42 слоями время падает с 5.80×10^{-3} мс при 256 агентах до 2.33×10^{-3} мс при 2 миллионах агентов, то есть более чем вдвое. При этом скорость снижения максимальна в диапазоне от 256 до 4096 агентов, а затем замедляется, приближаясь к асимптоте. Область насыщения, наступающая при достижении числа агентов, достаточного для полной загрузки всех потоковых мультипроцессоров GPU. Для GTX 1050Ti этот порог составляет примерно 4096–8192 агентов, после чего нормированное время стабилизируется на уровне 7.9×10^{-4} – 8.4×10^{-4} мс для графов с числом слоев более 672. Важно подчеркнуть, что в режиме насыщения дальнейшее увеличение числа агентов (до 262144 и более) практически не влияет на нормированное время и оно остается в пределах 7.8×10^{-4} – 8.2×10^{-4} мс, что свидетельствует о совершенной линейной масштабируемости по данным матричного

ACO на GPU. Деградация и исчерпание памяти, наблюдаемая при комбинациях большого числа слоев и большого числа агентов. Для графа с 344064 слоями уже при 256 агентах возникает нехватка видеопамяти, а для графа с 172032 слоями этот порог составляет 4096 агентов.

Таблица 1. Оценка времени (в мс) выполнения 500 итераций метода муравьиных колоний, нормированное по числу муравьев-агентов и количеству слоев параметрического графа на ускорителе NVIDIA GTX 1050Ti.

Количество	Количество слоев параметрического графа							
муравьев-агентов	42	168	672	2688	10752	43008	172032	344064
256	5.80E-03	3.56E-03	1.47E-03	9.29E-04	9.44E-04	9.13E-04	9.66E-04	9.82E-04
512	4.87E-03	3.00E-03	1.33E-03	8.94E-04	9.07E-04	9.02E-04	9.50E-04	
2048	2.65E-03	2.96E-03	1.23E-03	8.13E-04	8.12E-04	7.96E-04		
8192	2.43E-03	2.85E-03	1.18E-03	7.93E-04	7.85E-04			
32768	2.33E-03	2.83E-03	1.17E-03	7.89E-04				
131072	2.32E-03	2.83E-03	1.17E-03			Недостаточно памяти		
524288	2.33E-03	2.84E-03						
2097152	2.33E-03							

Анализ таблицы 1 также позволяет выявить оптимальную конфигурацию для GTX 1050Ti: минимальное нормированное время 7.85×10^{-4} мс достигается при 8192–16384 агентах и 5376–10752 слоях, что обеспечивает полную загрузку GPU без избыточного потребления памяти. Для графов с числом слоев менее 672 эффективность существенно ниже из-за недогрузки, а для графов с числом слоев более 21504 резко возрастают требования к памяти, ограничивая максимальное число агентов.

3.3. Анализ производительности OpenMP и AVX2 реализаций

Для оценки влияния архитектурных особенностей центральных процессоров, а также эффективности векторизации AVX2 на производительность матричного метода муравьиных колоний, было проведено систематическое тестирование на семи процессорных конфигурациях, охватывающих различные поколения и микроархитектуры: от мобильных Coffee Lake (2018) до серверного Skylake (2017) и настольных Zen 4 (2023) (таблица 2). Для каждого процессора представлены две реализации: базовая OpenMP (крупнозернистый параллелизм по агентам и параметрам) и OpenMP с дополнительной ручной векторизацией AVX2, использующей intrinsic-функции и оптимизацию структур данных под ширину регистра YMM (4 числа двойной точности). Исследуется время работы при максимальной загрузке CPU.

На малых размерностях параметрического графа (42–168 слоев) определяющим фактором является тактовая частота и эффективность работы с кэш-памятью первого уровня: здесь лучшие результаты (0.65–0.94 мс) демонстрируют Intel i5-10400F и AMD Ryzen 5 7500F, тогда как мобильный i5-8300H и гибридный i5-12450H уступают в 1.5–3 раза из-за более низких тактовых частот в режиме длительной нагрузки и, в случае Alder Lake, накладных расходов на диспетчеризацию задач между P- и E-ядрами. С ростом размерности графа до 672–2688 слоев картина выравнивается: все процессоры выходят на режим насыщения, при котором нормированное время снижается до 0.5–0.8 мс, а преимущество архитектуры AMD Zen 4 (Ryzen 5 7500F и Ryzen 9 7900) становится очевидным так как они достигают 0.36 и 0.35 мс соответственно, обходя

Intel Xeon Gold 6130 (0.81 мс) более чем в два раза. Это связано с большим объемом кэш-памяти L3 на ядро (1 МБ против 0.7 МБ у Xeon и 0.5–0.7 МБ у мобильных Intel) и более эффективной работой с SIMD-инструкциями. Применение векторных инструкций AVX2 вносит существенный, но немонотонный вклад в производительность. Наибольший выигрыш от векторизации наблюдается в диапазоне 672–10752 слоев, где все строки матриц феромонов и вероятностей полностью помещаются в кэш L3. Так, для Ryzen 5 7500F на 2688 слоях AVX2 снижает нормированное время с 0.34 до 0.22 мс — ускорение в 1.55 раза, а для Ryzen 9 7900 на 10752 слоях — с 0.32 до 0.27 мс. На процессорах Intel эффект скромнее: для i5-10400F ускорение составляет 1.4–1.5 раза, для i5-9400F — 1.3–1.5 раза.

Таблица 2. Оценка времени (в мс) выполнения 500 итераций по 500 муравьев-агентов метода муравьиных колоний, нормированное по количеству слоев параметрического графа.

Процессор	Число ядер	Алгоритм	42	168	672	2688	10752	21504	43008	86016
Intel i5-8300H	4+4HT	OMP	1.09	0.79	0.76	0.82	1.01	1.22	2.15	2.75
		OMP+AVX2	2.08	1.17	0.57	0.64	0.73	0.94	2.18	2.77
Intel i5-9400F	6	OMP	1.21	0.84	0.79	0.74	0.78	0.80	1.19	1.58
		OMP+AVX2	1.06	0.64	0.51	0.49	0.54	0.55	1.00	1.41
Intel i5-10400F	6+6HT	OMP	0.94	0.72	0.64	0.63	0.64	0.84	1.62	1.86
		OMP+AVX2	0.68	0.50	0.43	0.42	0.48	0.60	1.21	1.47
Intel i5-12450H	4P+4HT+4E	OMP	3.08	1.51	0.87	0.64	0.59	0.62	0.81	1.14
		OMP+AVX2	2.79	1.24	0.85	0.48	0.45	0.47	0.67	1.02
Ryzen 5 7500F	6+6SMT	OMP	0.65	0.42	0.36	0.34	0.35	0.38	0.48	0.69
		OMP+AVX2	0.67	0.31	0.24	0.22	0.24	0.25	0.28	0.45
Ryzen 9 7900	12+12SMT	OMP	0.89	0.54	0.35	0.40	0.32	0.34	0.36	0.52
		OMP+AVX2	0.78	0.35	0.29	0.26	0.27	0.30	0.31	0.46
Intel Xeon Gold 6130	16×2=32	OMP	3.71	1.52	0.81	0.54	0.51	0.50	0.56	0.67

Экспериментальное исследование OpenMP-реализации матричного ACO на двухпроцессорной системе с 24 задействованными ядрами при варьировании числа муравьев-агентов и числа слоев параметрического графа позволяет оценить масштабируемость по данным для OpenMP реализации ACO (Таблица 3). Компилятор gcc версии 9.1 с флагами `-O3 -fopenmp -mavx2 -mfma -march=native`. Существует нелинейная зависимость нормированного времени от числа агентов. При малом числе муравьев-агентов (256–1024) время для графов с малым числом слоев (42–168) составляет 1.25×10^{-2} – 6.63×10^{-3} мс и снижается с ростом числа муравьев-агентов примерно по гиперболическому закону, что соответствует классической модели ускорения Амдала при доминировании последовательной части. Однако уже при 2048–4096 муравьях-агентах для графов с 672–21504 слоями нормированное время достигает минимума в диапазоне 6.8×10^{-4} – 7.9×10^{-4} мс, после чего остается практически постоянным вплоть до 8 миллионов агентов. Это свидетельствует о том, что на CPU-реализации также достигается режим насыщения, при котором все ядра полностью загружены, а дальнейшее увеличение числа муравьев-агентов не приводит к снижению времени на одного агента поскольку они обрабатываются последовательно в циклах, но уже с максимально возможным параллелизмом.

Таблица 3. Оценка времени выполнения 500 итераций метода муравьиных колоний, нормированное по числу муравьев-агентов и количеству слоев параметрического графа на 24 ядрах Intel Xeon Gold 6130 2.1 GHz 16 Core.

Количество	Количество слоев параметрического графа							
муравьев-агентов	42	168	672	2688	10752	43008	172032	1376256
256	1.25E-02	4.64E-03	2.19E-03	1.50E-03	1.34E-03	1.37E-03	2.28E-03	2.32E-03
512	8.18E-03	3.29E-03	1.63E-03	1.22E-03	1.01E-03	1.05E-03	1.78E-03	1.85E-03
1024	6.63E-03	2.43E-03	1.18E-03	9.89E-04	8.05E-04	8.73E-04	1.55E-03	1.37E-03
8192	4.39E-03	1.43E-03	8.12E-04	7.14E-04	6.60E-04	7.37E-04	1.33E-03	
32768	4.29E-03	1.21E-03	7.14E-04	6.73E-04	6.49E-04	1.02E-03		
131072	3.92E-03	1.30E-03	7.42E-04	6.62E-04	6.89E-04			
524288	3.75E-03	1.21E-03	7.19E-04	6.94E-04				
2097152	3.90E-03	1.23E-03	7.25E-04			Недостаточно памяти		
8388608	3.83E-03	1.11E-03						
16777216	3.61E-03							

Важно отметить, что для графов с очень большим числом слоев (86016–1376256) наблюдается иной характер зависимости: при увеличении числа муравьев-агентов от 256 до 16384 нормированное время снижается с 2.14×10^{-3} до 1.24×10^{-3} мс, но затем, при дальнейшем росте до 262144 муравьев-агентов, остается стабильным на уровне 1.2×10^{-3} – 1.4×10^{-3} мс, что примерно в 1.5–2 раза выше, чем для графов средней размерности. Это объясняется тем, что при очень большом числе слоев матрицы феромонов и вероятностей перестают помещаться в кэш-память L3 (22 МБ на процессор Xeon Gold 6130), и алгоритм начинает работать в режиме memory-bound с частыми обращениями к оперативной памяти, что существенно снижает эффективность параллелизации. Критически важным результатом является то, что OpenMP-реализация на 24 ядрах Xeon Gold 6130 демонстрирует практически идеальную эффективность по числу агентов в широком диапазоне (от 1024 до 8 миллионов) при условии, что размерность графа не превышает объема кэш-памяти. Для графов с числом слоев 10752 и 21504 нормированное время составляет 6.45×10^{-4} – 7.13×10^{-4} мс, что лишь немногим больше минимального значения, достигнутого на GPU GTX 1050Ti (7.85×10^{-4} мс).

Исследовалось ускорение работы от применения AVX2 инструкций при различной загруженности вычислителя, которое рассчитывалось как отношение времени выполнения скалярной (без AVX2) версии к векторной (с AVX2). Максимальное ускорение (до 1.85 раз) достигается при использовании одного ядра и средних размерностях графа (672–5376 слоев) (Таблица 4). При этом ускорение растет с увеличением числа слоев от 1.58 раз (42 слоя) до 1.85 раз (672 слоя), после чего остается стабильным на уровне 1.84–1.85 раз вплоть до 5376 слоев, а затем начинает медленно снижаться до 1.77 раз при 86016 слоях.

Теоретически AVX2 может давать до 4-кратного ускорения за счет обработки 4 чисел двойной точности за одну инструкцию. Однако на практике ускорение ограничено последовательными фрагментами алгоритма (закон Амдала), накладными расходами на загрузку/выгрузку данных из YMM-регистров, узкой пропускной способностью памяти при больших размерностях графа и неполной векторизацией сложных циклов компилятором. Достигнутое ускорение 1.85× является типичным для реальных при-

Таблица 4. Ускорение, полученное в результате применения AVX инструкций совместно с OpenMP на Intel i5-9400F 6 ядер. Компилятор clang MSVC.

Ядра	Размерность параметрического графа											
	42	84	168	336	672	1344	2688	5376	10752	21504	43008	86016
1	1.58	1.67	1.77	1.80	1.85	1.84	1.85	1.84	1.81	1.81	1.81	1.77
2	1.46	1.61	1.72	1.79	1.83	1.83	1.82	1.81	1.81	1.80	1.78	1.70
3	1.39	1.57	1.68	1.78	1.81	1.81	1.82	1.81	1.79	1.79	1.75	1.54
4	1.31	1.54	1.68	1.77	1.79	1.82	1.82	1.80	1.78	1.78	1.62	1.39
5	1.29	1.52	1.66	1.75	1.78	1.80	1.81	1.78	1.76	1.76	1.45	1.27
6	1.17	1.49	1.51	1.74	1.76	1.80	1.81	1.78	1.76	1.73	1.33	1.20

ложений с нерегулярной структурой данных и представляет собой хороший результат, полученный без изменения алгоритма, только за счет использования intrinsic-функций и оптимизации структур данных под 4-элементные векторы.

3.4. Моделирование работы суррогатной модели и задержки времени при вычислении значений целевой функции

Для оценки эффективности параллельной реализации метода муравьиных колоний в условиях реальной вычислительной нагрузки, проведено моделирование добавленной задержки, имитирующей работу как имитационных или аналитических моделей по поиску значений целевой функции, так и времени добавленной суррогатной модели, позволяющей эффективнее осуществлять оптимизацию методом ACO. В экспериментах варьировалось время однократного вычисления целевой функции в диапазоне от 25 до 500 условных единиц. Второй тип — суррогатная модель (гауссовский процесс, случайный лес, градиентный бустинг), которая используется для быстрой предварительной оценки перспективности решений. В экспериментах задержка суррогатной модели варьировалась в том же диапазоне от 25 до 500 условных единиц, поскольку даже «быстрые» суррогаты при работе с большим числом опорных точек или глубокими деревьями могут требовать времени, соизмеримого с полномасштабной целевой функцией. Важно подчеркнуть, что исследование не привязано к конкретной предметной области, а моделирует универсальную вычислительную нагрузку, позволяя оценить масштабируемость самого алгоритма ACO при добавлении внешних вычислений, которые могут быть плохо параллелизуемы или требовать синхронизации.

При минимальной задержке (25 итераций) для суррогатной модели на размерностях графа 672–21504 слоев ускорение на 4 потоках относительно одного потока достигает 3.2–3.4 раза, что соответствует эффективности параллелизации 80–85%. При увеличении задержки до 500 итераций абсолютное время выполнения растет пропорционально нагрузке, однако относительное ускорение на 4 потоках снижается до 2.6–2.9 раза (эффективность 65–72%). Это снижение объясняется тем, что даже относительно легкая суррогатная модель при высоких задержках начинает вносить заметную последовательную компоненту (например, обновление общей модели или синхронизацию доступа к обучающей выборке). Правый график демонстрирует моделирование значительно более тяжелой вычислительной нагрузки, соответствующей многократным вызовам имитационной или аналитической целевой функции (те же диапазоны задержек 25–500 итераций, но физический смысл иной — здесь задержка отражает время полноценного прогона модели, а не быстрого суррогата). Ускорение

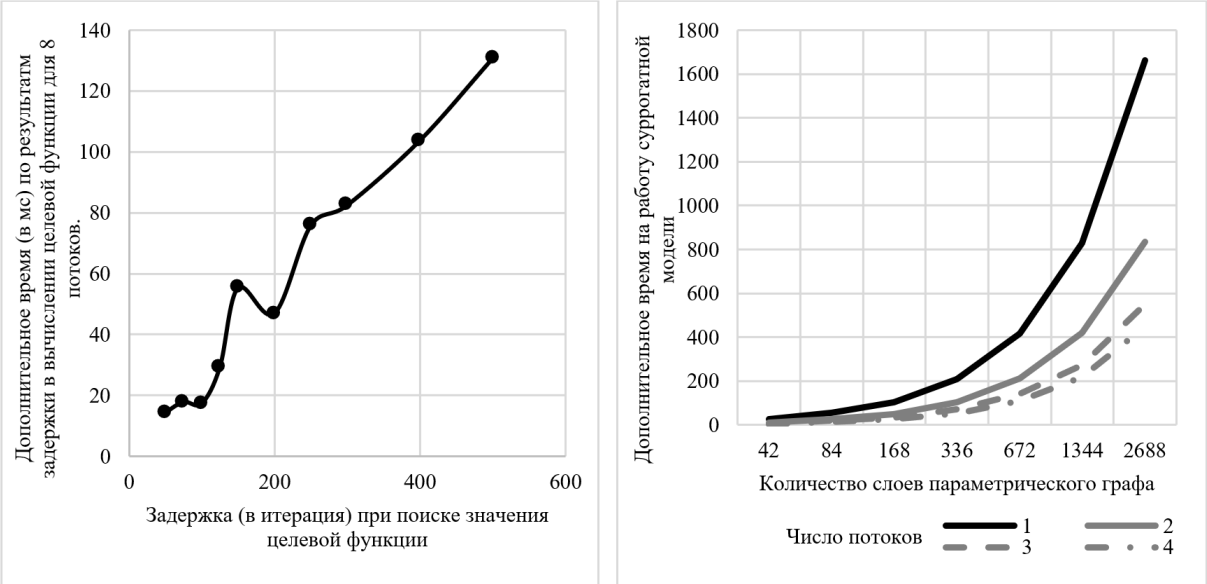


Рис. 3. Оценка эффективности работы матричного метода муравьиных колоний на Intel i5 8300H при моделировании работы суррогатной модели (справа) и имитационной модели поиска значения целевой функции (слева).

на 2 потоках относительно одного остается близким к идеальному (1.85–1.95 раза, эффективность 92–98%) для всех размерностей графа, начиная с 672 слоев. При переходе к 3 и 4 потокам ускорение продолжает расти, но уже не линейно: на 4 потоках достигается ускорение 2.9–3.3 раза (эффективность 72–82%).

При моделировании работы суррогатной модели производительность системы ограничена, прежде всего, скоростью работы самого метода муравьиных колоний. В этом режиме добавление 3-го и 4-го потока дает заметный выигрыш, поскольку основная работа хорошо параллелизуется. При моделировании работы целевой функции узким местом становятся внешние вычисления, которые требуют синхронизации доступа к хэш-таблице для занесения новых результатов. В результате эффективность 4 потоков падает до 70–75% для высоких задержек. Однако абсолютное время выполнения остается приемлемым благодаря тому, что основное время тратится на полезные вычисления модели, а не на простой. Даже в худшем случае (4 потока, задержка 500 итераций, размерность 86016 слоев) нормированное время составляет около 2.35 мс, что всего на 15–20% выше, чем на 2 потоках (2.05 мс), а по сравнению с однопоточной реализацией (около 4.2 мс) достигается ускорение 1.8 раза. Это подтверждает, что предложенная матричная формализация ACO сохраняет свою эффективность даже при интеграции с внешними моделями, требующими значительного времени на однократный вызов.

3.5. Масштабируемость по данным для гибридной MPI+GPU реализации на кластерных конфигурациях

Таблицы 5 и 6 представляют результаты оценки ускорения при использовании 2 и 4 GPU (MPI+GPU гибридная модель) по сравнению с гибридной моделью на одном GPU. Эксперименты проводились на конфигурации с несколькими ускорителями NVIDIA Tesla V100, объединенными через NVLink и InfiniBand, с синхронной моделью обновления феромонов через `MPI_Allgather` после каждой итерации. При малом числе муравьев-агентов (256–1024) ускорение на 2 GPU оказывается меньше 1 (0.55–

0.99) для большинства размерностей графа, что свидетельствует о доминировании коммуникационных накладных расходов над вычислительной работой. Передача феромонных матриц между процессами, даже с применением дельта-компрессии, требует времени, сопоставимого со временем выполнения самой итерации, когда агентов мало и каждый GPU простаивает в ожидании данных. При увеличении числа муравьев-агентов до 4096–16384 ситуация меняется кардинально: ускорение на 2 GPU достигает 1.34–1.62, а на 4 GPU — 2.16–2.60. Наиболее впечатляющие результаты получены при 65536–262144 агентах: ускорение на 2 GPU составляет 1.78–1.96, а на 4 GPU — 3.01–3.91, что приближается к идеальной параллельной эффективности. Это означает, что при достаточной вычислительной нагрузке (десятки тысяч муравьев-агентов) накладные расходы на коммуникацию становятся пренебрежимо малыми по сравнению со временем вычислений, и система демонстрирует эффективность параллелизации 89–98% для 2 GPU и 75–98% для 4 GPU.

Таблица 5. Оценка ускорения при применении 2 MPI-GPU по сравнению с одним GPU-ускорителем

Муравьи	Количество слоёв графа								
	42	168	672	2688	10752	43008	172032	688128	2752512
256	1.097	0.735	0.584	0.591	0.681	0.555	0.465	0.753	1.388
512	0.934	0.742	0.579	0.647	0.859	0.728	0.885	0.900	
1024	0.990	0.799	0.713	0.716	0.927	0.906	1.154	1.207	
4096	1.265	0.986	0.977	1.168	1.225	1.337	1.513		
16384	1.624	1.348	1.496	1.608	1.450	1.503			
65536	1.878	1.774	1.783	1.852	1.711				
262144	1.956	1.914	1.895	1.969					
1048576	1.948	1.879	1.967				Недостаточно памяти		
4194304	1.978	1.897							
16777216	1.987								

Анализ таблиц также показывает, что при переходе от 2 к 4 GPU эффективность параллелизации несколько снижается (с 95–98% до 90–95% для оптимальных конфигураций), что объясняется возросшими накладными расходами на коллективные операции `MPI_Allgather` с четырьмя участниками и увеличением доли последовательного кода в каждом процессе. Тем не менее, полученные значения ускорения (до 3.95× на 4 GPU) являются одними из лучших среди известных публикаций по параллельному ACO и демонстрируют, что предложенная гибридная MPI+GPU реализация матричного ACO для параметрических задач способна эффективно использовать кластерные ресурсы, обеспечивая почти линейное ускорение при достаточной вычислительной нагрузке.

Таблица 6. Оценка ускорения при применении 4 MPI-GPU по сравнению с одним GPU ускорителем.

Муравьи	Количество слоев графа							
	42	168	672	2688	10752	43008	172032	688128
256	0.949	0.722	0.522	0.563	0.674	0.531	0.448	0.835
512	0.828	0.720	0.563	0.637	0.830	0.762	0.981	1.176
1024	0.915	0.812	0.676	0.794	1.116	1.136	1.442	1.673
4096	1.225	1.086	1.083	1.513	1.751	2.098	2.323	
16384	2.161	1.824	2.117	2.595	2.221	2.541		
65536	3.212	3.077	3.208	3.484	3.006			
262144	3.672	3.676	3.622	3.909				
1048576	3.804	3.492	3.860	Недостаточно памяти				
4194304	3.796	3.862						
16777216	3.946							

4. Выводы и рекомендации

В настоящей работе представлены высокопроизводительные реализации матричной моди фикации метода муравьиных колоний для параметрических задач оптимизации на трех архитек турных моделях: SMP с использованием OpenMP и векторизации AVX2, SIMT на GPU с исполь зованием CUDA, а также распределенная MPP-модель с использованием MPI и гибридная MPI+GPU архитектура.

Для GPU-реализацииэкспериментально подтверждена масштабируемость по данным: при увеличении числа слоев параметрического графа от 42 до 1,38 миллиона нормированное время выполнения на Tesla V100 снижается с 2.05 до 0.10–0.12 мс (улучшение на 95%). На массовом GPU GTX 1050Ti минимальное нормированное время 0.447–0.453 мс достигается при 2688–5376 слоях. Платформа Jetson Orin показывает стабильное время 1.48–1.62 мс в диапазоне 336–43008 слоев, что в 12–15 раз медленнее V100, но подтверждает применимость метода на встраиваемых системах. Для гибридной MPI+GPU реализации, на конфигурации с 2–4 GPU Tesla V100 на од ной плате (NVLink) при достаточной вычислительной нагрузке (65536–262144 муравья-агента) ускорение на 2 GPU составляет 1.78–1.96 раза (эффективность 89–98%), на 4 GPU — 3.01–3.91 раза (эффективность 75–98%).

Среди всех исследованных CPU абсолютным лидером является AMD Ryzen 5 7500F с ис пользованием AVX2, достигающий на 2688 слоях нормированного времени 0.22×10^{-3} мс, что на 59% быстрее, чем Intel Xeon Gold 6130 (0.54×10^{-3} мс). Исследование мобильных процессоров показало, что гибридная архитектура i5-12450H при использовании всех 12 логических процес соров обеспечивает производительность, всего на 15–20% уступающую серверному Xeon Gold 6130 на размерностях 10752–21504 слоя.

Список литературы

[1] Sinitsyn I.N., Titov Y.P. Control of Set of System Parameter Values by the Ant Colony Method // Automation and Remote Control. 2023. Vol. 84, No. 8. P. 893–903. DOI: 10.1134/S0005117923080106

[2] Sudakov V.A., Titov Y.P. Matrix-Based ACO for Solving Parametric Problems Using Heterogeneous Reconfigurable Computers and SIMD Accelerators // Mathematics. 2025. Vol. 13, No. 8. P. 1284. DOI: 10.3390/math13081284

[3] Sudakov V.A., Titov Y.P. Investigation of the Parametric Graph Model in the Ant Colony Method // Mathematical Models and Computer Simulations. 2025. Vol. 17. P. 126–136. DOI: 10.1134/S2070048224700996

[4] Colorni A., Dorigo M., Maniezzo V. Distributed Optimization by Ant Colonies // Proceedings of the First European Conference on Artificial Life. Elsevier, 1991. P. 134–142.

[5] Dorigo M., Maniezzo V., Colorni A. Ant System: Optimization by a Colony of Cooperating Agents // IEEE Transactions on Systems, Man, and Cybernetics. 1996. Vol. 26, No. 1. P. 29–41. DOI: 10.1109/3477.484436

[6] Dorigo M., Gambardella L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem // IEEE Transactions on Evolutionary Computation. 1997. Vol. 1, No. 1. P. 53–66. DOI: 10.1109/4235.585892

[7] Stützle T., Hoos H.H. MAX–MIN Ant System // Future Generation Computer Systems. 2000. Vol. 16, No. 8. P. 889–914. DOI: 10.1016/S0167-739X(00)00043-1

[8] Dorigo M., Stützle T. Ant Colony Optimization. MIT Press, 2004. 321 p. DOI: 10.7551/mit-press/1290.001.0001

[9] Cordón O., Fernández de Viana I., Herrera F. Analysis of the Best-Worst Ant System and Its Variants on the TSP // Soft Computing. 2002. Vol. 9, No. 2–3. P. 177–192. DOI: 10.1007/s00500-002-0205-7

[10] Shen Y., Gao X. Adaptive Pheromone Evaporation for Dynamic Ant Colony Optimization // Swarm and Evolutionary Computation. 2024. Vol. 84. P. 101445. DOI: 10.1016/j.swevo.2023.101445

[11] Zhang L., Wang Y., Liu J. Multi-Objective Ant Colony Optimization with Elite Migration for Assembly Line Balancing // IEEE Access. 2023. Vol. 11. P. 12345–12358. DOI: 10.1109/ACCESS.2023.3234567

[12] Liao T., Stützle T., Montes de Oca M.A., Dorigo M. A Unified Ant Colony Optimization Algorithm for Continuous Optimization // European Journal of Operational Research. 2022. Vol. 296, No. 3. P. 789–804. DOI: 10.1016/j.ejor.2021.05.012

[13] ElSaid A., Wild B., El Jamiy F., Higgins J., Desell T. Using Ant Colony Optimization to Optimize Long Short-Term Memory Recurrent Neural Networks // Swarm and Evolutionary Computation. 2023. Vol. 76. P. 101215. DOI: 10.1016/j.swevo.2022.101215

[14] Hwang W., Kang D., Kim D. Brain Lateralisation Feature Extraction and Ant Colony Optimisation-Bidirectional LSTM Network Model for Emotion Recognition // IET Signal Processing. 2022. Vol. 16, No. 1. P. 45–61. DOI: 10.1049/sil2.12089

[15] Bullnheimer B., Kotsis G., Strauß C. Parallelization Strategies for the Ant System // Applied Optimization. 1998. Vol. 24. P. 87–100. DOI: 10.1007/978-1-4613-3279-4_6

[16] Randall M., Lewis A. A Parallel Implementation of Ant Colony Optimization // Journal of Parallel and Distributed Computing. 2002. Vol. 62, No. 9. P. 1421–1432. DOI: 10.1006/jpdc.2002.1854

[17] Zhou Y., He F., Hou N., Qiu Y. Parallel Ant Colony Optimization on Multi-Core SIMD CPUs // Future Generation Computer Systems. 2018. Vol. 79, No. 2. P. 473–487. DOI:

10.1016/j.future.2017.09.073

[18] Abouelfarag A.A., Aly W.M., Elbaily A.G. Performance Analysis and Tuning for Parallelization of Ant Colony Optimization by Using OpenMP // Lecture Notes in Computer Science. 2020. Vol. 12405. P. 231–242. DOI: 10.1007/978-3-030-63119-8_18

[19] Mansour I.B., Alaya I.B., Tagina M. A New Parallel Hybrid Multi-Objective Ant Colony Algorithm Based on OpenMP // Proceedings of the International Conference on Applied Computing (AC2020). 2020. P. 19–26. DOI: 10.33965/ac2020_202013003

[20] Bai H., OuYang D., Li X., He L., Yu H. MAX-MIN Ant System on GPU with CUDA // Proceedings of the Fourth International Conference on Innovative Computing, Information and Control (ICICIC'09). IEEE, 2009. P. 801–804. DOI: 10.1109/ICICIC.2009.255

[21] Skinderowicz R. The GPU-Based Parallel Ant Colony System // Journal of Parallel and Distributed Computing. 2016. Vol. 98. P. 48–60. DOI: 10.1016/j.jpdc.2016.04.014

[22] Skinderowicz R. Implementing a GPU-Based Parallel MAX-MIN Ant System // Future Generation Computer Systems. 2020. Vol. 106. P. 277–295. DOI: 10.1016/j.future.2020.01.011

[23] Huan Z.B., Fu G.T., Fa T.H., et al. High Performance Ant Colony System Based on GPU Warp Specialization with a Static-Dynamic Balanced Candidate Set Strategy // Future Generation Computer Systems. 2021. Vol. 125. P. 136–150. DOI: 10.1016/j.future.2021.06.041

[24] Zhi Z., Yuxing C., Kwok C.L., Hui L., Jinwei W. A Fast Fully Parallel Ant Colony Optimization Algorithm Based on CUDA for Solving TSP // IET Computers & Digital Techniques. 2023. Vol. 17. P. 9915769. DOI: 10.1049/2023/9915769

[25] Cecilia J.M., Llanes A., Abellan J.L., Gomez-Luna J., Chang L.W., Hwu W.M.W. High-Throughput Ant Colony Optimization on Graphics Processing Units // Journal of Parallel and Distributed Computing. 2018. Vol. 113. P. 261–274. DOI: 10.1016/j.jpdc.2017.11.002

[26] Pedemonte M., Nesmachnow S., Cancela H. A Survey on Parallel Ant Colony Optimization // Journal of Parallel and Distributed Computing. 2011. Vol. 71, No. 8. P. 1071–1089. DOI: 10.1016/j.jpdc.2011.04.001

[27] Tsutsui S. ACO on Multiple GPUs with CUDA for Faster Solution of QAPs // Lecture Notes in Computer Science. 2012. Vol. 7492. P. 180–189. DOI: 10.1007/978-3-642-32964-7_18

[28] Stojanov T., Fabian M. Asynchronous Migration Strategies for Ant Colony Optimization on Quadratic Assignment Problem // Computing and Informatics. 2021. Vol. 40, No. 3. P. 456–478. DOI: 10.31577/cai_2021_3_456

[29] Verstraete J. Metaheuristics for Heterogeneous Environments // Journal of Heuristics. 2022. Vol. 28, No. 2. P. 123–145. DOI: 10.1007/s10732-021-09478-w

[30] Titov Y. Optimization of the Structure of the Solution of Parametric Problems by Methods of Modification of Ant Colonies Taking into Account Time Factors // Mathematical Modeling and Supercomputer Technologies. MMST 2025. Communications in Computer and Information Science. – 2026. – V2815. Springer, Cham. DOI:10.1007/978-3-032-15761-4_9

[31] Титов Ю.П. Модификации матричного метода муравьиных колоний для параметрической оптимизации с применением GPU с технологией CUDA // Вычислительные методы и программирование / Numerical Methods and Programming. 2026. Т.27, № 1. с. 46-66. DOI:10.26089/NumMet.v27r104

Гетерогенный параллельный алгоритм для моделирования многоступенчатых турбомашин на основе метода нелинейных гармоник

Р.Р. Гайсин[1], А.В. Горобец[1], А.П. Дубень[1]

[1]Институт прикладной математики имени М.В. Келдыша РАН, Москва, Россия

Работа посвящена гетерогенной параллельной реализации метода нелинейных гармоник (NLH) для учета нестационарности при моделировании задач турбомашиностроения. Моделирование осуществляется с использованием схем повышенной точности на неструктурированных сетках и неявной схемы интегрирования по времени. Многоуровневый параллельный алгоритм реализован с помощью фреймворков MPI, OpenMP и OpenCL, что позволяет совместно использовать центральные и графические процессоры различных производителей на гибридных кластерных системах. В этой статье рассматриваются ключевые моменты реализации метода NLH для вычислений на графических процессорах с использованием стандарта OpenCL.

Ключевые слова: турбомашины, метод нелинейных гармоник, суперкомпьютерное моделирование, гетерогенные вычисления, вычисления на графических процессорах

1. Введение

Метод нелинейных гармоник (NLH), впервые предложенный в [1] и впоследствии применённый к реальным турбомашинам [2,3], позволяет учитывать нестационарные эффекты при моделировании компрессоров и турбин с использованием недорогих стационарных подходов. Течение моделируется на основе метода осреднённых по Рейнольдсу уравнений Навье–Стокса (RANS). Рассматривается только по одному межлопаточному каналу на венец (Рис. 1) в условиях периодичности, что крайне важно с точки зрения снижения ресурсоемкости расчета из-за обычно большого количества лопаток в каждом венце — от десятков до сотен. В частности, NLH метод используется [4,5] в коммерческом ПО инженерного анализа Cadence Fidelity FineTurbo, хорошо известный как один из наиболее эффективных инструментов моделирования для турбомашин.

Для интерфейсов ротор-статор используется метод поверхности смешения (Mixing Plane, MP) [6,7], который предполагает равномерность потока в окружном направлении на интерфейсе. Метод NLH позволяет улавливать нестационарное взаимодействие между соседними рядами лопаток за счёт передачи возмущений, связанных с частотой прохождения лопаток, через интерфейс. В рамках метода NLH решение раскладывается на составляющие окружные гармоники для учета нестационарного взаимодействия между смежными венцами роторов и статоров. Соответственно, метод NLH требует решения для каждой гармоники дополнительной системы уравнений для комплексных гармонических амплитуд. Таким образом, каждая гармоника добавляет к базовому стационарному RANS-расчёту в каждой расчетной ячейке по два набора переменных — для действительной и мнимой части. Более того, поскольку интегрирование по времени выполняется неявной схемой, на каждом шаге по времени решается система линейных уравнений с блочной разреженной матрицей Якоби. Размер блока

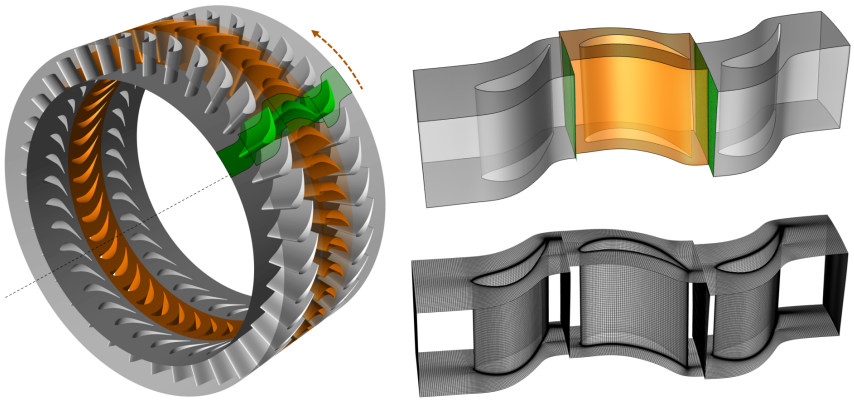


Рис. 1. Расчетная область для модельной турбины, в которой моделируется по одному межлопаточному каналу на венец в условиях окружной периодичности.

матрицы Якоби увеличивается с 5 до 10, что заметно увеличивает объём хранимых данных и вычислительные затраты. В [12] было предложено технологическое решение, направленное на кратное снижение потребления памяти.

Технология NLH была успешно реализована [8,9] с использованием стандартов MPI и OpenMP в рамках численного алгоритма повышенной точности для неструктурированных сеток в коде NOISEtte [10]. Однако этот код обладает многоуровневой гетерогенной параллелизацией [12] с использованием стандартов параллельного программирования MPI, OpenMP и OpenCL. Поэтому дальнейшее развитие метода направлено на эффективное использование графических процессоров.

2. Метод нелинейных гармоник

2.1. Математическая модель

В основе методики на основе NLH лежит решение уравнений Навье–Стокса с усреднением по Рейнольдсу (RANS) для моделирования турбулентного течения сжимаемого вязкого потока:

$$\frac{\partial \mathbf{Q}}{\partial t} + \nabla \cdot \mathcal{F}^C(\mathbf{Q}) - \nabla \cdot \mathcal{F}^D(\mathbf{Q}) = 0, \tag{1}$$

где $\mathbf{Q} = (\rho, \mathbf{m}, E)^T$ — вектор осредненных консервативных переменных, ρ — плотность, $\mathbf{m} = \rho\mathbf{u}$, где $\mathbf{u} = \{u, v, w\}$ — вектор скорости, $E = \rho(e + u^2/2)$ — полная энергия, e — удельная внутренняя энергия. Конвективные и диффузионные потоки:

$$\mathcal{F}^C(\mathbf{Q}) = \begin{pmatrix} \mathbf{m} \\ \mathbf{u} \otimes \mathbf{m} + p\mathbf{I} \\ (E + p)\mathbf{u} \end{pmatrix}, \quad \mathcal{F}^D(\mathbf{Q}) = \begin{pmatrix} 0 \\ \boldsymbol{\sigma} \\ \boldsymbol{\sigma} \cdot \mathbf{u} - \mathbf{q} \end{pmatrix}, \tag{2}$$

где $\mathbf{I}$ — единичный тензор, $\boldsymbol{\sigma}(\mathbf{u}) = \{\sigma_{jk}\}(\mathbf{u}) = \mu_{\text{eff}}(\nabla_j u_k + \nabla_k u_j - \delta_{jk}\text{div}\,\mathbf{u})$ — тензор напряжений, δ_{jk} — символ Кронекера, $\mathbf{q} = -\gamma\mu/\text{Pr}\nabla e$ — вектор тепловых потоков. $\gamma = c_P/c_V$ — коэффициент удельной теплоемкости, c_P и c_V — удельная теплоемкость при постоянном давлении и объеме, соответственно. $\text{Pr} = c_P\mu/\kappa$ — число Прандтля, κ — коэффициент теплопроводности. Предполагается, что справедливо уравнение

состояния идеального газа: $p = (\gamma - 1)\rho e$. $\mu_{\text{eff}} = \mu + \mu_t$, μ — динамическая вязкость, $\mu_t = \rho\nu_t$, ν_t — турбулентная вязкость, определяемая соответствующей моделью турбулентности.

Консервативные переменные $\mathbf{Q}$ в рамках метода NLH разделяются на усредненную по времени часть $\overline{\mathbf{Q}}(\mathbf{x})$ и периодические пульсации $\mathbf{Q}'(\mathbf{x}, t)$:

$$\mathbf{Q}(t, \mathbf{x}) = \overline{\mathbf{Q}}(\mathbf{x}) + \mathbf{Q}'(\mathbf{x}, t). \tag{3}$$

Пульсационная часть $\mathbf{Q}'(\mathbf{x}, t)$ представлена в виде суммы комплексных гармонических составляющих:

$$\mathbf{Q}'(\mathbf{x}, t) = \frac{1}{2} \sum_{k=1}^{\infty} \left[\widetilde{\mathbf{Q}}_k e^{ik\omega_0 t} + \widetilde{\mathbf{Q}}_{-k} e^{-ik\omega_0 t} \right], \tag{4}$$

где i — мнимая единица, $\widetilde{\mathbf{Q}}_k = \widetilde{\mathbf{Q}}_{a,k} + i\widetilde{\mathbf{Q}}_{b,k}$, $\widetilde{\mathbf{Q}}_{a,k} = \Re(\widetilde{\mathbf{Q}}_k)$ и $\widetilde{\mathbf{Q}}_{b,k} = \Im(\widetilde{\mathbf{Q}}_k)$ — действительная и мнимая части амплитуды для k-й гармоники, соответственно, $\widetilde{\mathbf{Q}}_{-k} = \widetilde{\mathbf{Q}}_{a,k} - i\widetilde{\mathbf{Q}}_{b,k}$ — комплексно-сопряженное для $\widetilde{\mathbf{Q}}_k$. Количество гармоник N_h определяет точность аппроксимации $\mathbf{Q}'(\mathbf{x}, t)$ в (4): чем больше гармоник, тем точнее, но с большими вычислительными затратами. Индексы гармоник k определяются частотами прохождения лопаток (BPF — Blade Passing Frequency) Ω_{BPF} соседних рядов: $k = n \cdot k_{\text{BPF}}$, $n = 1, 2, ..., N_h$. k_{BPF}, в свою очередь, определяется как $k_{\text{BPF}} = \Omega_{\text{BPF}}/\Omega_{\text{rot}}$, где Ω_{rot} — частота вращения ($k_{\text{BPF}} \in \mathbb{N}$ по определению). Таким образом, каждый домен (ротор или статор) имеет два набора гармонических индексов k, которые пропорциональны k_{BPF} из соседних доменов.

Согласно (3), система RANS (1) разлагается на систему для усредненных переменных и набор систем для амплитуд каждой гармоники. Система усредненных переменных $\overline{\mathbf{Q}}$ имеет вид:

$$\frac{\partial \overline{\mathbf{Q}}}{\partial t} + \nabla \cdot \overline{\mathcal{F}}^C(\overline{\mathbf{Q}}, \mathbf{Q}') - \nabla \cdot \overline{\mathcal{F}}^D(\overline{\mathbf{Q}}, \mathbf{Q}') = 0, \tag{5}$$

$\overline{\mathcal{F}}^C(\overline{\mathbf{Q}}, \mathbf{Q}') = \mathcal{F}^C(\overline{\mathbf{Q}}) + \mathcal{F}^C_{\text{NLH}}(\mathbf{Q}')$, $\overline{\mathcal{F}}^D(\overline{\mathbf{Q}}, \mathbf{Q}') = \mathcal{F}^D(\overline{\mathbf{Q}}) + \mathcal{F}^D_{\text{NLH}}(\mathbf{Q}')$. Нелинейные детерминированные вклады напряжений в амплитуды гармоник имеют вид:

$$\mathcal{F}^C_{\text{NLH}}(\mathbf{Q}') = (0, \overline{\mathbf{m}' \otimes \mathbf{u}'}, \overline{(E + p)'\mathbf{u}'})^T, \quad \mathcal{F}^D_{\text{NLH}}(\mathbf{Q}') = (0, \overline{\boldsymbol{\sigma}' \cdot \mathbf{u}'})^T,$$

где $\boldsymbol{\sigma}' = \boldsymbol{\sigma}(\mathbf{u}')$. Система (5) отличается от системы (1) наличием нелинейных членов $\mathcal{F}^C_{\text{NLH}}$ и $\mathcal{F}^D_{\text{NLH}}$, которые определяются $\mathbf{Q}'$.

Для краткости, индекс k гармонической амплитуды для частоты ω_k впредь будет опускаться при упоминании векторов гармонических амплитуд $\widetilde{\mathbf{Q}}_k$. Таким образом, система для гармонического вектора амплитуды $\widetilde{\mathbf{Q}}$ имеет вид:

$$\frac{\partial \widetilde{\mathbf{Q}}}{\partial t} + \nabla \cdot \widetilde{\mathcal{F}}^C(\widetilde{\mathbf{Q}}, \overline{\mathbf{Q}}) + i\omega\widetilde{\mathbf{Q}} - \nabla \cdot \widetilde{\mathcal{F}}^D(\widetilde{\mathbf{Q}}, \overline{\mathbf{Q}}) = 0, \tag{6}$$

$$\widetilde{\mathcal{F}}^C = \begin{pmatrix} \widetilde{\mathbf{m}} \\ \widetilde{\mathbf{u}} \otimes \overline{\mathbf{m}} + \overline{\mathbf{u}} \otimes \widetilde{\mathbf{m}} + \widetilde{p}\mathbf{I} \\ (\widetilde{E} + \widetilde{p})\overline{\mathbf{u}} + (\overline{E} + \overline{p})\widetilde{\mathbf{u}} \end{pmatrix}, \quad \widetilde{\mathcal{F}}^D = \begin{pmatrix} 0 \\ \widetilde{\boldsymbol{\sigma}} \\ \widetilde{\boldsymbol{\sigma}} \cdot \overline{\mathbf{u}} + \overline{\boldsymbol{\sigma}} \cdot \widetilde{\mathbf{u}} - \widetilde{\mathbf{q}} \end{pmatrix}, \tag{7}$$

где $\widetilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}(\widetilde{\mathbf{u}})$, $\widetilde{\mathbf{q}} = -\gamma\mu/\mathrm{Pr}\nabla\widetilde{e}$. $\mu_{\text{eff}} = \mu + \mu_t$, μ_t определяется моделью турбулентности, замыкающей уравнения для усредненных переменных.

Линейные системы уравнений для амплитуд гармоник (6)–(7) независимы друг от друга. Уравнения для усредненных переменных и гармонических амплитуд замыкаются несколькими соотношениями (см. подробнее [8]), в том числе следующими: $\overline{fg} = \bar{f}\bar{g} + \overline{f'g'}$; $(fg)' = f'\bar{g} + \bar{f}g'$; $\overline{f'g'} = 0.5\sum_{k=1}^{N_h}[\Re(\tilde{f}_k)\Re(\tilde{g}_k) + \Im(\tilde{f}_k)\Im(\tilde{g}_k)]$.

2.2. Дискретизация

Конвективные члены для гармонических амплитуд $\nabla\cdot\widetilde{\mathcal{F}}^C$ в (6) дискретизируются в пространстве с повышенной точностью на неструктурированной сетке с помощью реконструкции по линии сеточных ребер схемами EBR [14] для гладких решений и схемами EBR-TVD или EBR-WENO [15] для разрывных решений. Для аппроксимации вязких членов $\nabla\cdot\widetilde{\mathcal{F}}^D$ в (6) используется метод усредненного разбиения элементов (AES — Average Element Splitting) [16].

Поскольку усредненные переменные и амплитуды гармоник не зависят от времени, для получения стационарного решения используется псевдовременной итерационный процесс. Для этого используется неявная формула обратного дифференцирования (BDF) с квази-ньютоновской линеаризацией. Как уже упоминалось выше, система (6) отличается от (1) наличием членов $\mathcal{F}^C_{\text{NLH}}$ и $\mathcal{F}^D_{\text{NLH}}$. Мы рассматриваем их как исходные члены, не зависящие от $\mathbf{Q}$, решается по той же численной схеме, что и для обычного RANS без учета нестационарных возмущений. Кроме того, он позволяет выполнять псевдовременное продвижение для системы усредненных переменных (5) и гармонических амплитуд (6) по отдельности. На каждом временном шаге сначала вычисляется подшаг для усредненных переменных с замораживанием гармонических амплитуд, затем выполняется подшаг для гармонических амплитуд с обновленными усредненными переменными.

Полудискретная аппроксимация (6) для гармонической амплитуды может быть записана в виде:

$$\frac{d\tilde{\mathbf{Q}}}{dt} = -\tilde{\mathbf{F}}(\tilde{\mathbf{Q}}, \overline{\mathbf{Q}}, \omega) = 0,$$

где $\tilde{\mathbf{F}} = -\nabla\cdot\widetilde{\mathcal{F}}^C(\tilde{\mathbf{Q}}, \overline{\mathbf{Q}}) - i\omega\tilde{\mathbf{Q}} + \nabla\cdot\widetilde{\mathcal{F}}^D(\tilde{\mathbf{Q}}, \overline{\mathbf{Q}})$. Неявная схема записывается следующим образом:

$$\left(\frac{1}{\tau^n} + \tilde{\mathbf{J}}\right)(\tilde{\mathbf{Q}}^{n+1} - \tilde{\mathbf{Q}}^n) = \tilde{\mathbf{F}}(\tilde{\mathbf{Q}}^n, \overline{\mathbf{Q}}^n, \omega), \tag{8}$$

где τ^n — величина шага по времени на n-м псевдо-шаге, $\tilde{\mathbf{J}} \approx d\tilde{\mathbf{F}}/d\tilde{\mathbf{Q}}$ — приближенный якобиан по потоку. Подчеркнем, что матрица Якоби $\tilde{\mathbf{J}}$ как для конвективного $\widetilde{\mathcal{F}}^C$, так и для диффузионного $\widetilde{\mathcal{F}}^D$ потоков зависит только от усредненных переменных $\overline{\mathbf{Q}}$, в то время как вклад от исходного члена $i\omega\tilde{\mathbf{Q}}$ зависит только от ω.

Граничные условия для гармонических амплитуд, входа, выхода и твердых поверхностей реализуются так же, как и для осредненных переменных. Для интерфейса ротор-статор используется метод поверхности смешения [7] со слабо отражающими граничными условиями. Граничные условия для гармонических амплитуд на границе ротор-статор реализуются аналогично [2] на основе MP-функционала. Для секторной периодичности на границе раздела плоскостей смешения в методе NLH используются обобщенные периодические условия со сдвигом фаз для замыкания амплитуд гармоник с различным числом лопаток в соседних рядах, как в [19]. Более подробную информацию о численном методе можно найти в [8].

Для решения линейной системы с матрицей Якоби применяется параллельный предобусловленный решатель BiCGStab [17]. Для блочных разреженных матриц используются параллельные предобуславливатели [18] на основе метода Гаусса-Зейделя.

2.3. Адаптация линейного решателя

Каждая система уравнений для гармонических амплитуд включает в себя связанные между собой действительную и мнимую части. Таким образом, размер блоков разреженных блочных матриц для гармонических амплитуд составляет 10×10 (5 переменных для действительной и 5 для мнимой части). В коде NOISEtte все гармоники решаются последовательно, поскольку матрицы для каждой гармоники имеют одинаковый портрет, но разные диагональные элементы.

Код NOISEtte имеет собственный гетерогенный параллельный линейный решатель, который использует итерационный метод BiCGStab [17] с несколькими предобуславливателями, основанными на методе Гаусса-Зейделя [18]. Он поддерживает многоуровневое распараллеливание MPI+OpenMP+OpenCL для работы на множественных CPU и GPU. Решатель представляет собой мультисистемный решатель для систем с блочными разреженными матрицами, имеющими один и тот же портрет. Матрицы хранятся в блочном формате CSR (Compressed Sparse Row), плотные блоки коэффициентов размещаются в памяти линейно в порядке следования строк. Очевидно, что совместное использование одного портрета несколькими матрицами позволяет экономить память для хранения портретов. Кроме того, групповое решение для нескольких систем одновременно позволяет снизить накладные расходы на сетевые задержки в распределенном параллельном режиме. Сообщения MPI в общем итерационном процессе сгруппированы, включая операции обновления гало в разреженных матрично-векторных произведениях (SpMV) и редукционные обмены в скалярных произведениях, что позволяет сократить количество сообщений в несколько раз. Как правило, в базовом стационарном RANS подходе имеется одна система с 5×5 блоками для 5 основных переменных (плотность, 3 компоненты вектора скорости, давление) и от 1 до 4 дополнительных систем для переменных турбулентной модели.

Базовую OpenCL версию этого решателя можно было бы напрямую использовать и для метода NLH, но с размером блока матрицы 10×10, что очень расточительно, особенно по отношению к памяти графического процессора. Поэтому OpenCL версия решателя была модернизирована с помощью специального формата хранения матриц и специализированными кернел-функциями матрично-векторного произведения. Предобуславливание также было существенно переработано для нового формата матриц, включая функцию обращения диагональных блоков матрицы и хранение инвертированных диагональных блоков [12].

Портрет матрицы соответствует смежности узлов через ребра сетки. В матрицах гармоник каждый блок матрицы 10×10 соответствует 5 действительным и 5 мнимым частям комплексных переменных в узлах. Если переменные в блоках упорядочены соответствующим образом, сначала действительные части, затем мнимые, то диагональные и внедиагональные 10×10 блоки имеют следующую структуру, соответственно:

$$\tilde{\mathbf{A}}_{ii} = \begin{pmatrix} \mathbf{A}_{ii} & -\omega v_i \mathcal{I} \\ \omega v_i \mathcal{I} & \mathbf{A}_{ii} \end{pmatrix}, \quad \tilde{\mathbf{A}}_{ij} = \begin{pmatrix} \mathbf{A}_{ij} & 0 \\ 0 & \mathbf{A}_{ij} \end{pmatrix}, \quad i \neq j, \tag{9}$$

где $\mathbf{A}_{ii}$ — плотные 5×5 блоки, ω — частота, которой соответствуют амплитуды

этих гармоник (ω_k, в действительности), v_i — объем i-го контрольного объема, и $\mathcal{I}$ — матрица тождества 5×5.

Такая специфическая структура матрицы позволяет легко сократить потребление памяти для хранения матрицы примерно в 4 раза. Вместо 100 значений хранится 26: 25 значений из 5×5 блока $\mathbf{A}_{ii}$ и еще одно значение ωv_i.

Кроме того, матрицы для гармоник зависят только от усредненных переменных, за исключением диагональных коэффициентов, которые содержат вклады, определяемые исходными членами $i\omega\mathbf{Q}$. Таким образом, после заполнения матрицы Якоби для усредненных переменных она может быть повторно использована для гармоник с обновлением только диагональных элементов диагональных блоков. Для реализации этой стратегии экономии памяти, помимо незначительных изменений в формате матрицы, все необходимые операции, связанные с произведениями с блоками матрицы, должны быть предусмотрены для этого специализированного блочного представления. В случае с решателем BiCGStab была обновлена операция SpMV, предобуславливатель также потребовал изменений.

Также потребовалась модификация кернел-функций предобуславливателя, который основан на блочном методе Якоби или блочном методе Гаусса-Зейделя. Для работы предобуславливателя требуются инвертированные диагональные блоки. Инвертированные диагональные блоки могут не соответствовать блочному формату исходных блоков, поэтому требуется хранение полных 10×10 блоков (и примерно в 8 раз больше арифметических операций для инверсии по сравнению с 5×5 блоками). Также матрицы для гармоник можно рассматривать как блочные матрицы из 5×5 блоков с портретом в 4 раза больше. В этом случае только один 5×5 диагональный блок, $\mathbf{A}_{ii}$, необходимо инвертировать для каждого 10×10 блока $\tilde{\mathbf{A}}_{ii}$. Это требует меньших вычислительных усилий для обращения диагональных блоков и в 4 раза меньше памяти для их хранения. С другой стороны, блочный метод Гаусса-Зейделя потребует двойного прохода для каждой строки блока 10×10 (поскольку он рассматривается как два ряда блоков по 5×5), что достаточно неэффективно. Поэтому был выбран вариант с частичным обращением блоков: инвертированные блоки 10×10 представляются таким образом, чтобы сохранить только их диагональные подблоки 5×5, но при этом мы продолжаем работать со строками блоков 10×10 в методах Якоби и Гаусса-Зейделя, соответственно:

$$\tilde{\mathbf{A}}_{ii}^{-1} = \begin{pmatrix} \mathbf{A}_{ii}^{-1} & 0 \\ 0 & \mathbf{A}_{ii}^{-1} \end{pmatrix}, \quad \mathbf{\Omega} = \begin{pmatrix} 0 & -\omega v_i \mathcal{I} \\ \omega v_i \mathcal{I} & 0 \end{pmatrix}.$$

Использование этой приближенной инверсии, дополненной небольшой модификацией за счет включения дополнительного члена $\mathbf{\Omega}\tilde{\mathbf{x}}^m$, позволяет сэкономить много вычислительных усилий и памяти на работе с инвертированными диагональными блоками [12].

3. Особенности параллельной реализации на GPU

Общая схема шага неявного интегрирования по времени представлена на рисунке 2.

Алгоритм был полностью адаптирован к потоковой обработке. Все соответствующие вычислительные функции перенесены в OpenCL. Исходный код OpenCL хранится в отдельных входных файлах. Во время выполнения программы на CPU эти файлы загружаются и передаются компилятору OpenCL выбранной платформы. Таким

Рис. 2. Схема одного шага неявного интегрирования по времени.

образом, большинство входных параметров уже известны на этапе компиляции.

Из-за ограниченности объема локальной памяти на потоковых мультипроцессорах GPU, этап вычисления конвективных и диффузионных потоков был модифицирован для работы в цикле по гармоникам. С другой стороны, в сравнении с CPU-версией в коде для GPU пульсационная часть потоковых переменных по частотам была объединена в один массив, что позволило сократить число вызовов некоторых функций обработки этих переменных на GPU.

Решатель запускается в цикле для каждой гармоники, что позволяет использовать одну и ту же матрицу, модифицируя только внедиагональные элементы $-\omega v_i \mathcal{I}$ и $\omega v_i \mathcal{I}$ внутри диагональных блоков $\mathbf{A}_{ii}$ (9) для каждой частоты, также можно использовать один и тот же буфер для хранения решения. Все это позволяет сократить кратно количеству гармоник потребление ценной оперативной памяти на GPU на хранение матрицы якобиана и решения.

В таблице 1 представлено относительное время работы основных этапов шага неявного интегрирования по времени на CPU и GPU в процентах от общего времени шага, которое на тестовом примере Aachen Turbine с тремя гармониками (сетка 3 млн узлов) составило 37.8 секунд на CPU (Intel Xeon E5-2697 v2, 12 ядер, 51 ГБ/с) и 2.48 секунд на GPU (NVIDIA RTX A5000, 768 ГБ/с).

Таблица 1. Относительное время выполнения этапов шага неявного интегрирования, %

Название этапа	**CPU-версия**	**GPU-версия**
Вычисление усредненных переменных	27	19
Вычисление потоков	25	15.6
Конвективных и диффузионных потоков	24	10.6
Mixing Plane	0.3	3.4
Решатель BiCGSTAB	46	64
Предобуславливатель	36.7	53.4
SpMV	6.4	2.4

3.1. Матрично-векторное умножение (SpMV)

Для использования на GPU, на основе описанного выше формата хранения разреженной NLH-матрицы, были написаны две версии функций матрично-векторного произведения. Первый вариант представляет собой «наивный» перенос CPU версии, где один блок результирующего вектора из 10 элементов обрабатывается одной GPU-нитью.

Во втором варианте каждый элемент результирующего вектора обрабатывается шестью GPU-нитями — пять нитей на каждую строку блока $\mathbf{A}_{ii}$ и одна нить на внедиагональный элемент. Таким образом, каждый блок результирующего вектора, состоящий из 10 переменных, обрабатывается шестьюдесятью GPU-нитями.

В таблице 2 приведено сравнение времени выполнения двух реализаций матрично-векторного умножения для разреженных NLH-матриц; матрицы представляют собой матрицы смежности для тестовых сеток из 64000, 512000 и 1728000 узлов.

Таблица 2. Время выполнения в секундах различных версий операции SpMV

Версия операции SpMV	**64 000 узлов**	**512 000 узлов**	**1 728 000 узлов**
Одна GPU-нить на один блок	0.0006	0.0057	0.0196
60 GPU-нитей на блок	0.0002	0.0015	0.0050

Представленные результаты показывают ускорение SpMV примерно в 3.5 раза по сравнению с «наивной» реализацией для сеток объёмом от 64 тысяч до 1.7 миллиона узлов.

4. Анализ производительности

4.1. Описание тестовых задач

Для сравнения производительности CPU и GPU версий, а также для оценки параллельного ускорения используются три модельных задачи. Моделирование для

всех тестовых задач проводилось с использованием модели турбулентности Mentor SST [22]. Гексаэдральные сетки для тестовых примеров были построены с помощью программы TurboR&D.Mesher [23].

Первая задача — это изолированное рабочее колесо с 22 лопатками, NASA Rotor-67 [20]. Хотя статор отсутствует, расчетная область специально разделена на три части, как показано на рисунке 3. Таким образом, статические области проводят возмущения, создаваемые вращающимся рабочим колесом, которые учитываются методом NLH. Полученные ранее напорные характеристики Rotor-67 представлены в [9]. Здесь при проведении эксплуатационных испытаний рассматривается режим максимальной эффективности ротора. Расчетная сетка, состоящая из гексаэдров, содержит 300 тысяч узлов (однопластной сектор с периодическими граничными условиями). Во всех трех подобластях применяется одинаковое количество секторов периодичности.

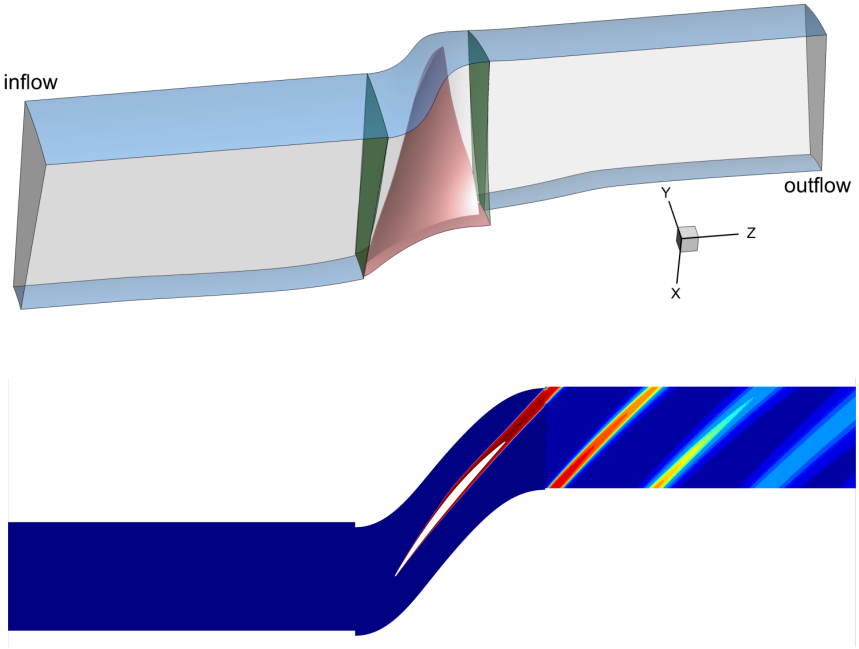
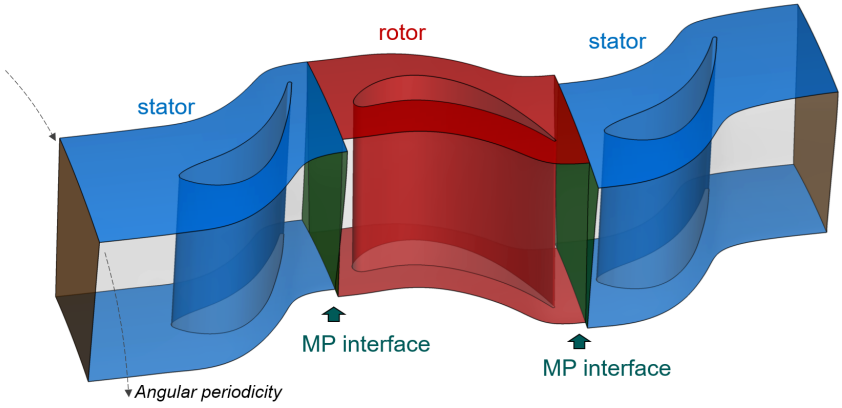


Рис. 3. Модельный NASA Rotor-67: расчетная область (сверху) и численное решение — поле течения в сечении по середине канала (поле энтропии).

Вторая задача, Aachen Turbine [13] представляет собой одноступенчатую осевую газовую турбину, состоящую из направляющего аппарата (статора), рабочего колеса (ротора) и второго направляющего аппарата, идентичного первому (рисунок 4). Сетка содержит 3 миллиона узлов.



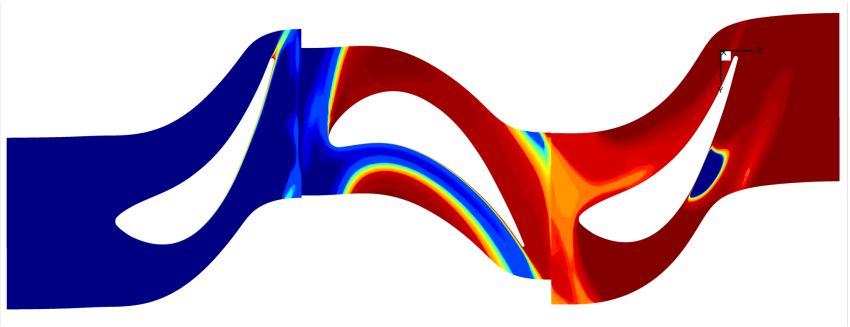


Рис. 4. Модельная турбина Aachen Turbine: расчетная область (сверху) и численное решение — поле течения в сечении по середине канала (поле энтропии).

Третья задача, более трудоемкая с точки зрения вычислений, представляет собой часть модельного осевого компрессора газотурбинного двигателя с масштабированным числом лопаток (см. рисунок 5). Более подробную информацию можно найти в [21]. Рассматриваемая конфигурация состоит из входной направляющей лопатки и четырех ступеней ротора и статора, что в общей сложности представляет собой 8 интерфейсов ротор-статор. Сетка для этой конфигурации содержит 11.9 миллионов узлов. Как и для Rotor-67, характеристики напора можно найти в [9]. Здесь рассматривается режим наивысшего КПД для тестов производительности.

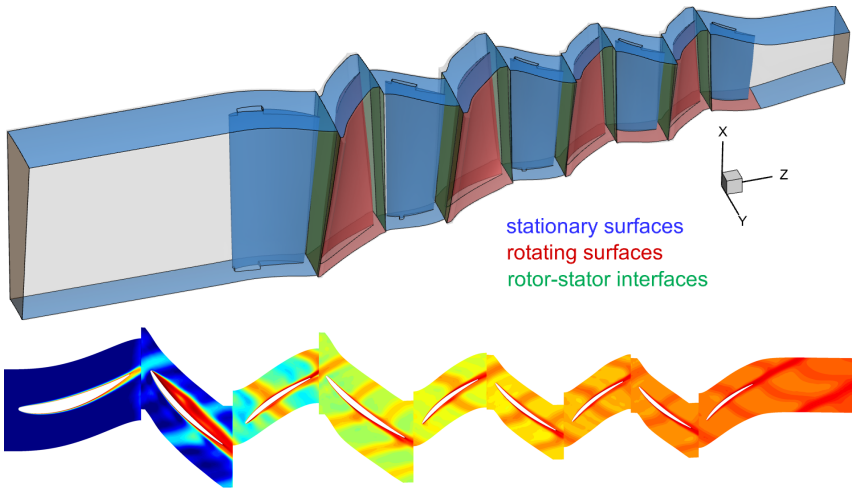
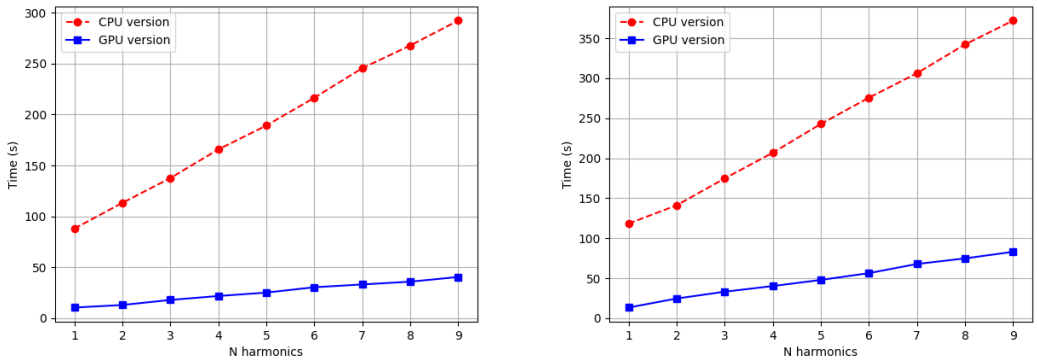


Рис. 5. Осевой многоступенчатый компрессор: расчетная область (сверху) и численное решение — поле течения в сечении по середине канала (поле энтропии)

4.2. Сравнение производительности CPU и GPU версий

Для первой тестовой задачи производительность тестировалась на 12 OpenMP-нитях на 12-ядерном процессоре Intel Xeon CPU E5-2697 v2 и GPU NVIDIA RTX A5000 (24 GB, 768 GB/s, 64 SM). Замеры времени проводились каждые 100 шагов неявного интегрирования по времени. Вторая задача тестировалась на двух MPI-процессах каждый на 16 OpenMP-нитях запущенных на 16-ядерных процессорах Intel Xeon Gold 5218 CPU (128 GB/s) и двух GPU NVIDIA RTX A5000. Графики сравнения производительности приведены на рисунке 6, время замерялось каждые 10 шагов. GPU-версия кода в таких условиях работает примерно в 5 раз быстрее.

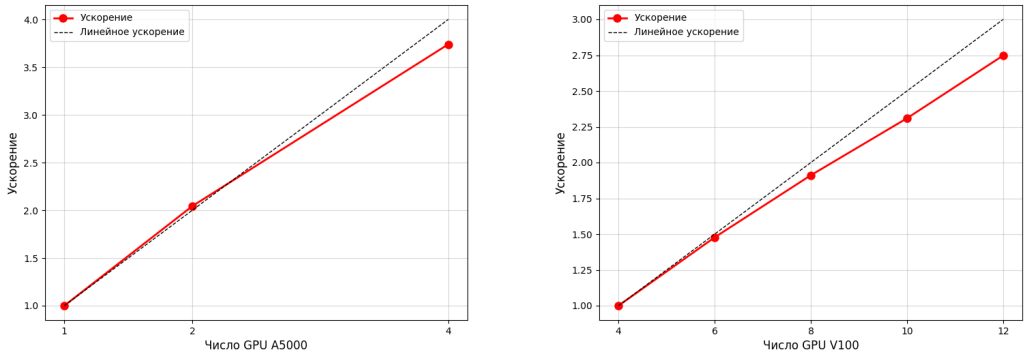


Rotor67 Aachen Turbine

Рис. 6. Сравнение производительности GPU и CPU [11] реализаций

4.3. Параллельное ускорение

Параллельное ускорение измерялось путем запуска тестовой задачи Aachen Turbine на одном, двух и четырех GPU NVIDIA A5000 (левый график на рисунке 2). Также тестирование было проведено на задаче для многоступенчатого компрессора на вычислительном кластере, имеющем по 4 GPU NVIDIA V100 (40 GB, 900 GB/s, 80 SM) на узел, на одном (4 GPU), двух (для 6 и 8 GPU) и трех узлах (для 10 и 12 GPU) (правый график на рисунке 2).



Aachen Turbine Многоступенчатый компрессор

Рис. 7. Параллельное ускорение на одном узле (слева) и на вычислительном кластере

Заключение

В настоящей работе представлена гетерогенная параллельная реализация метода нелинейных гармоник (NLH) в рамках неявного по времени подхода для моделирования нестационарных течений в турбомашинах. Основное внимание уделено снижению потребления памяти и повышению производительности вычислений на графических процессорах (GPU) с использованием фреймворка OpenCL.

Алгоритм метода NLH был полностью адаптирован к потоковой обработке. Все соответствующие вычислительные функции перенесены в OpenCL.

Предложена оптимизированная реализация матрично-векторного умножения (SpMV) на GPU. Разработана версия ядра OpenCL, в которой один блок результирующего вектора обрабатывается 60 GPU-нитями (5 нитей на каждую из 10 переменных, плюс нить на внедиагональный элемент). Представленные результаты показывают ускорение SpMV в 3.5–3.9 раза по сравнению с «наивной» реализацией для сеток объёмом от 64 тысяч до 1.7 миллиона узлов.

Подтверждена эффективность на тестовых задачах. Проведённое сравнение производительности на задачах NASA Rotor-67 и Aachen Turbine демонстрирует существенное превосходство GPU-реализации (NVIDIA RTX A5000) над многоядерной CPU-реализацией (Intel Xeon). Полученное ускорение делает метод NLH практически применимым для промышленных расчётов на гибридных суперкомпьютерных системах.

Представленное в таблице 1 распределение времени по этапам расчёта показывает, что основная доля времени на GPU приходится на решатель BiCGSTAB (64%) и предобуславливатель (53.4%), тогда как на CPU эти этапы занимают 46% и 36.7% соответственно. Это указывает на то, что дальнейшая оптимизация должна быть направлена прежде всего на эти компоненты. В то же время этап вычисления потоков на GPU существенно ускорен (15.6% против 25% на CPU), что подтверждает эффективность переноса этой части алгоритма.

Список литературы

[1] He, L., Ning, W.: Efficient approach for analysis of unsteady viscous flows in turbomachines. AIAA J., 36(11), pp. 2005–2012 (1998). https://doi.org/10.2514/2.328

[2] Chen, T., Vasanthakumar, P., He, L.: Analysis of unsteady blade row interaction using nonlinear harmonic approach. J. Propuls. Power, 17(3), pp. 651–658 (2001). https://doi.org/10.2514/2.5792

[3] He, L., Chen, T., Wells, R.G., Li, Y.S., Ning, W.: Analysis of Rotor-Rotor and Stator-Stator Interferences in Multi-Stage Turbomachines. J. Turbomach. Trans. ASME, 124(4), pp. 564–571 (2002). https://doi.org/10.1115/1.1508382

[4] Vilmin, S., Lorrain, E., Hirsch, C., Hirsch, M., Swoboda, M.: Unsteady flow modeling across the rotor/stator interface using the non-linear harmonic method. ASME Turbo Expo 2006 GT2006–90210, pp. 1227–1237 (2006). https://doi.org/10.1115/GT2006–90210

[5] Vilmin, S., Lorrain, E., Tartinville, B., Capron, A., Hirsch, C.: The Nonlinear Harmonic Method: From single stage to multi-row effects. Int. J. Comput. Fluid Dyn., 27(2), pp. 88–99 (2013). https://doi.org/10.1080/10618562.2012.752074

[6] Burgos, M.A., Contreras, J., Corral, R.: Efficient Edge-Based Rotor/Stator Interaction Method. AIAA J., 49(1), pp. 19–31 (2011). https://doi.org/10.2514/1.44512

[7] Duben, A., Gorobets, A., Soukov, S., Marakueva, O., Shuvaev, N., Zagitov, R.: Supercomputer Simulations of Turbomachinery Problems with Higher Accuracy on Unstructured Meshes. In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds.) RuSCDays 2022. LNCS, vol. 13708, pp. 356–367. Springer (2022). https://doi.org/10.1007/978-3-031-22941-1

[8] Duben, A.P., Zagitov, R.A., Shuvaev, N.V.: Nonlinear harmonics method for supercomputer simulations of fluid dynamics in turbomachines with higher accuracy on unstructured meshes. Lobachevskii J. Math., 45(7), pp. 3007–3016 (2024). https://doi.org/10.1134/S1995080224603916

[9] Duben, A.P., Zagitov, R.A., Shuvaev, N.V., Marakueva, O.V.: Towards an Adaptation

of the Nonlinear Harmonics Method Realized in an Unstructured Flow Solver for Simulation of Turbomachinery Problems on Supercomputers. In: Voevodin, V., Antonov, A., and Nikitenko, D. (eds.) Supercomputing. RuSCDays 2024. LNCS, vol. 15406, pp. 253–266. Springer, Cham. (2025). https://doi.org/10.1007/978-3-031-78459-0_19

[10] Abalakin, I.V., Bakhvalov, P.A., Bobkov, V.G., Duben, A.P., Gorobets, A.V., Kozubskaya, T.K., Rodionov, P.V., Zhdanova, N.S.: NOISEtte CFD&CAA Supercomputer Code for Research and Applications. Supercomputing Frontiers and Innovations, 11(2), 78–101 (2024). https://doi.org/10.14529/jsfi2402016

[11] Gorobets, A., Bakhvalov, P.: Heterogeneous CPU+GPU parallelization for high-accuracy scale-resolving simulations of compressible turbulent flows on hybrid supercomputers. Computer Physics Communications. Vol. 271, 108231 (2022). https://doi.org/10.1016/
j.cpc.2021.108231

[12] Duben, A., Gorobets, A.: Application-Specific Parallel Linear Solver for Nonlinear Harmonics Method with Implicit Time Integration. Supercomputing Frontiers and Innovations, 12(1), 60–72 (2025). https://doi.org/10.14529/jsfi250105

[13] Gallus, H.E., et al.: Endwall and Unsteady Flow Phenomena in an Axial Turbine Stage. ASME J. Turbomach., 117(4), pp. 562–570 (1995). https://doi.org/10.1115/1.2836568

[14] Bakhvalov, P.A., Abalakin, I.V., Kozubskaya, T.K.: Edge-based reconstruction schemes for unstructured tetrahedral meshes. Int. J. Numer. Methods Fluids, 81(6), pp. 331–356 (2016). https://doi.org/10.1002/fld.4187

[15] Bakhvalov, P.A., Kozubskaya, T.K.: EBR-WENO scheme for solving gas dynamics problems with discontinuities on unstructured meshes. Comput. Fluids, 157, pp. 312–324 (2017). https://doi.org/10.1016/j.compfluid.2017.09.004

[16] Bakhvalov, P., Surnachev, M.: Method of averaged element splittings for diffusion terms discretization in vertex-centered framework. J. Comput. Phys., 450, 110819 (2022). https://doi.org/10.1016/j.jcp.2021.110819

[17] Van der Vorst, H.: Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput., 13, pp. 631–644 (1992). https://doi.org/10.1137/0913035

[18] Magomedov, A.R., Gorobets, A.V.: Heterogeneous Implementation of Preconditioners Based on Gauss-Seidel Method for Sparse Block Matrices. Computational Mathematics and Modeling, 33, pp. 438–442 (2022). https://doi.org/10.1007/s10598-023-09585-2

[19] Gerolymos, G.A., Michon, G.J., Neubauer, J.: Analysis and Application of Chorochronic Periodicity in Turbomachinery Rotor/Stator Interaction Computations. J. Propuls. Power, 18(6), pp. 1139–1152 (2002). https://doi.org/10.2514/2.6065

[20] Strazisar, A.J., Wood, J.R., Hathaway, M.D., Suder, K.L.: Laser Anemometer Measurements in a Transonic Axial-Flow Fan Rotor. NASA TP-2879, 1989.

[21] Voroshin, D.V., Marakueva, O.V., Muraveiko, A.S.: Modeling Unsteady Phenomena in an Axial Compressor. Math. Models Comput. Simul., 12, pp. 413–421 (2020). https://doi.org/10.1134/S2070048220030187

[22] Menter, F.R., Kuntz, M., Langtry, R.: Ten Years of Industrial Experience with the SST Turbulence Model. Proceedings of the 4th International Symposium on Turbulence, Heat and Mass Transfer, Begell House Inc., West Redding, pp. 625–632 (2003). https://doi.org/10.2514/3.12149

[23] Zagitov, R.A., Salnikov, S.D., Shuvaev, N.V.: Automatic Block-Structured Grid Generation in Turbo Machine Blade Passages by TurboR&D.Mesher Software. Math. Models Comput. Simul., 16, pp. 112–122 (2024). https://doi.org/10.1134/S2070048224010125

Дальнейшее ускорение параллельной версии модели ПЛАВ*

М.А. Толстых[1,2], Г.С. Гойман[1,2,3], Е.О. Бирючева[1,2], Р.Ю. Фадеев[1,2,3]

[1]Институт вычислительной математики им. Г.И. Марчука РАН
[2]Гидрометцентр России
[3]Московский физико-технический институт

Представлены работы по ускорению расчетов прогнозов глобальной модели атмосферы ПЛАВ. Эти работы включают работы по дальнейшей реализации вычислений с одинарной точностью в блоке динамики модели и оптимизацию обменов данными между процессорами. В результате время расчета среднесрочного прогноза заблаговременностью 10 суток моделью ПЛАВ10 с разрешением около 10 км уменьшилось на 7 % или на 10 минут при использовании 2916 процессорных ядер вычислительной системы Cray XC40.

Ключевые слова: численный прогноз погоды, гидродинамическая модель атмосферы, вычисления с уменьшенной точностью

1. Введение

Повышение точности прогноза погоды остается актуальной задачей. Основным методом прогноза погоды с заблаговременностью от нескольких часов до нескольких месяцев сегодня являются глобальные гидродинамические модели общей циркуляции атмосферы, требующие больших вычислительных ресурсов. Это связано с требуемыми сроками готовности прогнозов: обычно время расчета среднесрочного прогноза на сутки должно быть не более 10 минут. Поэтому вычислительные системы национальных метеослужб традиционно представлены в списке самых мощных суперкомпьютеров Top500 [12].

Типичная модель для среднесрочного прогноза имеет шаг сетки по горизонтали от 7 до 15 км, размерность расчетной области составляет при этом 10^8-10^9. Модели для долгосрочного прогноза обычно имеют более грубое разрешение, около 30-70 км, размерность вычислительной области для таких моделей порядка 10^7. Однако для долгосрочного прогноза требуется расчет калибровочного ансамбля по начальным данным того же дня, что и начальные данные прогноза, но за период в двадцать предшествующих лет. Таким образом, для получения вероятностного прогноза осредненных аномалий погоды необходимо рассчитать около 700 отдельных прогнозов. Это также означает требования к вычислительной эффективности расчетов, в этом случае для конфигурации прогностической модели с умеренным разрешением.

В последние годы интенсивно развиваются модели прогноза погоды, основанные на методах машинного обучения [5], однако такие модели пока не в состоянии обеспечить полную номенклатуру прогностической информации с требуемым пространственным и временным разрешением, а также успешность прогноза опасных метеорологических явлений, например, тропических циклонов. Наряду с моделями на основе искусственного интеллекта, требующими после обучения очень незначительных вычислительных

*Работы по ускорению расчетов долгосрочных прогнозов модели ПЛАВ выполнены при поддержке про-екта РНФ 26-17-00314.

ресурсов, успешно развиваются и гибридные модели, в которых решение гидродинамических моделей на каждом шаге по времени притягивается к крупномасштабному решению ИИ-моделей [6].

В рамках проекта WP-MIP Всемирной метеорологической организации в настоящее время ведется систематическое сравнение качества прогнозов по моделям различного типа, в том числе, гибридных; предварительные результаты представлены в [6].

Современные высокопроизводительные вычислительные системы, в свою очередь, развиваются в направлении повышения доли графических процессоров (GPU) по сравнению с процессорами общего назначения (CPU). Это связано, в том числе, с существенно меньшими энергозатратами GPU в расчете на одну арифметическую операцию. Таким образом, весьма актуальной становится задача реализации гидродинамических моделей атмосферы на GPU. Еще в 2012 году региональная модель атмосферы COSMO была портирована на такие системы [3].

Типичная глобальная модель атмосферы, применяемая для прогноза погоды, содержит порядка многих сотен тысяч строк кода, чаще всего на языке Фортран. Многие модели используют двойную точность (64 бит) во всех вычислениях с плавающей точкой. Еще около 10 лет назад появились исследования, позволяющие утверждать, что большая часть вычислений в гидродинамических моделях атмосферы может быть переведена на одинарную точность [4, 13]. В то же время, расчеты на GPU показывают наибольший эффект при применении вычислений с одинарной точностью (32 бит). Логичным шагом при реализации существующих моделей прогноза погоды на GPU является перевод основной части вычислений на одинарную точность. Далее одним из возможных путей реализации большого кода модели на GPU является выделение сравнительно небольших фрагментов кода, включающих большой объем арифметических операций и которые могут быть с небольшими усилиями портированы на язык CUDA, с последующим вызовом этих фрагментов из кода на языке Фортан.

Данная статья описывает прогресс в дальнейшей реализации вычислений с одинарной точностью в параллельном программном комплексе глобальной модели атмосферы ПЛАВ. В разделе 2 представлено краткое описание модели ПЛАВ, ее параллельной программной реализации, развития алгоритмической части модели и некоторые примеры оперативных субсезонных прогнозов погоды. В разделе 3 представлены работы по дальнейшей реализации вычислений с одинарной точностью, выполненные после предыдущей публикации 2023 года, а также работы по ускорению MPI обменов между процессорами. В разделе 4 приводятся результаты вычислений.

2. Глобальная модель атмосферы ПЛАВ

Глобальная модель атмосферы ПЛАВ разработана в Институте вычислительной математики им. Г.И. Марчука РАН и Гидрометцентре России и применяется в Гидрометцентре России в конфигурациях с различным пространственным разрешением для оперативного детерминистического и ансамблевого прогноза погоды (до 10 суток) и ее осредненных аномалий (с заблаговременностью до нескольких месяцев).

В настоящее время применяются следующие конфигурации модели:

- **ПЛАВ10** предназначена для детерминистического среднесрочного прогноза погоды (до 10 суток), имеет горизонтальное разрешение 0.1° по долготе, переменное разрешение по широте от 0.08° в Северном полушарии до 0.13° в Южном полушарии. Сетка по вертикали имеет 104 узла. Размерность задачи (количество степеней свободы) составляет 7.3×10^8.
- **ПЛАВ20** применяется для ансамблевого (вероятностного) среднесрочного прогноза

с заблаговременностью до 10 суток. Разрешение этой версии модели составляет 0.225° по долготе, разрешение по широте меняется от 0.16 до 0.24 градуса, 51 уровень по вертикали. Размер ансамбля составляет 36. Размерность задачи 7.1×10^7.

- **ПЛАВ072L96** используется для субсезонного и сезонного вероятностного прогноза осредненных за неделю либо месяц аномалий погоды. Разрешение – 0.9° по долготе, 0.72° по широте, 96 уровней по вертикали, размер ансамбля также составляет 36, а количество степеней свободы - 9.6×10^6.

Отметим, что в 2024 решением Исполнительного совета Всемирной метеорологической организации Гидрометцентру России на основе аттестации прогнозов модели ПЛАВ072L96 был присвоен статус Глобального продуцирующего центра по субсезонным прогнозам погоды [14].

Динамическое ядро использует оригинальную формулировку прогностических уравнений с использованием абсолютной завихренности в качестве прогностической переменной, алгоритма восстановления компонент скорости горизонтального ветра, конечных разностей четвертого порядка. Применяется традиционная для таких моделей гибридная вертикальная координата, огибающая рельеф вблизи поверхности Земли, и плавно переходящая в координату давления в свободной атмосфере. Подробное описание динамического ядра приведено в [8].

Описание алгоритмов параметризаций, применяемых в модели ПЛАВ, и их усовершенствований приведено в [1].

Параллельная реализация модели использует сочетание технологий MPI и OpenMP, детальное описание приведено в [9]. Программный комплекс модели масштабируется вплоть до 13600 процессорных ядер. С 2018 года в модели и ее параллельной реализации был выполнен ряд усовершенствований [10, 11]:

- уточнение алгоритма расчета горизонтальной диффузии, позволившее увеличить шаг по времени в 2-3 раза в зависимости от конфигурации;
- оптимизация длины вектора в векторных арифметических операциях;
- перевод части вычислений в одинарную точность;
- асинхронный параллельный вывод прогностической продукции;
- оптимизация доступа в оперативную память для локальных массивов в блоке параметризаций.

Модель ПЛАВ в последние годы разрабатывается и применяется на вычислительной системе Cray XC40 пиковой производительностью 1.293 Пфлопс, установленной в ГВЦ Росгидромета. Система состоит из 936 вычислительных узлов, объединенных интерконнектом Aries. Каждый узел содержит 2 18-ядерных процессора Intel Xeon E2697v4. Используются компилятор Cray Fortran и оптимизированная под интерконнект Aries библиотека Cray MPI.

В итоге вышеприведенных усовершенствований время расчета прогноза на 24 часа для модели ПЛАВ10 с учетом времени на запись прогностической продукции сократилось с 42 минут на 3888 процессорных ядрах в 2019 году до 14 минут на 2916 ядрах в 2024 году. Однако и эти величины не соответствуют современным требованиям.

3. Усовершенствования и результаты

В модели ПЛАВ был реализован новый блок расчета коротко- и длинноволновой радиации ecRad, который позволил существенно ускорить соответствующие расчеты в модели при повышении точности долгосрочного прогноза [2]. В предназначенной для такого прогноза версии модели, ПЛАВ072L96, вычисления ускорились на 17%, а время расчета одного участника ансамбля прогноза с заблаговременностью четыре

месяца уменьшилось с 81 до 67 минут. В версии для среднесрочного ансамблевого прогноза ПЛАВ20 (горизонтальное разрешение 0,225° по долготе, 0,16–0,24° по широте, 51 уровень по вертикали) достигнуто ускорение около 15 % [2].

В версии модели ПЛАВ10, ориентированной на среднесрочный детерминистический прогноз, ускорение расчетов меньше, около 8 %: 13 минут расчета прогноза на 24 часа вместо 14. На рисунке 1 слева показаны подпрограммы, потреблявшие наибольшую долю времени до внедрения нового блока радиации при расчете прогноза ПЛАВ10 на 5 суток на 2916 ядрах вычислительной системы Cray XC40, а на рис. 1 справа – после внедрения этого блока. Использованная конфигурация расчетов содержала 81 вычислительный узел по 32 ядра на каждом, при этом на каждом узле работали 8 MPI-процессов по 4 потока OpenMP каждый.

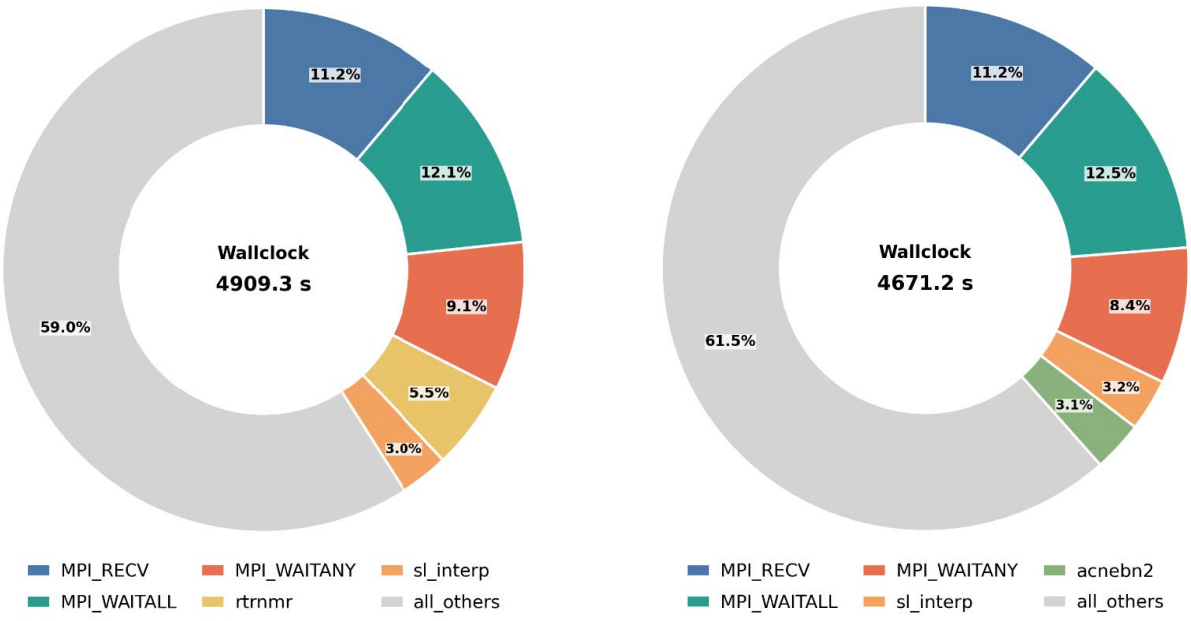


Рис. 1. Доля наиболее вычислительно затратных подпрограмм версии ПЛАВ10. Слева – с блоком радиации CLIRAD_SW + RRTMG LW, справа – ecRad. Внизу указано полное время счета прогноза на 5 сут с учетом чтения начальных данных и записи результата прогноза.

Можно видеть, что подпрограмма расчета длинноволновой радиации rrtmg_lw_rtrnrmr, рассчитывающая длинноволновое излучение поверхности и атмосферы с учетом отражения и рассеяния на облаках, больше не присутствует в диаграмме подпрограмм, потребляющих наибольшее время, приведенной на рис. 1.

Из рис. 1 также видно, что существенную часть времени счета прогноза (более 40 %) затрачивается на MPI обмены между процессорами. Для уменьшения этого времени в параллельном программном комплексе модели ПЛАВ были выполнены следующие усовершенствования:

1. В новую оперативную версию модели ПЛАВ в блоке полулагранжевого переноса внедрены MPI обмены данными между процессорами, определяемой актуальным полем ветра (обмены «по требованию»), ранее реализованные в экспериментальной версии. Подробно эта модификация и ее влияние на объем пересылаемых данных описана в [12].

2. Дополнительно уменьшен объем обменов данными за счет учета различной ширины шаблона интерполяции для разных интерполируемых в этом блоке слагаемых уравнений динамики.
3. Массивы прогностических переменных, ранее не переведенные в одинарную точность и участвующие в обменах данными между процессорами и соответствующие промежуточные величины (кроме влажности воздуха) переведены в одинарную точность. В итоге объем пересылаемых данных для этих переменных сократился вдвое.

Все эти усовершенствования проверялись на точность прогноза с помощью верификации рассчитываемых прогнозов в различных регионах Земли по стандарту Всемирной метеорологической организации, отдельно для версий ПЛАВ10 и ПЛАВ072L96. Данные модификации кода не привели к ухудшению качества прогнозов.

Эффект от этих усовершенствований для конфигурации ПЛАВ10 составил величину, примерно равную выигрышу за счет внедрения нового блока радиации, то есть одну минуту реального времени на 24 часа прогноза. При этом вклад MPI-обменов «по требованию» в уменьшение времени расчета примерно равен вкладу уменьшения объема пересылаемых данных за счет реализации пп.2 и 3, описанных выше. На рис. 2 показаны подпрограммы, потребляющие наибольший процент времени после усовершенствований. Видно, что при общем уменьшении времени счета на 7 % удалось сократить с 8.4 до 5.6% долю времени, затрачиваемую подпрограммой библиотеки MPI MPI_WAIT_ANY. Также немного сократилась доля времени, затрачиваемая подпрограммой MPI_WAITALL. Среди вычислительных подпрограмм больше всего времени занимает подпрограмма полулагранжевой адвекции (переноса) sl_interp, включающая интерполяции переносимых переменных в исходные точки траекторий, а на втором месте – подпрограмма расчета облачности acnebn2.

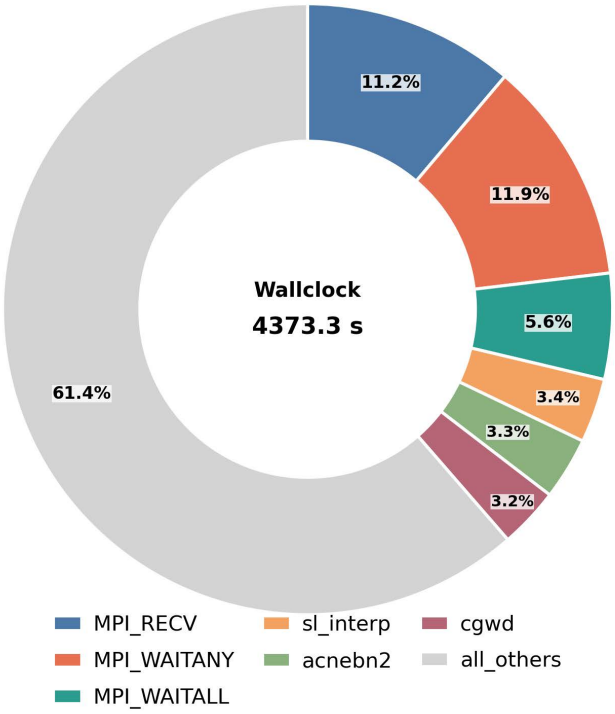


Рис. 2. Доля наиболее вычислительно затратных подпрограмм версии ПЛАВ10 с блоком радиации ecRad после оптимизации MPI-обменов и перехода на одинарную точность остающейся части массивов в блоке динамики.

Версия модели ПЛАВ для долгосрочного прогноза ПЛАВ072L96 требует меньших вычислительных ресурсов: обычно применяется от 32 до 192 процессорных ядер, размещенных в 1-6 вычислительных узлах. Объем пересылаемых между процессорами данных существенно меньше и выигрыш за счет вышеуказанных усовершенствований кода модели для этой версии модели незначителен. Распределение наиболее вычислительно затратных подпрограмм для прогноза на 123 дня почти не изменилось, оно приведено на рис. 3 для случая использования 6 вычислительных узлов по 32 ядра на каждом (всего 192 ядра).

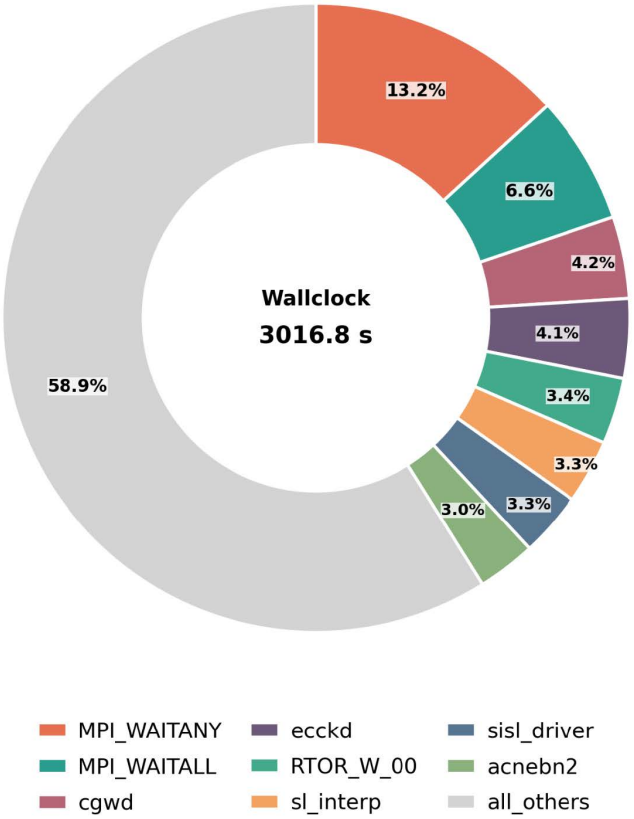


Рис. 3. Доля наиболее вычислительно затратных подпрограмм версии модели ПЛАВ072L96 с блоком радиации ecRad после оптимизации MPI-обменов и переходу на одинарную точность остающейся части массивов в блоке динамики.

Заключение

В результате выполненных работ по оптимизации MPI обменов данными между процессорами и дальнейшему переводу вычислений блока решения уравнений динамики атмосферы глобальной модели ПЛАВ10 с шагом сетки по горизонтали около 10 км удалось снизить затраты времени на расчет среднесрочных прогнозов примерно на 7% при использовании 2916 процессорных ядер системы Cray XC40, что означает экономию около 10 минут на расчет прогноза на 10 суток, который ранее занимал примерно 140 минут. Это является существенной величиной, с учетом ограниченного времени на расчет прогноза. Время расчета долгосрочных прогнозов по модели ПЛАВ с небольшим размером вычислительной области практически не изменилось.

Список литературы

[1] Толстых М.А., Фадеев Р.Ю., Шашкин В.В, Зарипов Р.Б., Травова С.В., Гойман Г.С., Алипова К.А., Мизяк В.Г., Тищенко В.А., Круглова Е.Н. Модель долгосрочного метеорологического прогноза ПЛАВ072L96 // Метеорология и гидрология. 2024. № 7. С. 25-39.

[2] Фадеев Р.Ю., Толстых М.А., Бирючева Е.О., Гойман Г.С. Включение параметризации ecRad в модель ПЛАВ и ее влияние на атмосферную циркуляцию на годовом и сезонном масштабах // Гидрометеорологические исследования и прогнозы. 2025. № 3 (397). С. 49-63 DOI: 10.37162/2618-9631-2025-3-49-63

[3] Fuhrer O., Chadha T., Hoefler T., Kwasniewski G., Lapillonne X., Leutwyler D., Lüthi D, Osuna C., Schär C., Schulthess T., Vogt H. Near-global climate simulation at 1km resolution // Geosci. Model. Dev. 2018. Vol.11, No.4. P. 1665-1681. DOI: 10.5194/gmd-11-1665-2018

[4] Lang S., Dawson A., Diamantakis M., Düben P., Hatfield S., Leutbecher M., Palmer T., Prates F., Roberts C., Sandu I., Wedi N. More Accuracy with Less Precision // Quart. J. Roy. Meteor. Soc. 2021. Vol. 147. DOI: 10.1002/qj.4181.

[5] Lang S., and Coauthors, 2024: AIFS–ECMWF's data-driven forecasting system. arXiv, preprint 551 arXiv:2406.01465v2.

[6] McTaggart-Cowan R., Magnusson L., Polichtchouk I., et al. WP-MIP: An Artificial Intelligence, Hybrid and Physically Based Model Intercomparison Project for Weather Prediction. Submitted to Bull. Amer. Meteor. Soc. 2026. https://arxiv.org/pdf/2604.16643

[7] Tolstykh, M., Shashkin, V., Fadeev, R., Goyman, G.: Vorticity-divergence semi-Lagrangian global atmospheric model SL-AV20: dynamical core // Geosci. Model Dev. 2017. Vol. 10. P. 1961-1983. DOI: 10.5194/gmd-10-1961-2017.

[8] Tolstykh M., Goyman G., Fadeev R., Shashkin V. Structure and algorithms of SL-AV atmosphere model parallel program complex // Lobachevskii Journal of Mathematics. 2018. Vol. 39. P. 587-595.

[9] Tolstykh, M., Goyman, G., Fadeev, R., Shashkin, V., Lubov, S.: SL-AV model: numerical weather prediction at extra-massively parallel supercomputer. // Voevodin, V., Sobolev, S. (eds.) RuSCDays 2018. CCIS, vol. 965, pp. 379–387. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05807-4_32

[10] Tolstykh M.A., Fadeev R.Yu., Shashkin V.V., Goyman G.S. Improving the computational efficiency of the global SL-AV numerical weather prediction model // Supercomputer Frontiers and Innovations. 2021. Vol. 8, No. 4. P. 11-23. DOI: 10.14529/jsfi210402.

[11] Tolstykh M.A., Goyman G.S., Biryucheva E.O., Shashkin V.V., Fadeev R.Yu. Reduced Precision Computations in the SL-AV Global Atmosphere Model // Supercomputing. 9th Russian Supercomputing Days, Moscow, Russia, September 25–26, 2023. Revised Selected Papers, Part I, Lecture Notes in Computer Sciences. 2023. V. 14388, pp. 190–201. https://doi.org/10.1007/978-3-031-49432-1_15

[12] TOP500 List - November 2025 | TOP500. URL: https://top500.org/lists/top500/list/2025/11/?page=3 (дата обращения 30.04.2026)

[13] Váňa, F., Düben ,P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., Carver G. Single Precision in Weather Forecasting Models: An Evaluation with the IFS // Mon. Wea. Rev. 2017. Vol. 145. P. 495–502. DOI: 10.1175/MWR-D-16-0228.1.

[14] WMO Lead Centre for Sub-seasonal forecast multi-model ensemble. URL: https://charts.ecmwf.int/wmo/ (дата обращения 30.04.2026)

Иммерсивные технологии виртуальной и дополненной реальности в преподавании компьютерного зрения для авиационных информационных систем*

М.В. Медведев[1], С.В. Новикова[1], А.С. Сытник[1], М.П. Шлеймович[1]
[1]Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ (КНИТУ-КАИ)

В статье рассматривается инновационная методика преподавания технологий компьютерного зрения в рамках магистерской образовательной программы «Интеллектуальная обработка данных в авиационных системах», реализуемой в специальном образовательном пространстве Передовой инженерной школы «Комплексная авиационная инженерия» КНИТУ-КАИ. Предложенная методика основана на интеграции образовательного процесса, научно-исследовательской деятельности студентов и практико-ориентированного обучения. Особое внимание уделяется формированию профессиональных компетенций в области обработки изображений, машинного обучения и разработки интеллектуальных систем для авиационных приложений. В работе описана педагогическая модель обучения, включающая проектно-ориентированную подготовку, использование специализированных лабораторий компьютерного зрения и применение иммерсивных технологий. Рассматриваются возможности использования виртуальной и дополненной реальности для моделирования технологических процессов, проведения виртуальных лабораторных работ и формирования практических инженерных навыков. Показано, что интеграция технологий компьютерного зрения, виртуальной и дополненной реальности способствует повышению эффективности подготовки специалистов в области интеллектуальной обработки данных и созданию современной цифровой образовательной среды для инженерного образования.

Ключевые слова: компьютерное зрение, интеллектуальная обработка данных, машинное обучение, инновационные образовательные технологии, проектно-ориентированное обучение, авиационные системы, виртуальная реальность, дополненная реальность, иммерсивные образовательные технологии, виртуальные лаборатории

1. Введение

Современное развитие авиационной техники, беспилотных летательных аппаратов и интеллектуальных информационных систем требует подготовки специалистов нового поколения, обладающих компетенциями в области анализа данных, машинного обучения и компьютерного зрения [1-3]. Одним из ключевых направлений подготовки таких специалистов является образовательная программа магистратуры «Интеллектуальная обработка данных в авиационных системах», реализуемая в Казанском национальном исследовательском техническом университете им. А.Н. Туполева-КАИ (КНИТУ-КАИ) [4]. Данная программа направлена на формирование инженерных и научных компетенций, необходимых для разработки программных и аппаратных средств интеллектуальной обработки информации.

Актуальность разработки инновационных методик преподавания компьютерного зрения обусловлена несколькими факторами:

*Работа выполнена в рамках государственного задания Министерства науки и высшего образования FZSU-2026-0007, рег. номер НИОКТР 126020516513-4.

- быстрым развитием технологий искусственного интеллекта;
- ростом потребности авиационной отрасли в специалистах по анализу визуальных данных;
- необходимостью интеграции научных исследований и образовательного процесса;
- переходом к компетентностной модели подготовки инженеров.

Целью данной работы является анализ и описание инновационной методики преподавания технологий компьютерного зрения, реализуемой в рамках образовательной программы «Интеллектуальная обработка данных в авиационных системах».

2. Особенности образовательной программы

Образовательная программа ориентирована на подготовку магистров по направлению 09.04.02 «Информационные системы и технологии» и реализуется в специальном образовательном пространстве, объединяющем образовательную, научно-исследовательскую и проектную деятельность. Основной миссией программы является подготовка высококвалифицированных специалистов, способных разрабатывать программное обеспечение для интеллектуальных систем обработки данных, применяемых в авиационной отрасли.

Подготовка специалистов в области искусственного интеллекта, анализа данных и компьютерного зрения осуществляется в ряде российских университетов, включая МФТИ, Университет ИТМО, НИУ ВШЭ и Университет Иннополис. Большинство подобных образовательных программ ориентировано на изучение методов машинного обучения, анализа данных и разработки интеллектуальных систем. Отличительной особенностью рассматриваемой программы является её ориентация на авиационные информационные системы, а также интеграция проектно-ориентированного обучения, научно-исследовательской деятельности студентов и технологий виртуальной и дополненной реальности в рамках единой образовательной среды.

Ключевые особенности программы включают:

- интеграцию технологий искусственного интеллекта в образовательный процесс [5];
- практико-ориентированную подготовку студентов [6];
- тесное взаимодействие с промышленными предприятиями [7];
- активное вовлечение обучающихся в научные исследования [8].

Особое внимание уделяется формированию компетенций в следующих областях:

- интеллектуальные информационные системы;
- технологии компьютерного зрения и обработки изображений;
- машинное обучение и распознавание образов;
- моделирование информационных процессов;
- разработка программного обеспечения для авиационных систем.

Таким образом, образовательная программа ориентирована на подготовку специалистов, способных решать сложные научно-исследовательские и инженерные задачи.

3. Методика преподавания компьютерного зрения

Научная новизна предлагаемой методики преподавания технологий компьютерного зрения заключается в разработке интегрированной педагогической модели подготовки специалистов в области интеллектуальной обработки визуальных данных для авиационных систем. Основные элементы научной новизны можно сформулировать следующим образом:

1. **Многоуровневая модель формирования компетенций в области компьютерного зрения**, включающая последовательные этапы подготовки: фундаментальную математическую подготовку, освоение методов цифровой обработки изображений, изучение алгоритмов компьютерного зрения и разработку инженерных интеллектуальных систем.
2. **Интеграция научно-исследовательской деятельности студентов в образовательный процесс**, при которой учебные дисциплины, лабораторные исследования и научные проекты образуют единую систему подготовки специалистов.
3. **Проектно-ориентированная модель обучения**, реализующая полный цикл разработки интеллектуальной системы компьютерного зрения от постановки задачи до программной реализации и экспериментальной проверки.
4. **Модель интеграции технологий виртуальной реальности в инженерное образование**, обеспечивающая иммерсивную подготовку студентов и повышение эффективности формирования практических навыков.
5. **Образовательная среда подготовки специалистов для авиационной отрасли**, объединяющая лаборатории компьютерного зрения, лаборатории виртуальной реальности, научные исследования и проектную деятельность студентов.

Предложенная методика позволяет повысить эффективность формирования инженерных компетенций в области разработки интеллектуальных систем анализа визуальных данных.

3.1. Проектно-ориентированное обучение

В рамках дисциплин, связанных с компьютерным зрением, студенты выполняют проекты, ориентированные на решение реальных задач анализа изображений. Примерами таких задач являются обнаружение и распознавание объектов на изображениях [9], определение характеристик объектов, анализ видеопотоков [10], обработка данных с беспилотных летательных аппаратов [11] и др. (Рис. 1).

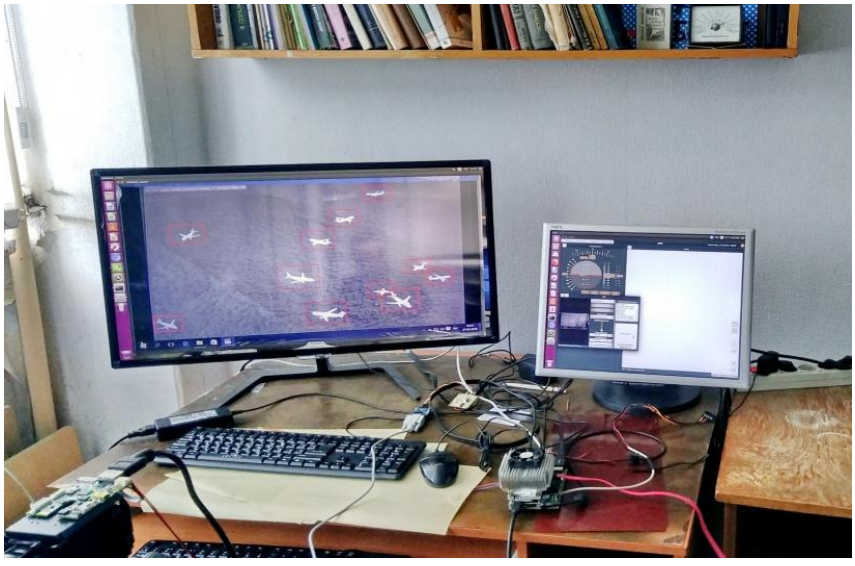


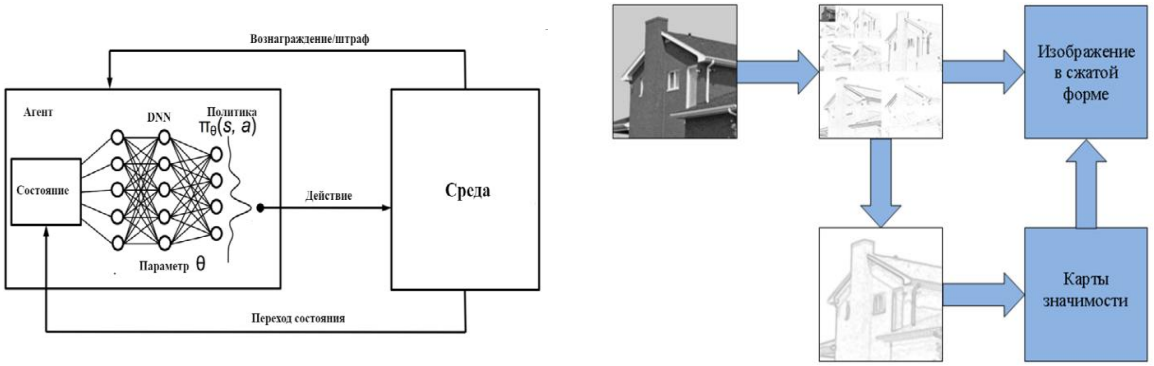
Рис. 1. Стенд наземных отработок информации, полученной с БПЛА.

Проектная деятельность позволяет студентам развивать навыки постановки инженерной задачи, разработки алгоритмов обработки изображений, обучения нейронных сетей, анализа результатов экспериментов.

3.2. Интеграция научных исследований

Особенностью образовательной программы является активное участие студентов в научно-исследовательской деятельности. Научные исследования направлены на разработку новых методов обработки изображений и обучения нейронных сетей (Рис. 2). Среди ключевых результатов можно выделить:

- методы расширения графических датасетов для задач обнаружения объектов [12];
- методы сжатия изображений на основе весовых моделей [13];
- методы дешифрирования тепловизионных изображений с использованием сверточных нейронных сетей [14];
- методы обнаружения радиально-симметричных объектов на изображениях [15].



а) б)

Рис. 2. Примеры студенческих научных разработок.
а) расширение графических датасетов на основе ИИ; б) сжатие изображений.

Участие студентов в таких исследованиях способствует развитию научного мышления и практических навыков работы с современными алгоритмами компьютерного зрения.

3.3. Использование лабораторной инфраструктуры

Важным элементом инновационной методики является использование специализированных лабораторий компьютерного зрения и виртуальной реальности. В лаборатории компьютерного зрения студенты выполняют исследования в области:

- улучшения и восстановления изображений;
- поиска изображений по заданным характеристикам;
- обнаружения и локализации объектов;
- измерения параметров объектов;
- анализа ключевых особенностей изображений;
- сжатия изображений.

Использование современного оборудования позволяет моделировать реальные условия работы интеллектуальных систем.

3.4. Отличительные особенности предлагаемой методики

В отличие от традиционных подходов к преподаванию компьютерного зрения, ориентированных преимущественно на изучение теоретических основ и выполнение лабораторных работ на стандартных наборах данных, предлагаемая методика основана на интеграции образовательного процесса, научных исследований и проектной

деятельности студентов. Обучающиеся участвуют в полном цикле разработки интеллектуальных систем — от постановки инженерной задачи и формирования датасетов до создания программной реализации и экспериментальной проверки разработанных алгоритмов на реальных данных, включая данные беспилотных летательных аппаратов. Дополнительным отличием методики является использование технологий виртуальной и дополненной реальности в сочетании с проектно-ориентированным обучением. Виртуальные лаборатории обеспечивают возможность безопасной отработки технологических операций, организации коллективной работы обучающихся в многопользовательской среде и формирования практических инженерных компетенций. Таким образом, предложенный подход объединяет технологии компьютерного зрения, научно-исследовательскую деятельность, проектное обучение и иммерсивные технологии в единую образовательную модель подготовки специалистов для авиационных информационных систем.

4. Практическая реализация методики

Практическая реализация инновационной методики осуществляется через выполнение прикладных проектов. Примерами реализованных проектов являются:

- система подсчета и классификации транспортных средств на перекрестках;
- система оценки состояния дорожного покрытия;
- система обнаружения препятствий для сельскохозяйственной техники.

Примеры обрабатываемых в ходе выполнения прикладных проектов изображений приведены на Рис. 3.

а) б) в)

Рис. 3. Примеры практического применения методики проектного обучения: а) подсчет и классификация транспорта; б) обнаружение препятствий сельхозтехники; в) система обработки данных БПЛА

Такие проекты позволяют студентам применять методы компьютерного зрения для решения задач из различных областей: транспорта, сельского хозяйства и авиации.

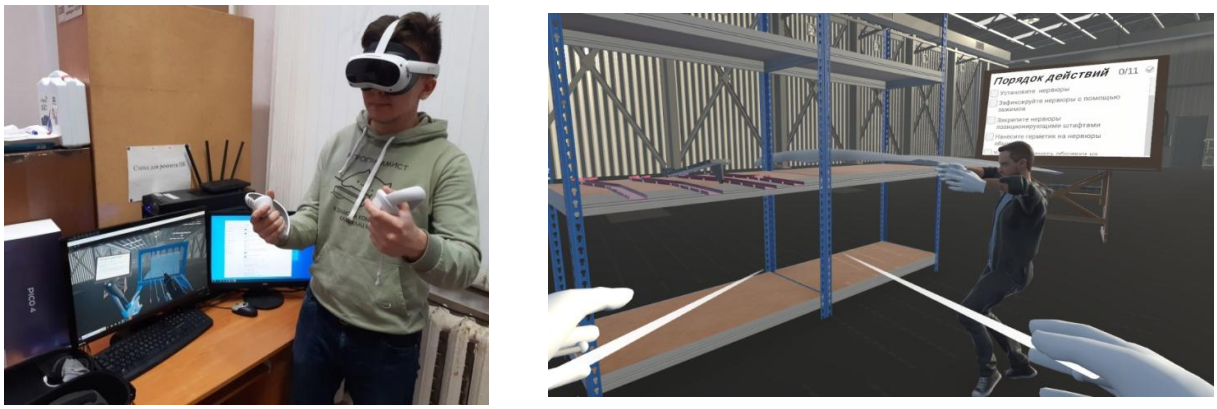
Кроме того, в рамках программы разрабатываются системы обработки данных с беспилотных летательных аппаратов. Камеры, интегрированные с одноплатными компьютерами, позволяют выполнять анализ окружающей среды в режиме реального времени. Это создает условия для формирования у студентов компетенций, необходимых для разработки автономных интеллектуальных систем.

5. Роль виртуальной реальности в образовательном процессе

Важным элементом инновационной образовательной среды подготовки специалистов в области интеллектуальной обработки данных является использование технологий виртуальной и дополненной реальности. Применение иммерсивных технологий позволяет существенно расширить возможности практико-ориентированного обучения и повысить эффективность формирования профессиональных инженерных компетенций.

В КНИТУ-КАИ разработан ряд виртуальных лабораторных тренажеров, предназначенных для подготовки специалистов авиационной отрасли [16]. Использование виртуальной среды позволяет моделировать реальные производственные процессы и технологические операции, что особенно важно в условиях ограниченного доступа к дорогостоящему оборудованию и сложным производственным установкам.

Студенты участвуют в разработке трехмерных моделей объектов, виртуальных тренажеров, интерактивных образовательных систем. Примером является виртуальный тренажер сборки элементов авиационных конструкций, реализованный в среде Unity [17], элементы которого представлены на Рис. 4. В рамках данной системы студент взаимодействует с трехмерными моделями деталей и выполняет последовательность технологических операций, включающих позиционирование элементов конструкции, сверление отверстий и установку крепежных элементов. Виртуальная система контролирует корректность действий обучаемого и не позволяет переходить к следующему этапу технологического процесса до завершения предыдущего, если последовательность операций нарушена.



а) б)

Рис. 4. Лабораторные занятия в VR-лаборатории:
а) студент с VR-оборудованием; б) учебная сцена внутри симулятора

Другим примером виртуальной лабораторной работы является моделирование процесса пропитки композиционных материалов с использованием инжекционных технологий [18]. В виртуальной среде воспроизводится работа технологического оборудования, включая термопресс и вспомогательные инструменты. Студент должен правильно подготовить заготовку изделия, разместить её в установке и выполнить последовательность операций согласно технологическому регламенту. При этом виртуальная среда моделирует физические свойства материалов и взаимодействие пользователя с инструментами.

Существенной особенностью разработанных систем является поддержка многопользовательского режима работы. Виртуальная лаборатория функционирует на основе клиент-серверной архитектуры, представленной на Рис.5: компьютер преподавателя выполняет функции сервера, а рабочие станции студентов подключаются к системе в качестве клиентов. Пользователи представлены в виртуальной среде в виде анимированных аватаров, что позволяет наблюдать действия других участников и взаимодействовать с ними в процессе выполнения лабораторных заданий. Такой подход способствует формированию навыков коллективной инженерной работы и повышает вовлеченность обучающихся в образовательный процесс [19].

Рис. 5. Архитектура виртуальной лаборатории и педагогические результаты её использования в образовательном процессе.

Использование виртуальной реальности обладает рядом значительных преимуществ для инженерного образования. Прежде всего, виртуальная среда позволяет многократно воспроизводить сложные технологические процессы без риска повреждения оборудования и без дополнительных затрат на материалы [20]. Студенты могут экспериментировать с различными вариантами выполнения операций и анализировать последствия своих действий, что способствует более глубокому пониманию технологических процессов.

При этом виртуальные технологии рассматриваются не как замена традиционных лабораторных занятий, а как их эффективное дополнение. Наиболее результативной является комбинированная модель обучения, при которой виртуальные тренажеры используются для предварительного освоения технологических операций, а работа с реальным оборудованием служит для закрепления полученных навыков.

Перспективным направлением развития образовательных технологий является внедрение систем дополненной реальности [21]. В этом случае студент выполняет лабораторную работу на реальной установке, а в очках дополненной реальности отображаются интерактивные подсказки, технологические параметры и последовательность выполнения операций. Такой подход позволяет объединить преимущества виртуальной среды и реального производственного оборудования.

Таким образом, интеграция технологий виртуальной и дополненной реальности в образовательный процесс способствует созданию иммерсивной образовательной среды, обеспечивающей эффективное формирование практических инженерных компетенций и повышение качества подготовки специалистов для высокотехнологичных отраслей промышленности, включая авиационную отрасль.

6. Оценка эффективности предлагаемой методики

Для оценки эффективности предлагаемой методики был выполнен сравнительный анализ результатов обучения студентов до и после её внедрения. В качестве контрольной группы рассматривался выпуск 2024 года, обучавшийся по предыдущей версии образовательной программы, а в качестве экспериментальной группы — выпуск 2026 года, завершивший обучение после внедрения проектно-ориентированного подхода, интеграции научно-исследовательской деятельности и технологий виртуальной и дополненной реальности.

Сравнение результатов показало положительную динамику основных образовательных показателей. Доля студентов, успешно завершивших обучение по программе, увеличилась с 42,1 % до 76,2 %, то есть на 34,1 процентного пункта. Количество выпускников, завершивших обучение на оценки «отлично» и «хорошо», возросло с 5 до 12 человек. Сравнение показателей до и после внедрения методики наглядно представлено на Рис. 6-7.

а) б)

Рис. 6. Доля студентов, успешно завершивших обучение по образовательной программе: а) выпуск 2024 г. (до внедрения методики); б) выпуск 2026 г. (после внедрения методики)

Рис. 7. Влияние внедрения образовательной методики на качество подготовки магистров.

Полученные результаты свидетельствуют о положительном влиянии предлагаемой методики на академическую успешность обучающихся и качество подготовки специалистов в области интеллектуальной обработки данных и компьютерного зрения.

Заключение

В статье представлена инновационная методика преподавания технологий компьютерного зрения, реализуемая в рамках магистерской образовательной программы «Интеллектуальная обработка данных в авиационных системах». Предлагаемый подход основан на интеграции образовательного процесса, научно-исследовательской деятельности студентов, проектно-ориентированного обучения и технологий виртуальной и дополненной реальности в единую образовательную среду подготовки специалистов для авиационной отрасли. Показано, что использование реальных инженерных задач, участие студентов в научных исследованиях и применение специализированной лабораторной инфраструктуры способствуют формированию профессиональных компетенций в области компьютерного зрения, машинного обучения и интеллектуальной обработки данных. Важным элементом методики является использование виртуальных лабораторий и многопользовательских VR-сред, обеспечивающих безопасную отработку технологических операций, развитие навыков коллективной инженерной работы и повышение вовлеченности обучающихся в образовательный процесс.

Результаты сравнительного анализа эффективности образовательной программы показали положительную динамику основных показателей подготовки специалистов после внедрения предложенной методики. Доля студентов, успешно завершивших обучение по программе, увеличилась с 42,1 % до 76,2 %, а количество выпускников, завершивших обучение на оценки «отлично» и «хорошо», возросло с 5 до 12 человек. Полученные результаты подтверждают эффективность интеграции проектного обучения, научно-исследовательской деятельности и иммерсивных технологий в процесс подготовки специалистов по интеллектуальной обработке данных и компьютерному зрению.

Таким образом, предложенная методика позволяет повысить качество подготовки кадров для высокотехнологичных отраслей промышленности и может быть рекомендована для внедрения в образовательные программы, связанные с искусственным интеллектом, компьютерным зрением и разработкой интеллектуальных информационных систем. Перспективы дальнейших исследований связаны с расширением применения технологий виртуальной и дополненной реальности, развитием цифровых образовательных платформ и внедрением современных методов искусственного интеллекта в инженерное образование.

Список литературы

[1] Ермаков К.С. Исследование современных подходов к обучению и разработке квалификационных требований к пилоту/оператору беспилотной авиационной системы в соответствии со стандартами ИКАО // Научный вестник Московского государственного технического университета гражданской авиации. 2023. Т. 26. № 1. С. 34–48. DOI: 10.26467/2079-0619-2023-26-1-34-48.

[2] Грибачев К.К., Галко Д.Г. Системный подход к отбору и обучению операторов беспилотных летательных аппаратов с использованием симуляторов и математических моделей // Организационная психология и психология труда. 2025. Т. 10. № 1. С. 131–172. DOI: 10.38098/ipran.opwp_2025_34_1_006.

[3] Katkuri A.V.R., Madan H., Khatri N., Abdul-Qawy A., Patnaik K. Autonomous UAV Navigation using Deep Learning-Based Computer Vision Frameworks: A Systematic Literature Review // Array. 2024. Vol. 23. Article 100361. DOI: 10.1016/j.array.2024.100361.

[4] Передовая инженерная школа «Комплексная авиационная инженерия». URL: https://kai.ru/web/pish/09.04.02 (дата обращения: 17.03.2026).

[5] Сытник А.С., Фролова А.В., Шлеймович М.П. Интеллектуальные информационные системы – главный технологический тренд подготовки IT-специалистов на кафедре АСОИУ КНИТУ-КАИ // Вестник НЦБЖД. 2023. № 2(56). С. 71–79.

[6] Жмайло М.В., Гукаленко О.В. Тенденции практико-ориентированной подготовки будущих инженеров в зарубежных технических вузах в эпоху индустрии 4.0 // Отечественная и зарубежная педагогика. 2025. № 4. С. 28–40.

[7] Толстых Т.О., Кочетова О.О. Кооперация вузов и промышленных предприятий для достижения технологического суверенитета страны // Регион: системы, экономика, управление. 2023. № 4(63). С. 77–84. DOI: 10.22394/1997-4469-2023-63-4-77-84.

[8] Панина Е.А. Научно-исследовательская деятельность студентов как базовый компонент будущей профессиональной деятельности // Вестник Майкопского государственного технологического университета. 2025. Т. 17. № 1. С. 70–82. DOI: 10.47370/2078-1024-2025-17-1-70-82.

[9] Song Sh., Liu Ju., Shleimovich M.P. et al. An Adaptive Radial Object Recognition Algorithm for Lightweight Drones in Different Environments // Computer Optics. 2025. Vol. 49. No. 3. P. 480–492. DOI: 10.18287/2412-6179-CO-1534.

[10] Колобова Д.А., Шлеймович М.П. Генерация видео подстилающей поверхности на основе одиночного снимка // Вестник Технологического университета. 2025. Т. 28. № 12. С. 124–134. DOI: 10.55421/3034-4689_2025_28_12_124.

[11] Моисеев В.С., Новикова С.В., Валитова Н.Л., Кремлева Э.Ш. Интегрированная модель управления БПЛА для мониторинга объектов с ограниченно известным местоположением // Вестник Технологического университета. 2025. Т. 28. № 10. С. 115–119. DOI: 10.55421/3034-4689_2025_28_10_115.

[12] Евдокимова Т.С. Влияние методов расширения наборов данных на качество обучения нейросетевых моделей. Адаптивный подход расширения наборов данных // Инженерный вестник Дона. 2024. № 8(116). С. 282–290.

[13] Ляшева М.М. Сжатие изображений в информационно-измерительных системах // САПР и моделирование в современной электронике: труды IV Международной научно-практической конференции. Брянск, 2020. С. 113–114. DOI: 10.51932/9785907271739_113.

[14] Мингалев А.В., Белов А.В., Шушарин С.Н., Шлеймович М.П. Способ семантической сегментации тепловизионных изображений // Инженерный вестник Дона. 2025. № 3(123). С. 313–328.

[15] Шакирзянов Р.М. Обнаружение сигналов светофоров с использованием цветовой сегментации и детектора радиальной симметрии // Вестник Воронежского государственного технического университета. 2020. Т. 16. № 6. С. 25–33. DOI: 10.36622/VSTU.2020.16.6.004.

[16] Борисов Д.М., Медведев М.В. Система проведения лабораторных работ в виртуальной реальности // Технологии и техника: пути инновационного развития: материалы 2-й Международной научно-технической конференции. Воронеж, 2024. С. 95–97.

[17] Программа для демонстрации и обучения процессу сборки авиационных конструкций в виртуальной среде: свидетельство о государственной регистрации программы

для ЭВМ № 2024681675. Опубл. 12.09.2024.

[18] Программа для демонстрации и обучения технологии производства композитных изделий в виртуальной среде: свидетельство о государственной регистрации программы для ЭВМ № 2024681739. Опубл. 12.09.2024.

[19] Шелепаева А.Х. Виртуальная реальность: практика встраивания в образовательный процесс // Открытое образование. 2025. Т. 29. № 4. С. 19–28. DOI: 10.21686/1818-4243-2025-4-19-28.

[20] Ананин Д.П., Сувирова А.Ю. Иммерсивные технологии в образовательной практике российской высшей школы // Высшее образование в России. 2024. Т. 33. № 5. С. 112–135. DOI: 10.31992/0869-3617-2024-33-5-112-135.

[21] Данилова Т.В., Бурыкина М.Ю., Крамарева И.Е. Использование виртуальной и дополненной реальности в высшем образовании // Управление образованием: теория и практика. 2024. № 1–2. С. 133–142. DOI: 10.25726/f9680-6664-9306-k.

Использование GPU для ускорения расчета аэродинамического шума на основе решения решеточных уравнений Больцмана

К.К. Забело[1,2], А.В. Гарбарук[1]

[1]Санкт-Петербургский политехнический университет Петра Великого
[2]Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики

Предсказание аэродинамического шума является важной задачей при проектировании транспортных средств, вентиляционных систем и других технических устройств. Одним из наиболее точных подходов к ее решению является метод, основанный на первых принципах, в котором сначала производится расчет источника шума с использованием методов вычислительной гидроаэродинамики, а затем вычисляется распространение шума с помощью интегральных методов. В работе рассматривается применение графических процессоров на архитектуре CUDA для ускорения моделирования расчета аэродинамического шума с использованием решения решеточных уравнений Больцмана. В рамках GPU-версии алгоритма реализована модель D2Q9 с BGK-оператором столкновений и рекурсивной регуляризацией. Проведено численное исследование обтекания круглого цилиндра при числе Рейнольдса 150 и Маха 0.2 с последующим расчетом акустического поля методом Фокса Вильямса–Хокингса. Полученные результаты хорошо согласуются с литературными данными и результатами расчетов в ANSYS Fluent. Показано значительное ускорение вычислений (более чем на три порядка) по сравнению с ANSYS Fluent. Проведено тестирование на сильное и слабое масштабирование программы при проведении расчетов на нескольких видеокартах с использованием MPI. Продемонстрирована хорошая эффективность как для сильного, так и слабого масштабирования.

Ключевые слова: метод решеточных уравнений Больцмана, аэроакустика, GPU-вычисления, LBM, аэродинамический шум, метод Фокса Вильямса–Хокингса, обтекание цилиндра, высокопроизводительные вычисления, численное моделирование

1. Введение

Метод решеточных уравнений Больцмана (РУБ, или LBM — Lattice Boltzmann Method) предназначен для решения модельных кинетических уравнений и позволяет описывать течения жидкости или газа. Данный метод широко используется благодаря простоте реализации, высокой масштабируемости и способности моделировать сложные течения. В последние годы непрерывно возрастает интерес к применению РУБ в задачах аэроакустики, где, по мнению ряда исследователей, он способен обеспечить более высокую эффективность по сравнению с традиционными численными методами [1].

На практике при решении задач аэроакустики часто используется двухэтапный метод расчета шума, предполагающий раздельное моделирование генерации и распространения звука. Это связано с тем, что расстояние до наблюдателя, как правило, многократно превосходит длину акустических волн, поэтому прямой расчет их распространения до наблюдателя сопряжен с использованием огромных по размеру сеток и, как следствие, с неприемлемо высокими вычислительными затратами. На

первом этапе двухэтапного метода производится гидродинамический расчет, в процессе которого информация о нестационарных характеристиках потока сохраняется на контрольных поверхностях. На втором этапе с использованием этой информации производится расчет распространения звука до положения наблюдателя с помощью соответствующих интегральных подходов, таких как метод Кирхгофа [2] или Фокса Вильямса–Хокингса (FWH) [3].

Эффективность данного подхода во многом определяется возможностью точного описания нестационарных акустических возмущений на этапе гидродинамического расчета. В этой связи особую актуальность приобретает разработка высокопроизводительных реализаций метода РУБ, способных обеспечивать требуемую точность при разумных вычислительных затратах.

В настоящей работе рассматривается реализация метода решеточных уравнений Больцмана для решения задач аэроакустики с использованием графических процессоров. Численный алгоритм реализован для изотермической модели D2Q9 с BGK-оператором столкновений и рекурсивной регуляризацией функции распределения. Для повышения эффективности использования памяти применяется схема EsoTwist на неструктурированной сетке, позволяющая сократить объем используемой видеопамяти за счет хранения функций распределения в одном массиве.

Целью настоящей работы является валидация разработанного подхода и оценка производительности его реализации на графических процессорах. Для этого используется задача генерации и распространения аэродинамического шума при ламинарном обтекании круглого цилиндра при числе Рейнольдса $Re = 150$ и числе Маха $M = 0.2$. Полученные результаты сопоставляются с данными, представленными в литературе [4-7], а также с результатами расчетов в коммерческом пакете ANSYS Fluent.

2. Постановка задачи

Рассматривается задача расчета акустического шума, возникающего при обтекании двумерного цилиндра диаметром D однородным потоком со скоростью U_0 при числах Рейнольдса и Маха $Re = \frac{U_0 D}{\nu} = 150$ и $M = \frac{U_0}{c_s}$, соответственно, где ν – коэффициент кинематической вязкости, c_s – скорость звука. Параметры данной задачи были выбраны в соответствии со статьей [4], в которой проводился расчет сжимаемого течения на сетке, обеспечивающей разрешение акустических волн вплоть до положения наблюдателей. В настоящей работе, следуя обозначениям [4], этот расчет называется «прямое численное моделирование» (DNS), хотя термин обычно не применяют к двумерным ламинарным течениям.

Течение при таком числе Рейнольдса является ламинарным и нестационарным, оно описывается системой уравнений Навье–Стокса для сжимаемого газа:

$$\begin{cases} \partial_t \rho + \partial_\alpha(\rho u_\alpha) = 0; \\ \partial_t(\rho u_\alpha) + \partial_\beta(\rho u_\alpha u_\beta) = -\partial_\alpha p + \partial_\beta \left(\mu(\partial_\beta u_\alpha + \partial_\alpha u_\beta - \frac{2}{3}\delta_{\alpha\beta}\partial_\gamma u_\gamma) + \mu_b \delta_{\alpha\beta}\partial_\gamma u_\gamma \right), \end{cases}$$

где μ и $\mu_b = \frac{2}{3}\mu$ – коэффициенты динамической и объемной вязкости соответственно. Система замыкается уравнением состояния. Поскольку в рассматриваемом случае числа Маха малы, можно использовать изотермическое уравнение состояния:

$$p - p_0 = (\rho - \rho_0)c_s^2$$

Расчетная область имеет форму квадрата со стороной $D_{out} = 200D$ (рис. 1 а). В центре расчетной области располагается цилиндр. На внешней границе задаются

характеристические граничные условия [8] в сочетании с демпфирующим слоем (PML [9]). Демпфирующий слой определяется толщиной ($L_{PML} = 11.6D$) и коэффициентом демпфирования σ, меняющимся по параболическому закону от 0 до $\sigma_{max} = 0.05$ к границе области. На поверхности цилиндра задается граничное условие обратного отскока с интерполяцией [10].

Расчетная сетка является O-октри-сеткой с базовым шагом $\Delta x_0 = 0.025D$ с количеством уровней $N_{lvl} = 4$ и $N_{\Delta x} = 20$ узлов в радиальном направлении между уровнями (рис. 1 б). Переход между уровнями сетки осуществляется с использованием компактной интерполяции второго порядка [11]. Шаг по времени на самом грубом уровне составляет менее 0.5% от периода колебаний подъемной силы.

Для расчета шума использовалась непроницаемая FWH-поверхность, совпадающая с поверхностью цилиндра. Поскольку метод FWH является трехмерным, FWH поверхность была вытянута на $5000D$ в третьем направлении. Наблюдатели располагались на окружности радиуса $R = 100D$.

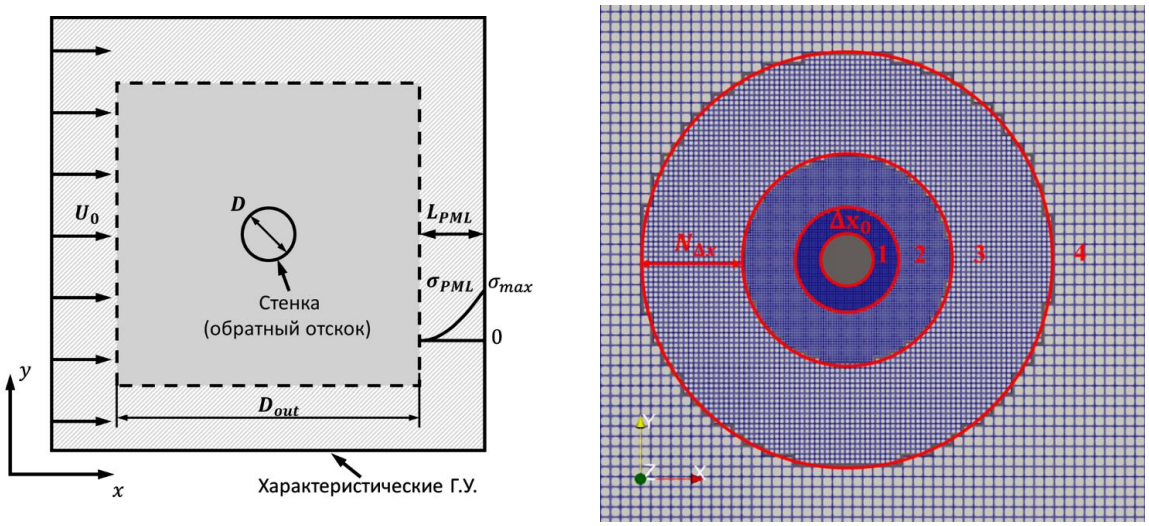


Рис. 1. Постановка задачи: а) Схема расчетной области; б) Расчетная сетка вблизи цилиндра

3. Используемая реализация метода решеточных уравнений Больцмана

Традиционно решеточное уравнение Больцмана (РУБ) записывают в безразмерном виде, при этом для обезразмеривания используются так называемые масштабы решетки: шаг по времени Δt, шаг по пространству Δx и характерная плотность ρ_0 — шаг по времени, пространству и характерная плотность соответственно. Для простоты записи в данном разделе все переменные полагаются безразмерными, если не указано иное.

Решеточное уравнение Больцмана (РУБ) получается путем дискретизации модельных кинетических уравнений Бхатнагара–Гросса–Крука. Дискретизация в фазовом пространстве скоростей выполняется путем разложения функций распределения $f(\mathbf{r}, \mathbf{u}, t)$ по полиномам Эрмита $H^{(n)}$ для заданного набора скоростей $\mathbf{c}_i$ с весами w_i, обеспечивающего точное вычисление первых трех моментов.

Полученное таким образом дискретное по скоростям уравнение Больцмана дискретизируется по времени и пространству путем интегрирования вдоль характеристик

методом трапеций. В результате после замены переменной $f_i \to f_i + \Omega_i/2$, РУБ принимает вид:

$$f_i(\vec{r} + \vec{c}_i, t+1) - f_i(\vec{r}, t) = \Omega_i(\vec{r}, t).$$

Данное уравнение описывает процесс переноса функций распределения, которые были модифицированы оператором столкновений $\Omega_i(\mathbf{r}, t)$, в соседние узлы расчетной сетки вдоль соответствующих дискретных скоростей. Обычно этот процесс эволюции функций распределения реализуется в два шага:

- Шаг столкновения: $f_i^*(\mathbf{r}, t) = f_i(\mathbf{r}, t) + \Omega_i(\mathbf{r}, t)$,
- Шаг переноса: $f_i(\mathbf{r} + \mathbf{c}_i, t+1) = f_i^*(\mathbf{r}, t)$.

Переход к макроскопическим величинам может быть осуществлен путем взятия соответствующих моментов от функции распределения:

$$\rho = \sum f_i; \quad \rho\vec{u} = \sum f_i\vec{c}_i; \quad \Pi = \sum f_i\vec{c}_i\vec{c}_i,$$

здесь Π — второй момент функции распределения, связанный с конвективным слагаемым и тензором напряжений в уравнении баланса импульса.

Следует отметить, что используемая традиционная формулировка метода решеточных уравнений Больцмана предполагает, что за один шаг по времени функция распределения передается в соседний узел сетки. Это приводит к необходимости использовать целочисленные значения проекций дискретных скоростей. В результате при изменении уровня октри-сетки вместе с шагом по пространству меняется и шаг по времени, обеспечивая постоянство дискретной скорости. Данное масштабирование принято называть акустическим [11].

Численное решение РУБ выполняется на видеокарте с использованием CUDA и MPI, где каждому MPI-процессу соответствует свой GPU. Декомпозиция расчетной области между MPI-процессами осуществляется по уровням октри-сетки с блокирующими halo-обменами. Организация данных в программе осуществляется по методу structure of arrays. Данные хранятся в виде одномерных массивов, элементы которых отсортированы так, чтобы обеспечить коалесцированный доступ к памяти на GPU. Все вычисления производились с двойной точностью, при этом оценка влияния точности представления чисел с плавающей точкой на погрешность вычислений не проводилась.

3.1. Модель скоростей

Выбор модели скоростей в РУБ во многом определяет точность описания моделируемых процессов, а также устойчивость метода. В настоящей работе используется наиболее часто применяемый для двумерных задач набор скоростей D2Q9, характеристики которого (набор $\mathbf{c}_i^*$ – вектор скорости и w_i – весовой коэффициент) представлены в таблице 1 [12]. Использование данного набора при расчете слабосжимаемых течений позволяет получить решение системы уравнений Навье–Стокса со вторым порядком точности по пространству.

3.2. Оператор столкновения

В кинетической теории оператор столкновений имеет сложную форму, зависящую от типа столкновений и других факторов, что вынуждает использовать упрощенные модели. В настоящей работе используется наиболее популярная модель оператора столкновений Бхатнагара–Гросса–Крука (BGK) [13]:

$$\Omega_i(\vec{r}, t) = -\frac{f_i(\vec{r}, t) - f_i^{eq}(\vec{r}, t)}{\tau}.$$

Таблица 1. Набор скоростей D2Q9.

i	$\mathbf{c}_i$	w_i
1	$(0,0)$	$4/9$
2-5	$(0,\pm1),(\pm1,0)$	$1/9$
6-9	$(\pm1,\pm1)$	$1/36$

Данный оператор был приближенно получен в предположении, согласно которому функция распределения приходит к своему равновесному состоянию f_i^{eq} за время релаксации τ. Здесь равновесное распределение определяется как:

$$f_i^{eq} = w_i \rho \left(1 + \frac{c_{id}u_d}{c_s^2} + \frac{u_d u_p (c_{id}c_{ip} - c_s^2 \delta_{ap})}{2c_s^4}\right).$$

Определив формулировку метода решеточных уравнений Больцмана, важно убедиться, что она верно воспроизводит рассматриваемые уравнения в макроскопическом пределе, для чего применяется анализ Чепмена–Энскога. Из данного анализа следует, что представленные выше решеточные уравнения соответствуют системе уравнений Навье–Стокса с изотермическим уравнением с точностью $O(M^2)$. Из этого анализа также следует прямая связь между коэффициентом кинематической вязкости и временем релаксации:

$$\nu = c_s^2 \left(\tau - \frac{1}{2}\right)$$

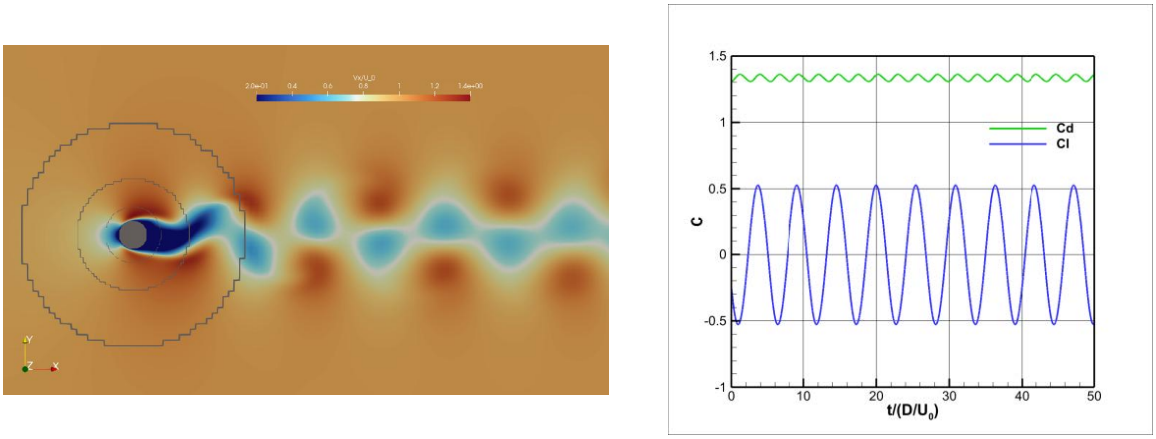
Дополнительно для повышения устойчивости метода применяется рекурсивная регуляризация, выполняемая перед шагом столкновения [14].

4. Результаты расчетов

4.1. Гидродинамический расчет

На рисунке 2 (а) приведено поле продольной компоненты скорости, полученное методом LBM, на котором отчетливо видна дорожка Кармана в следе за цилиндром. Ее образование вызвано попеременным сходом вихрей с разных сторон цилиндра с частотой f, что приводит к осцилляциям коэффициентов сил: $C_d = \frac{F_x}{\rho U_0^2/2}$ и $C_l = \frac{F_y}{\rho U_0^2/2}$, представленным на рис. 2 (б).

Сравнение решения, полученного методом LBM, с известными литературными данными и результатами, полученными в ANSYS Fluent, приведены в таблице 2 ($\langle C_d \rangle$ – средний коэффициент сопротивления, C_{lmax} – амплитуда колебаний коэффициента подъемной силы и $Sh = fD/U_0$ – число Струхаля). Видно хорошее согласование по числу Струхаля и среднему коэффициенту сопротивления, однако для амплитуды колебаний подъемной силы различие более существенное. Видно, что в расчете LBM эта величина завышена примерно на 2%, аналогичное завышение наблюдается в расчете с использованием ANSYS Fluent.



а) б)

Рис. 2. Решение задачи методом LBM:а) Мгновенное поле продольной компоненты скорости. б) Эволю-ция коэффициентов сил, действующих на цилиндр

Таблица 2. Сравнение результатов расчета LBM с известными из литературы данными и решением конечно-объемным методом.

Результат	Sh	$\langle C_d \rangle$	C_{lmax}
DNS	0.183	—	0.52
Kwon & Choi (1996) [5]	—	—	0.515
Williamson (1989) [6]	0.180	—	—
König & Eckelmann (1998) [7]	0.184	—	—
Fluent 2025R2	0.184	1.347	0.525
LBM	0.184	1.334	0.528

4.2. Расчет шума

Оценка точности расчета шума осуществлялась путем сравнения акустического давления (SPL), вычисляемого методом FWH, с результатами из работы [4], полученными непосредственно из решения уравнений Навье–Стокса. Здесь акустическое давление вычисляется по следующей формуле:

$$SPL = 10 \log_{10} \left(\frac{\overline{p'^2}}{p_{SPL}^2} \right),$$

где $\overline{p'^2}$ – среднеквадратичное отклонение пульсаций давления, p_{SPL} – пороговое давление (для воздуха при нормальных условиях – 20 мкПа).

Результаты расчетов представлены в виде полярной диаграммы SPL (рис. 3). Видно, что результат, полученный методом LBM, практически совпадает с результатом ANSYS Fluent, а отличие от DNS [6] (до 2 дБ) связано, скорее всего, с использованием непроницаемой FWH поверхности (таблица 3).

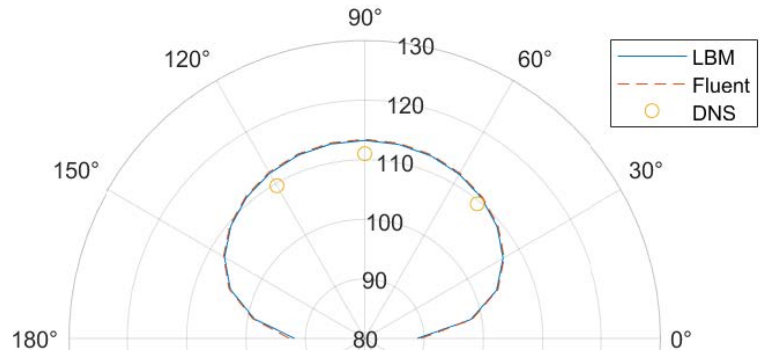


Рис. 3. SPL в точках наблюдателя на окружности $R = 100D$.

Таблица 3. SPL в точках наблюдателя.

θ, °	SPL, дБ DNS	SPL, дБ Fluent	SPL, дБ LBM
50	109.5	110.9	110.7
90	111.0	113.4	113.3
120	109.6	112.2	112.0

4.3. GPU производительность

Расчеты проводились на двух видеокартах GeForce RTX 5060 Laptop (26 мультипроцессоров, 3328 CUDA-ядер, 8 ГБ GDDR7 с пропускной способностью 384 ГБ/с и шириной памяти в 128 бит) и Tesla K40m (15 мультипроцессоров, 2880 CUDA-ядер, 12 ГБ GDDR5 с пропускной способностью 288.4 ГБ/с и шириной памяти в 384 бит). Во всех расчетах использовалась одномерная сетка из блоков по 32 нити в каждом, во Fluent данные настройки задаются автоматически.

Скорость работы GPU-реализации метода LBM существенно превосходит скорость работы коммерческого кода ANSYS Fluent 2025R2, использующего метод конечных объемов. Из таблицы 4, в которой приведено время расчета одного периода колебаний подъемной силы, видно, что предлагаемая реализация LBM обеспечивает ускорение более чем на 3 порядка. Вероятнее всего такое различие связано с малым размером задачи и в случае больших задач оно может стать меньше.

Таблица 4. Сравнение скорости решения на Fluent 2025R2 и LBM на GPU (GeForce RTX 5060 Laptop).

Fluent 2025R2, с	LBM, с
1827.1	1.1

В таблицах 5 и 6 приведены результаты тестирования на слабое и сильное масштабирование разработанного GPU-кода соответственно. Количество временных шагов во всех расчетах задавалось одинаковым. Полученные результаты демонстрируют незначительное изменение времени решения при пропорциональном увеличении размерности задачи и вычислительных ресурсов, что подтверждает эффективное слабое

масштабирование. В случае сильного масштабирования наблюдается практически линейное ускорение.

Таблица 5. Тестирование кода на слабое масштабирование на Tesla K40m.

Размер сетки, млн	Количество процессов	GPU память, MiB	Время, с
4.5	1	722	1304
9.0	2	1380	1418
18.0	4	2660	1211
36.0	8	5230	1184

Таблица 6. Тестирование кода на сильное масштабирование на Tesla K40m.

Размер сетки, млн	Количество процессов	GPU память, MiB	Время, с
36.0	1	5230	8656
36.0	2	5230	4423
36.0	4	5230	2182
36.0	8	5230	1184

Заключение

В ходе работы была разработана и валидирована реализация метода решеточных уравнений Больцмана для задач аэроакустики с использованием графических процессоров. Проведенные расчеты показали, что предложенный подход способен с высокой точностью воспроизводить основные гидродинамические характеристики течения, включая частоту срыва вихрей, коэффициенты сил и число Струхаля, демонстрируя хорошее согласие как с известными из литературы данными, так и с результатами, полученными в ANSYS Fluent.

Анализ акустических характеристик показал, что метод хорошо совпадает с результатами ANSYS Fluent для непроницаемой поверхности. Имеющееся отклонение от литературных данных в 2 дБ скорее всего связано с использованием непроницаемой поверхности.

Ключевым результатом работы является демонстрация высокой вычислительной эффективности разработанной GPU-реализации. Получено ускорение более чем на три порядка по сравнению с традиционным конечно-объемным методом на GPU, что делает предложенный подход особенно перспективным для решения ресурсоемких задач аэроакустики.

Таким образом, разработанная реализация метода РУБ на GPU представляет собой эффективный инструмент для моделирования аэродинамического шума, сочетая в себе высокую производительность и приемлемую точность.

Работа выполнена в рамках госзадания Минобрнауки России № 075-00558-26-00 от 12.01.2026 «Развитие теории модельно-ориентированного проектирования для задач создания трансформных робототехнических систем с использованием методов вычислительной механики и гидрогазодинамики» (FNRG-2025-0005 1024042600099-7-2.2.2;1.2.1) с использованием ресурсов кластера «Торнадо» суперкомпьютерного центра СПбГПУ.

Список литературы

[1] Alexandre Suss, Ivan Mary, Thomas Le Garrec, Simon Marié. Comprehensive comparison between the lattice Boltzmann and Navier–Stokes methods for aerodynamic and aeroacoustic applications // Computers & Fluids. – 2023. – Vol. 257. – DOI: 10.1016/j.compfluid.2023.105881.

[2] Farassat F., Myers M. K. Extension of Kirchhoff's Formula to Radiation from Moving Surfaces // Journal of Sound and Vibration. - 1988. - Vol. 123, No. 3. - P.451-460. - DOI: 10.1016/S0022-460X(88)80162-7.

[3] Ffowcs Williams J. E., Hawkings D. L. Sound Generated by Turbulence and Surfaces in Arbitrary Motion // Philosophical Transactions of the Royal Society. - 1961. - Vol. A264, No. 1151. - P.321-342. - DOI: 10.1098/rsta.1969.0031.

[4] O. Inoue and N. Hatakeyama. Sound Generation by a Two-Dimensional Circular Cylinder in a Uniform Flow. J. Fluid Mech., 471, 2002, p.285 – 314. – DOI: 10.1017/S002211200200212.

[5] Kwon, K. & Choi, H. 1996 Control of laminar vortex shedding behind a circular cylinder using splitter plates. Phys. Fluids 8, 479-486. - DOI: 10.1063/1.868801.

[6] Williamson, C. H. K. 1989 Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers. J. Fluid Mech. 206, 579-627. - DOI: 10.1017/S0022112089002429.

[7] Fey, U., König, M. & Eckelmann, H. 1998 A new Strouhal-Reynolds-number relationship for the circular cylinder in the range $47 < Re < 105$. Phys. Fluids 10, 1547-1549. - DOI: 10.1063/1.869675.

[8] Hirsch C. Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows. - Chichester: John Wiley & Sons, 1991. - 672 p.

[9] A. Najafi-Yazdi, L. Mongeau. An absorbing boundary condition for the lattice Boltzmann method based on the perfectly matched layer // Computers & Fluids. - 2012. - Vol. 68. - P. 203-218. - DOI: 10.1016/j.compfluid.2012.07.017.

[10] Bouzidi, M. Firdaouss, P. Lallemand, Momentum transfer of a Boltzmann-lattice fluid with boundaries, Phys. Fluids (1994-present) 13 (11) (2001) 3452-3459. - DOI: 10.1063/1.1399290.

[11] M. Schönherr, K. Kucher, M. Geier, M. Stiebler, S. Freudiger, M. Krafczyk, Multi-thread implementations of the lattice Boltzmann method on nonuniform grids for CPUs and GPUs, Comput. Math. Appl. 61 (12) (2011) 3730-3743. - DOI: 10.1016/j.camwa.2011.04.012.

[12] Krüger T., Kusumaatmaja H., Kuzmin A., Shardt O., Silva G., Viggen E. The Lattice Boltzmann Method. Principles and Practice. - Cham: Springer, 2017. - 403 p. - DOI: 10.1007/978-3-319-44649-3.

[13] Коган М. Н. Динамика разреженного газа. - М.: Наука, 1967. - 440 с.

[14] Malaspinas O., Chopard C. Increasing stability and accuracy of the lattice Boltzmann scheme: recursivity and regularization // arXiv: Fluid Dynamics. - 2015. - DOI: 10.48550/arXiv.1505.06900.

Оптимизация значений гиперпараметров нейросетевой модели классификации и сегментации сигналов ЭКГ*

Д.А. Ануфриев, Е.Н. Варсеева, А.С. Зайцев, Е.Г. Голдинова, К.С. Егоров, М.А. Усова, Д.А. Карчков, И.Г. Лебедев, К.А. Баркалов

Нижегородский государственный университет им. Н.И. Лобачевского

В статье рассматривается задача оптимизации гиперпараметров нейросетевой модели классификации и сегментации сигналов электрокардиограммы (ЭКГ) с использованием программной системы Globalizer. Объектом исследования являются архитектуры нейросетей MobileNetV3 Small и U-Net 1D, направленных на решение задачи классификации и сегментации соответственно, адаптированные для обработки 10-секундных фрагментов ЭКГ, записанных с частотой дискретизации 500 Гц. Задача классификации заключается в определении ведущего ритма сердечной деятельности: синусовый ритм, синусовые вариации, другие типы ритма. Задача сегментации направлена на локализацию в сигнале базовых элементов кардиоцикла, соответствующих сокращению предсердий сердца, желудочков и восстановлению миокарда. Для решения задачи настройки гиперпараметров применен информационно-статистический алгоритм глобального поиска, реализованный в системе Globalizer и принадлежащий классу методов липшицевой глобальной оптимизации. Представлены результаты сравнения эффективности Globalizer с известными фреймворками Optuna и HyperOpt, демонстрирующие сопоставимое качество оптимизации при существенном сокращении времени настройки. Полученные результаты подтверждают перспективность применения детерминированных методов липшицевой оптимизации для точной настройки гиперпараметров глубоких нейронных сетей в медицинских диагностических задачах.

Ключевые слова: глобальная оптимизация, многоэкстремальные функции, параллельные вычисления, нейронные сети, искусственный интеллект

1. Введение

Электрокардиография (ЭКГ) является одним из наиболее распространенных и информативных методов диагностики сердечно-сосудистых заболеваний. Ежегодно в мире регистрируются миллионы электрокардиограмм, и автоматический анализ этих данных представляет собой актуальную задачу медицинской информатики. В последние годы для классификации ЭКГ-сигналов активно применяются методы глубокого обучения, в частности сверточные нейронные сети, которые демонстрируют высокую точность при решении задач распознавания ритмов и выявления аномалий [1, 2, 3, 4]. Однако эффективность таких моделей существенно зависит от правильного выбора гиперпараметров — значений, определяющих архитектуру сети и процесс ее обучения.

Задача оптимизации гиперпараметров моделей ИИ заключается в поиске такой комбинации гиперпараметров, которая обеспечивает наилучшее значение целевой метрики качества модели. С математической точки зрения эта задача относится

*Работа выполнена при поддержке программы стратегического академического лидерства «Приоритет 2030»

к классу задач глобальной оптимизации с ограничениями, причем целевая функция, как правило, является многоэкстремальной, не имеет аналитического описания (представляет собой «черный ящик»), а ее однократное вычисление требует значительных вычислительных ресурсов, связанных с обучением нейронной сети. Указанные особенности делают задачу оптимизации гиперпараметров чрезвычайно сложной и требуют развитие специализированных методов оптимизации.

Среди наиболее известных фреймворков для автоматической настройки гиперпараметров можно выделить Optuna [5] и HyperOpt [6], которые реализуют байесовские методы оптимизации, случайный поиск, а также эволюционные и генетические алгоритмы [7, 8, 9]. Эти подходы являются стохастическими по своей природе, что приводит к вариабельности результатов при многократных запусках и не гарантирует нахождения глобального оптимума за ограниченное число итераций. Альтернативой стохастическим методам выступают детерминированные алгоритмы глобальной оптимизации [10, 11, 12], одним из которых является информационно-статистический алгоритм глобального поиска, разработанный в Нижегородском государственном университете им. Н.И. Лобачевского [13, 14]. Данный алгоритм основан на предположении липшицевости целевой функции и использует редукцию размерности с помощью кривых Пеано для сведения многомерной задачи к одномерной. Важным преимуществом этого подхода является его детерминированность, обеспечивающая воспроизводимость результатов и гарантированную сходимость к глобальному минимуму при выполнении определенных условий [13].

Программная реализация описанных методов выполнена в виде программной системы Globalizer (доступна в открытом репозитории `https://github.com/OptimLLab/Globalizer`), которая предназначена для решения задач выбора оптимальных параметров сложных систем, включая модели искусственного интеллекта и машинного обучения. Система поддерживает работу с частично целочисленными переменными, невыпуклыми ограничениями, а также предоставляет возможности для параллельных вычислений с использованием технологий OpenMP, MPI и CUDA.

Настоящая статья посвящена применению программной системы Globalizer для оптимизации гиперпараметров модели классификации и сегментации сигналов ЭКГ. В качестве базовых архитектуры классифицирующей сети выбрана адаптированная версия MobileNetV3 Small — легковесная сверточная нейронная сеть. Модель предназначена для классификации 10-секундных фрагментов ЭКГ, зарегистрированных в двух отведениях с частотой дискретизации 500 Гц. Задача классификации заключается в определении ведущего ритма сердечной деятельности: синусовый ритм, синусовые вариации или другие типы ритма. Варьируемыми гиперпараметрами являются число классифицирующих нейронов в выходном слое (в диапазоне от 80 до 200) и вероятность отключения нейронов (Dropout, в диапазоне от 0.0 до 1.0). Для решения задачи сегментации сигнала ЭКГ с целью локализации моментов активации предсердий, желудочков и восстановления миокарда используется архитектура U-Net, адаптированная для анализа одномерных сигналов. В угоду потоковой обработки ЭКГ средствами данной нейросетевой модели решено сформировать обучающую и тестовую выборку из участков 12-ти канальных сигналов, длительностью 1 секунда, или 500 отсчетов, что гарантирует попадание хотя бы одного кардиоцикла полностью или частично. Варьируемыми гиперпараметрами являются количество уровней архитектуры (значение глубины изменяется от 2 до 5), что позволяет параметрически изменять архитектуру сети и, как следствие, количество обучаемых параметров, а также вероятность отключения нейронов, используемая для регуляризации и предотвращения переобучения.

В качестве исходных данных для обучения и тестирования используется расширенный датасет LUDB [15], который содержит записи ЭКГ с метками как для задачи сегментации, так и для задачи классификации. LUDB является репрезентативной базой данных, широко применяемой в исследованиях по автоматическому анализу электрокардиограмм.

Статья имеет следующую структуру. В разделе 2 приводится постановка задачи оптимизации гиперпараметров и формальное описание используемой модели классификации ЭКГ. Раздел 3 содержит описание параллельного алгоритма глобального поиска. В разделе 4 описывается методика проведения вычислительных экспериментов, включая характеристики использованного вычислительного оборудования, параметры запуска алгоритмов и критерии сравнения. Раздел 5 посвящен анализу полученных результатов. В заключении сформулированы основные выводы и намечены направления дальнейших исследований.

2. Постановка задачи

Разработка диагностических алгоритмов, позволяющих определять наличие конкретной нозологии в сигнале ЭКГ, может базироваться на одной из фундаментальных задачах машинного обучения: задача классификации или задачи сегментации.

2.1. Задача классификации

Задача классификации в контексте машинного обучения сводится к отнесению исследуемого объекта к одному из заранее известных классов или категорий. При разработке нейросетевой модели, предпочтение отдано архитектуре, решающую задачу мультиклассовой классификации, при которой каждый сигнал относится только к одному классу. В основу задачи классификации ЭКГ сигналов заложено разделение всех сигналов по типу ритма:

- Синусовый ритм: нормальный ритм и его физиологические вариации;
- Синусовые вариации: синусовая аритмия, брадикардия, тахикардия и др.;
- Другой ритм: предсердный, атриовентрикулярный, идиовентрикулярный и др.

Таким образом, поставлена задача разработки и обучения нейросетевой модели, адаптированной под классификацию сигналов ЭКГ, при условии варьирования параметров модели, таких как:

- Число классифицирующих нейронов: в диапазоне от 80 до 200;
- Вероятность отключения нейронов (Dropout): в диапазоне от 0.0 до 1.0.

Важно отметить, что полученная модель в дальнейшем должна быть интегрирована в SoC микроконтроллер, для локального выделения нозологий на устройстве сбора данных без привлечения персональных компьютеров и серверных мощностей.

2.1.1. Описание данных

Обучающая и тестовая выборка сформирована из сигналов открытого датасета Lobachevsky University Electrocardiography Database, содержащего 200 аннотированных сигналов ЭКГ [15]. Регистрация сигнала проводилась в 12-ти стандартных отведениях с частотой дискретизации 500 Гц в течении 10 секунд. Пример сигнала из датасета представлен на рис. 1.

Согласно описанию, записи в датасете, в зависимости от ритма, распределены следующим образом:

- Синусовый ритм: 143;

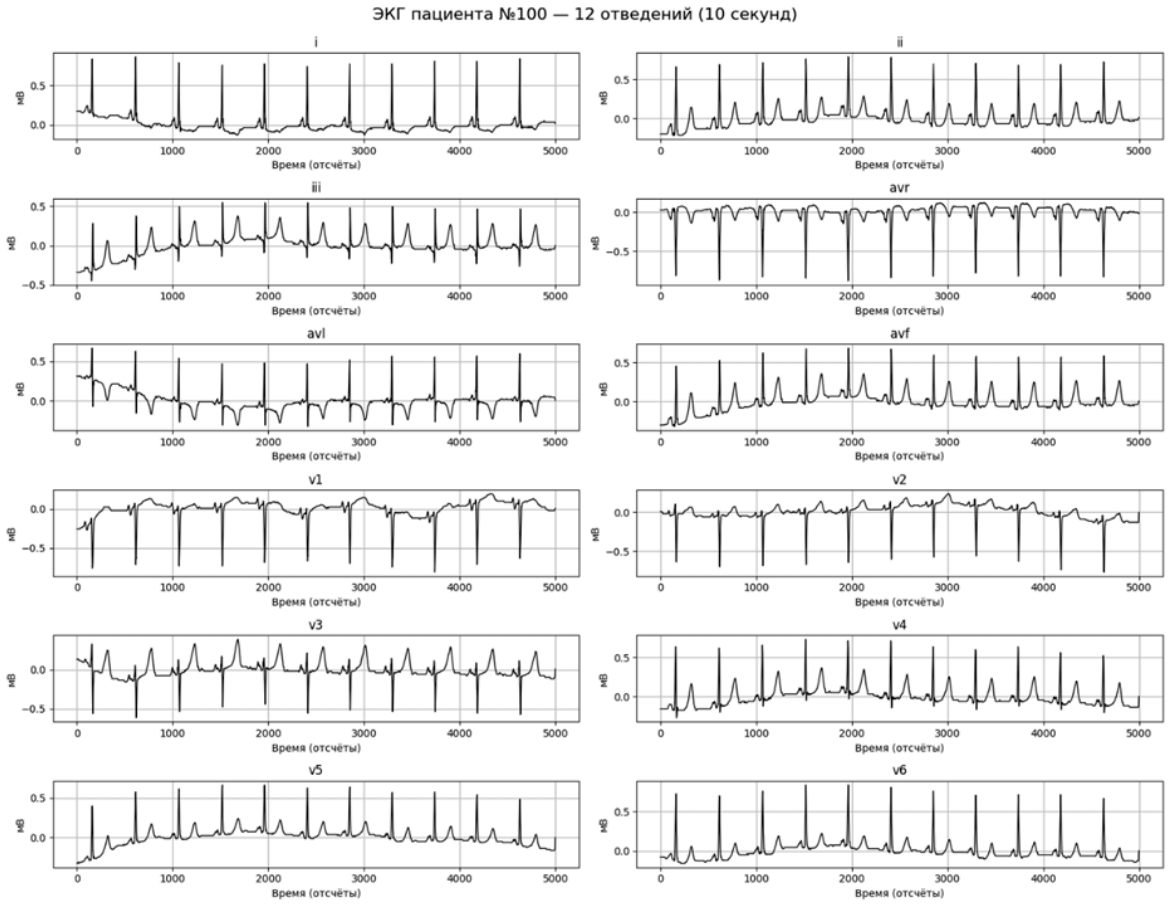


Рис. 1. Образец ЭКГ сигнала в базе данных Lobachevsky University Electrocardiography Database, с отдельным отображением профиля каждого отведения

- Синусовая тахикардия: 4;
- Синусовая брадикардия: 25;
- Синусовая аритмия: 8;
- Нерегулярный синусовый ритм: 2;
- Аномальный ритм 19.

С целью уменьшения размера итоговой модели при формировании обучающей и тестовой выборки выбраны первое (I) и второе (II) отведение. Кроме того, с медицинской точки зрения выбранные типы ритма хорошо представлены на графиках данных отведений, поэтому исключение остальных отведений из выборки не должно отрицательно сказаться на качестве классификации.

Таким образом, при разделении датасета на 80% обучающей выборки и 20% тестовой выборки мы имеем соответственно 160 и 40 примеров размером (2 x 5000).

2.1.2. Архитектура нейронной сети

В силу ограниченного объема оперативной памяти на устройстве инференса базовой архитектурой выбрана MobileNetV3 Small, адаптированная под входные данные в формате (batch, 2, 5000). Графически архитектура представлена на рис. 2.

Применение блоков с инвертированными остаточными связями (inverted residual) позволяет сети не разрастаться, сохраняя малое число параметров и компактный размер, что важно при работе на SoC устройствах. Слой усреднения по глобальному

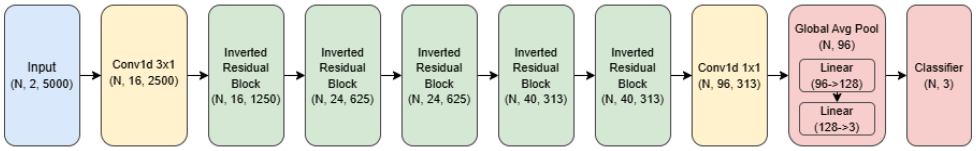


Рис. 2. Графическое представление архитектуры MobileNetV3 Small, адаптированной под классификацию сигнала ЭКГ по двум отведениям

окну (Global Average Pooling) позволяет агрегировать временную информацию, делая выход сети инвариантным к локальным сдвигам, что позволяет концентрироваться на наличии паттерна конкретной нозологии в сигнале ЭКГ, а не на его локализации в сигнале.

Для обучения модели использовался фреймворк машинного обучения PyTorch. Данный инструмент поддерживает автоматическое дифференцирование, адаптивен для разработки модифицированных архитектур с возможностью контроля их параметров, но главное — позволяет использовать мощности GPU карт при обучении. Описанные особенности упрощают возможность оптимизации гиперпараметров нейросетевой модели при ее обучении.

Оптимизатор Adam (адаптивный метод оптимизации на основе градиентного спуска) хорошо справляется с разреженными градиентами, встречающихся в свертках, а также обеспечивает быструю сходимость на первых эпохах обучения.

В качестве функции потерь выбрана кросс-энтропийная функция (CrossEntropy Loss), представляющая собой комбинацию логарифмического SoftMax (LogSoftmax) и отрицательного логарифмического правдоподобия (Negative Log Likelihood), штрафующая модель, если вероятность правильного класса низкая. Данная функция потерь считается лучшей практикой при многоклассовой классификации в силу отсутствия проблемы взрыва или затухания градиента при сохранении интерпретируемости.

Параметр скорости обучения (Learning Rate) установлен в 1e-3, при этом оптимизатор Adam адаптивно меняет скорость для каждого веса.

Выбор макро-усреднённой F1-меры (F1-macro) в качестве основной метрики качества обусловлен необходимостью корректной оценки модели в условиях выраженного дисбаланса, характерного для реальных клинических данных. В отличие от точности (accuracy), которая при доминировании одного класса может создавать иллюзию высокой эффективности, F1-macro представляет собой гармоническое среднее между полнотой (recall) и точностью (precision), что позволяет одновременно учитывать как пропуски целевых событий (ложноотрицательные ответы), так и ложные срабатывания (ложноположительные ответы). Использование макро-усреднённой F1-меры позволяет нивелировать влияние доминирующего класса и дает равнозначную оценку вклада каждого типа ритма, что является разумным компромиссом на этапе разработки и сравнительного анализа методов оптимизации гиперпараметров.

2.2. Задача сегментации

Сегментацией в кардиологии называют выделение в ЭКГ сигнале всех элементов кардиоциклов, соответствующих сокращению предсердий, желудочков и восстановлению миокарда. Точная локализация элементов кардиоциклов позволит однозначно определить нозологию, описанную в медицинской литературе, что позволит разработчику написать аналитический алгоритм.

Сегментация сигнала ЭКГ является не тривиальной задачей из-за существования низкоамплитудных зубцов, описывающих работу сердечной мышцы, в следствии чего интерпретация сигнала затрудняется и приводит к ошибочной диагностике.

Наиболее эффективный способ сегментации ЭКГ сигналов состоит в применении нейронных сетей, изначально специализированных на сегментации изображений. Однако обучение таких моделей осложнено наличием гиперпараметров, для которых требуется точная настройка.

Таким образом, необходимо разработать нейросетевую модель, для решения задачи сегментации (выделения компонентов кардиоциклов) сигнала ЭКГ, и выбрать оптимальные значения ее гиперпараметров.

2.2.1. Описание данных

Исходные данные представляют собой набор json файлов, где каждый сигнал описывается двумя файлами:

- Файл с мета информацией о записи;
- Файл с сигналом и временными метками комплексов.

При формировании датасета использовалось 680 сигналов ЭКГ, локализацию сегментов в которых осуществляли врачи функционалисты, разрабатывающих первую версию базы данных Lobachevsky University Electrocardiography Database [15]. Для обучения моделей сегментации каждая запись была преобразована в набор окон фиксированной длины с покомпонентной разметкой.

Основные шаг в подготовке датасета:

1. Чтение сигнала. Для каждой записи считывается сигнал размера [число сэмплов окна, число входных каналов];
2. Построение меток по каждому каналу. Для каждого канала отдельно считывается соответствующий файл аннотаций;
3. Полученные файлы аннотаций переводятся в покомпонентный вектор меток числа сэмплов окна. Для классов поддерживается общий словарь: фон = 0, остальные классы нумеруются с 1;
4. Для удаления неразмеченных участков применяется обрезка отсчетов с начала и конца. Если после обрезки длина записи меньше размера окна, запись исключается из рассмотрения.
5. Нарезка на окна с перекрытием. Сигнал разбивается на окна длиной W_s с заданным перекрытием.

Длина исходной записи составляет порядка 5000 отсчетов. Перед нарезкой выполняется обрезка краев в 450 отсчетов с начала и конца записи, что позволяет исключить неразмеченные участки и оставить для обучения центральную часть сигнала. Далее сигнал разбивается на окна фиксированной длины по 500 сэмплов. Данный размер окна подбирался так, чтобы в одном примере целиком присутствовал по меньшей мере один кардиоцикл вместе с его морфологическими компонентами, что важно для корректной сегментации сегментов зубцов P, QRS, T с учетом контекста. Для повышения устойчивости разметки на границах окон применяется перекрытие 64 отсчета. Пример исходной разметки выбранного комплекса представлен на рис. 3. Здесь черная кривая соответствует графику ЭКГ, а красная кусочно-постоянная линия отражает принадлежность отсчета сигнала одному из целевых классов: фон — 0, комплекс QRS — 1, T волна — 2, P волна — 3.

Наличие перекрытия уменьшает риск потери информативных фрагментов в случае, когда границы сегментов или короткие события (например, часть комплекса QRS) попадают в область разрыва между соседними окнами: один и тот же временной

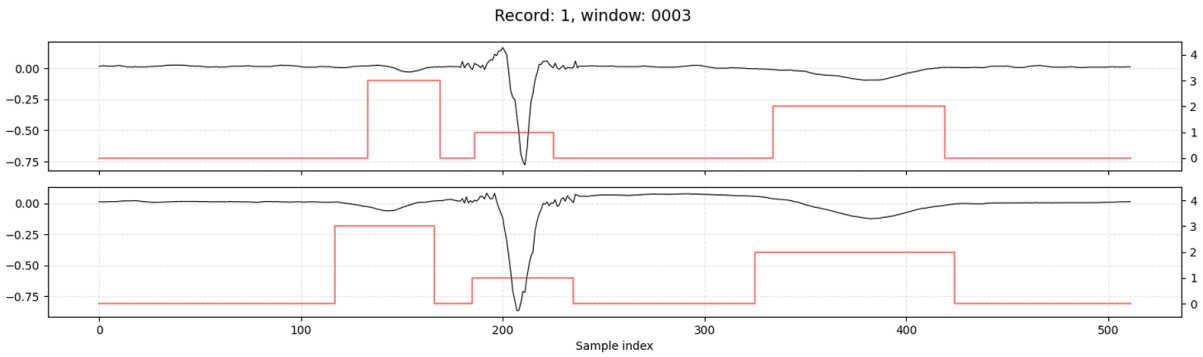


Рис. 3. Пример участка сигнала ЭКГ с разметкой элементов кардиоцикла

участок будет присутствовать как минимум в двух соседних примерах. Это повышает стабильность обучения и снижает чувствительность модели к «краевым» эффектам при сегментации.

2.2.2. Архитектура нейронной сети

С помощью U-Net 1D архитектуры решается задача покомпонентной разметки (segmentation / sequence labeling) одномерного сигнала. Модель получает на вход тензор ЭКГ-сигнала

$$x \in \mathbb{R}^{B \times C_{\text{in}} \times T},$$

где B — размер пакета (batch size); C_{in} — число входных каналов (например ЭКГ отведений); T — размер окна в отсчетах сигнала.

На выходе модель формирует логиты (logits) классов

$$\text{logits} \in \mathbb{R}^{B \times C_{\text{cls}} \times T},$$

где C_{cls} — число классов сегментации (*фон / P / QRS / T*). Впоследствии к логитам применяется softmax для получения вероятностей классов по каждому временному отсчету t. Во всех реализациях используются одномерные свертки с ядром $k = 3$ и дополнением (padding) равным единицей, а также одномерная пакетная нормализация и нелинейность выпрямителя (ReLU).

Согласно принципу формирования U-Net архитектур имеем, что энкодер уменьшает временное разрешение (downsampling) и увеличивает число каналов, а декодер восстанавливает временное разрешение (upsampling) и объединяет признаки с соответствующих уровней энкодера через skip-соединения.

Downsampling. Down-блоки используют оператор

$$\texttt{MaxPool1d}(2),$$

где значением аргумента является размерность окна субдискретизации (pooling). В результате длина последовательности уменьшается в 2 раза: $T \to T/2$. После операции субдискретизации выполняется сверточная обработка признаков.

Upsampling, sequence padding. Up-блоки используют транспонированную свертку

$$\texttt{ConvTranspose1d}(k{=}2, s{=}2),$$

где k — размер ядра, s — шаг (stride). Размер данных увеличивается в 2 раза: $T \to 2T$. Из-за возможных несовпадений пространственной размерности между тензором,

полученным в результате повышения дискретизации, и тензором пропускающего соединения в реализации архитектуры применяется операция симметричного дополнения (симметричного паддинга), для обеспечения корректной конкатенации

$$x = \texttt{cat}\big[x_{\text{skip}},\, x_{\text{up}}\big]$$

по оси отведений. Здесь x_{up} — признаки, полученные в результате повышения дискретизации в декодере, x_{skip} — признаки с соответствующего уровня энкодера (до downsampling), а $\texttt{cat}[\cdot,\cdot]$ — конкатенация по оси каналов ЭКГ сигнала.

2.2.3. Структура блоков одномерной архитектуры U-Net

DoubleConv1D. Базовый блок энкодера и декодера задается как:

$$\text{DoubleConv}(x) = \text{ReLU}\big(\text{BN}(\text{Conv}_3(x))\big) \to \text{ReLU}\big(\text{BN}(\text{Conv}_3(\cdot))\big),$$

где $\text{Conv}_3(\cdot)$ — одномерная свертка с ядром размера 3; $\text{BN}(\cdot)$ — пакетная нормализация; $\text{ReLU}(\cdot)$ — функция активации. Стрелка обозначает композицию слоев на данном архитектурном уровне.

Down1D. Down-блок:

$$\text{Down}(x) = \text{DoubleConv}(\text{MaxPool}(x)),$$

где операция $MaxPool(\cdot)$ реализована как одномерное субдискретизирующее отображение, обеспечивающее двукратное сокращение временно́й протяженности сигнала.

Up1D. Up-блок:

$$\text{Up}(x_{\text{low}}, x_{\text{skip}}) = \text{DoubleConv}\left(\texttt{cat}\big[x_{\text{skip}},\, \text{UpConv}(x_{\text{low}})\big]\right),$$

где x_{low} — признаки с более низким временным разрешением; x_{skip} — признаки пропускающего соединения с соответствующего уровня энкодера; $\text{UpConv}(\cdot)$ — операция повышения дискретизации (upsampling); $\texttt{cat}[\cdot,\cdot]$ — конкатенация по каналам.

Пусть F_0 — число базовых фильтров. Тогда число каналов по уровням U-Net: F_0, $2F_0$, $4F_0$, $8F_0$, $16F_0$.

Encoder.

$$\begin{aligned}
x_1 &= \text{DoubleConv}(x) \in \mathbb{R}^{B\times F_0\times T},\\
x_2 &= \text{Down}(x_1) \in \mathbb{R}^{B\times 2F_0\times T/2},\\
x_3 &= \text{Down}(x_2) \in \mathbb{R}^{B\times 4F_0\times T/4},\\
x_4 &= \text{Down}(x_3) \in \mathbb{R}^{B\times 8F_0\times T/8},\\
x_5 &= \text{Down}(x_4) \in \mathbb{R}^{B\times 16F_0\times T/16}.
\end{aligned}$$

x_i — карты признаков на i-м уровне энкодера. Деление $T/2, T/4, \ldots$ отражает редукцию временно́й протяженности сигнала в результате операции субдискретизации (pooling).

Bottleneck.

$$x_b = \text{DoubleConv}(x_5) \in \mathbb{R}^{B\times 16F_0\times T/16},$$

где x_b — признаки на нижнем уровне архитектуры U-Net (называемые bottleneck), содержащие наиболее контекстную информацию.

Decoder.

$$x = \mathrm{Up}(x_b, x_4) \in \mathbb{R}^{B\times 8F_0\times T/8},$$
$$x = \mathrm{Up}(x, x_3) \in \mathbb{R}^{B\times 4F_0\times T/4},$$
$$x = \mathrm{Up}(x, x_2) \in \mathbb{R}^{B\times 2F_0\times T/2},$$
$$x = \mathrm{Up}(x, x_1) \in \mathbb{R}^{B\times F_0\times T}.$$

В декодирующей части архитектуры на каждом уровне последовательно выполняются: повышение дискретизации (upsampling), конкатенация с признаками пропускающего соединения (skip-connection) и последующая сверточная обработка.

Выход сети. На финальном этап процедуры сегментации выполняется свертка 1×1:

$$\mathrm{logits} = \mathrm{Conv}_1(x) \in \mathbb{R}^{B\times C_{\mathrm{cls}}\times T},$$

где $\mathrm{Conv}_1(\cdot)$ — одномерная свертка с размером ядра, равным единице, осуществляющая линейную классификацию для каждого временного отсчета t.

В представленной архитектуре реализуется иерархическое извлечение многомасштабных признаков посредством комбинации нисходящих и восходящих путей. На глубоких уровнях энкодера, благодаря последовательному уменьшению пространственного разрешения и расширению рецептивного поля, формируются признаки, обобщающие контекст на протяженных временных интервалах. В свою очередь, пропускающие соединения (skip connections), передающие активации с соответствующих уровней энкодера на соответствующие им уровни декодера, обеспечивают доступность пространственно-временной информации, что необходимо для локализации границ сегментов кардиоцикла.

3. Параллельный алгоритм для решения задач глобальной оптимизации

Задача настройки гиперпараметров модели машинного обучения с математической точки зрения является задачей поиска глобального оптимума целевой метрики качества. В общем виде задача такого типа может быть записана следующим образом:

$$\varphi(y^*) = \min\{\varphi(y) : y \in D\}, \tag{1}$$
$$D = \left\{y \in \mathbb{R}^N : a_i \le y_i \le b_i,\ a_i, b_i \in \mathbb{R},\ 1 \le i \le N\right\}, \tag{2}$$

где целевая функция $\varphi(y)$ является многоэкстремальной, задана в вида «черный ящик» (т.е. не имеет аналитического описания). Как и во многих других подходах к разработке методов глобальной оптимизации [16, 17, 18, 19] будем предполагать, что $\varphi(y)$ в области поиска удовлетворяет условию Липшица с априори неизвестной константой $L,\ L < \infty$,, т.е.

$$|\varphi(y_1) - \varphi(y_2)| \le L\,\|y_1 - y_2\|,\ y_1, y_2 \in D.$$

Для решения многомерных задач глобальной оптимизации в системе Globalizer используется редукция размерности на основе кривых Пеано [20, 13]. Кривая Пеано $y(x)$ непрерывно и однозначно отображает единичный отрезок $[0, 1]$ на N-мерный гиперкуб D:

$$D = \left\{y \in \mathbb{R}^N : -2^{-1} \le y_i \le 2^{-1}, 1 \le i \le N\right\} = \{y(x) : 0 \le x \le 1\}.$$

С помощью такого отображения исходная многомерная задача (1) сводится к одномерной задаче

$$\varphi(y^*) = \varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1]\},$$

где функция $\varphi(y(x))$ удовлетворяет равномерному условию Гёльдера

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \le H\,|x_1 - x_2|^{1/N}$$

с константой Гёльдера H, связанной с константой Липшица L соотношением $H = 2L\sqrt{N+3}$.

Алгоритм глобального поиска (АГП) строит последовательность точек x^i, в которых проводятся поисковые испытания (здесь и далее будем называть процесс вычисления значения функции $f(x) = \varphi(y(x))$ *испытанием*, а пару $\{x, z = f(x) = \varphi(y(x))\}$ — результатом испытания). Пусть после выполнения k испытаний накоплена поисковая информация Ω_k, представленная в виде набора

$$\Omega_k = \{(x_i, y_i = y(x_i), z_i = \varphi(y_i)) : 1 \le i \le k\},$$

где точки упорядочены по возрастанию x: $x_1 < x_2 < \ldots < x_k$.

Для выбора точки следующего испытания для каждого интервала (x_{i-1}, x_i) вычисляется характеристика

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{(r\mu)^2\Delta_i} - 2\frac{z_{i-1} + z_i}{r\mu}, \tag{3}$$

где $\Delta_i = (x_i - x_{i-1})^{1/N}$, $r > 1$ — параметр алгоритма, μ — оценка константы Гёльдера, вычисляемая по формуле

$$\mu = \max_{1<i\le k} \frac{|z_i - z_{i-1}|}{\Delta_i},$$

причем если $\mu = 0$, то полагается $\mu = 1$.

Новое испытание проводится в точке x^{k+1}, принадлежащей интервалу с максимальной характеристикой $R(t) = \max\{R(i)\}$. Если правой точке этого интервала соответствует номер t в множестве Ω_k, то

$$x^{k+1} = \frac{x_t + x_{t-1}}{2} - \frac{\operatorname{sign}(z_t - z_{t-1})}{2r}\left(\frac{|z_t - z_{t-1}|}{\mu}\right)^N. \tag{4}$$

Алгоритм завершает работу при выполнении условия $\Delta_t < \epsilon$, где $\epsilon > 0$ — заданная точность, либо при достижении максимального числа испытаний $K_{\max}$.

Для ускорения процесса оптимизации в системе Globalizer реализованы параллельные версии алгоритма. Основная идея распараллеливания заключается в одновременном проведении нескольких испытаний на разных вычислительных устройствах (процессах или потоках) [21, 22, 23]. Вычислительная схема реализует принцип «мастер-рабочие», мастер-процесс выполняет основные шаги алгоритма глобального поиска: накапливает поисковую информацию, вычисляет оценку константы Гёльдера, определяет новые пробные точки и распределяет их между рабочими процессами. Рабочие процессы получают точки испытаний от главного процесса, вычисляют значения целевой функции в этих точках и отправляют результаты мастеру.

В синхронной схеме вычислений переход к следующей итерации осуществляется только после завершения всех p испытаний текущей итерации. Достоинством синхронной схемы является простой реализации, однако при значительном разбросе времени

вычисления целевой функции в разных точках возможны простои вычислительных ресурсов.

Для устранения недостатков синхронной схемы разработана асинхронная версия параллельного алгоритма. Ее ключевая идея заключается в том, чтобы не ожидать завершения всех испытаний текущей итерации, а отправлять новую точку на вычисление сразу после получения результата от любого рабочего процесса.

Пусть в нашем распоряжении имеется $p+1$ вычислительных процессов: один мастер (процесс 0) и p рабочих процессов. В начале поиска мастер инициирует параллельное выполнение p испытаний в различных точках области поиска (начальные точки распределяются равномерно или случайным образом).

Предположим, что выполнено $k \geq 0$ испытаний и рабочие процессы выполняют испытания в точках $x^{k+1}, x^{k+2}, \ldots, x^{k+p}$. Обозначим $I_k = \{x^{k+1}, x^{k+2}, \ldots, x^{k+p}\}$ — множество прообразов точек, в которых испытания начаты, но еще не завершены.

Алгоритм выполнения асинхронной схемы включает следующие шаги.

Шаг 1. Перенумеровать множество $X_k = \{x^1, x^2, \ldots, x^{k+p}\}$, содержащее все прообразы точек, в которых испытания либо завершены, либо выполняются, в порядке возрастания:

$$0 = x_1 < x_2 < \ldots < x_{k+p} = 1.$$

Шаг 2. Вычислить оценки:

$$M_1 = \max\left\{\frac{|z_i - z_{i-1}|}{(x_i - x_{i-1})^{1/N}} : x_{i-1} \notin I_k, x_i \notin I_k, 2 \leq i \leq k+p\right\},$$

$$M_2 = \max\left\{\frac{|z_{i+1} - z_{i-1}|}{(x_{i+1} - x_{i-1})^{1/N}} : x_i \in I_k, 2 \leq i < k+p\right\},$$

$$M = \max\{M_1, M_2\},$$

где $z_i = \varphi(y(x_i))$, если $x_i \notin I_k$. Если $M = 0$, положить $M = 1$.

Шаг 3. Для каждого интервала (x_{i-1}, x_i) с $x_{i-1} \notin I_k$, $x_i \notin I_k$ вычислить характеристику

$$R(i) = \Delta_i + \frac{(z_i - z_{i-1})^2}{(rM)^2 \Delta_i} - 2\frac{z_i + z_{i-1}}{rM},$$

где $\Delta_i = (x_i - x_{i-1})^{1/N}$.

Шаг 4. Выбрать интервал (x_{t-1}, x_t) с максимальной характеристикой:

$$R(t) = \max\{R(i) : x_{i-1} \notin I_k, x_i \notin I_k, 2 \leq i \leq k+p\}.$$

Шаг 5. Определить новую точку для проведения испытания:

$$x^{k+p+1} = \frac{x_t + x_{t-1}}{2} - \operatorname{sign}(z_t - z_{t-1})\frac{1}{2r}\left(\frac{|z_t - z_{t-1}|}{M}\right)^N.$$

Мастер добавляет точку x^{k+p+1} к множеству I_k и отправляет ее освободившемуся рабочему процессу для проведения нового испытания.

Алгоритм завершает работу при выполнении условия $\Delta_t < \varepsilon$ или $k + p > K_{\max}$.

Основное преимущество асинхронной схемы заключается в минимизации простоев вычислительных ресурсов: мастер обрабатывает результаты по мере их поступления, а рабочие процессы немедленно получают новые задания после завершения предыдущих.

4. Результаты численных экспериментов

Вычислительные эксперименты проводились на узле суперкомпьютера «Лобачевский» в следующей конфигурации: операционная система Linux CentOS 8.2, 2 процессора Intel Xeon Platinum 8558 2.1/4GHz (48 ядер каждый), оперативная память 2048 Gb, 8 графических ускорителей Nvidia RTX PRO 6000. Рассмотренные алгоритмы оптимизации реализованы на C++, использовался компилятор Intel C++ Compiler 2023 с поддержкой OpenMP. Обучение моделей искусственного интеллекта проводились с использованием фреймворка PyTorch, использовался Python 3.12.

Для проведения экспериментов использовались две нейросетевые модели, описанные в разделе 2: модель классификации на основе MobileNetV3 Small и модель сегментации на основе одномерной U-Net. Вычислительные затраты на одну итерацию оптимизации (одно испытание) составляли:

- для задачи классификации — обучение модели в течение 100 эпох с оценкой качества на валидационной выборке;
- для задачи сегментации — обучение модели в течение 40 эпох с вычислением метрики качества.

Параметры запуска алгоритма глобального поиска в системе Globalizer:

- параметр надежности $r = 10$;
- точность глобального поиска $\varepsilon = 10^{-2}$;
- максимальное число поисковых испытаний $K_{\max} = 320$;
- плотность построения развертки $m = 10$.

Для синхронной и асинхронной схем распараллеливания использовалась схема мастер-рабочий (один мастер и несколько рабочих), всего использовалось MPI процессов: 1, 3, 5, 9, 17, 33. В качестве целевой метрики использовалась макро-усреднённая F1.

На первом этапе экспериментов выполнялась оптимизация гиперпараметров модели классификации ЭКГ-сигналов. Варьируемыми параметрами являлись:

- число классифицирующих нейронов в выходном слое: $f \in [80, 200]$;
- Вероятность отключения нейронов: $p \in [0.0, 1.0]$.

Для визуализации характера целевой функции был построен график (рис. 4), на котором представлена интерполяция значений целевой метрики в зависимости от числа нейронов и вероятность отключения нейронов. Видно, что функция имеет выраженный многоэкстремальный характер, что подтверждает необходимость применения методов глобальной оптимизации.

В табл. 1 представлены результаты масштабирования параллельных алгоритмов при оптимизации гиперпараметров модели классификации. Для каждого количества процессов указано найденное значение целевой метрики и достигнутое ускорение относительно последовательного запуска, длительность которого составила 11.3 минуты.

Анализ полученных результатов показывает, что асинхронная схема распараллеливания демонстрирует более высокую эффективность по сравнению с синхронной схемой, особенно при увеличении числа процессов. Это объясняется тем, что время обучения нейросетевой модели может варьироваться в зависимости от конкретных значений гиперпараметров, и асинхронная схема позволяет минимизировать простои вычислительных ресурсов. При этом при разном количестве процессов мы получаем разные траектории, в которых суммарное время вычислений значительно отличается.

Для сравнительной оценки эффективности разработанного подхода была выполнена настройка гиперпараметров модели классификации с использованием широко распространенных фреймворков Optuna [5] и HyperOpt [6] (в последовательном режи-

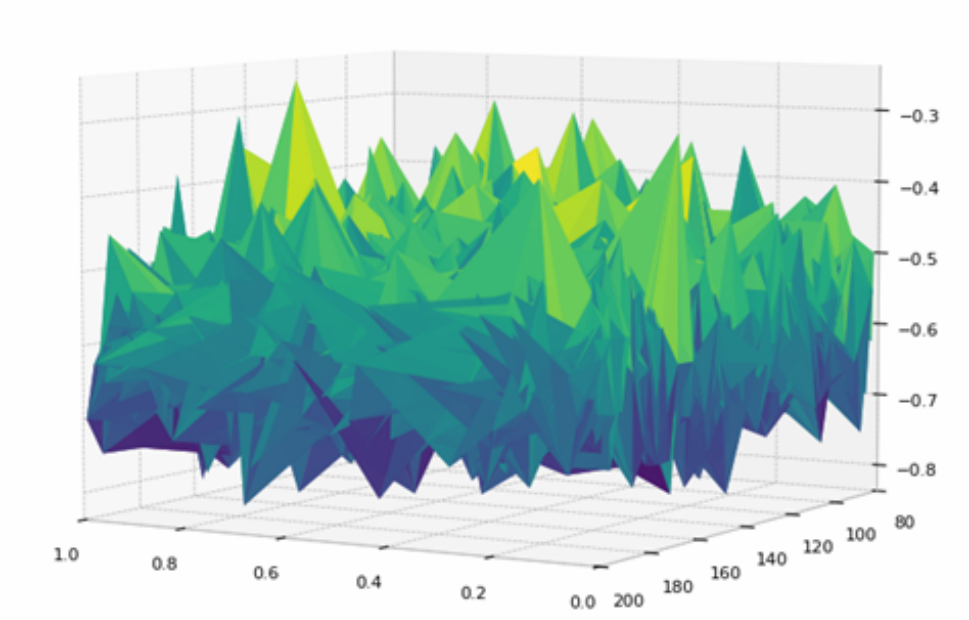


Рис. 4. Поверхность целевой функции для задачи классификации ЭКГ в зависимости от числа нейронов и вероятности Dropout

ме). В ходе оптимизации с помощью Optuna достигнуто значение целевой метрики (максимизируемой макро-усреднённой F1-меры), равное 0.75; время решения составило 13.4 минуты. Применение HyperOpt позволило получить значение метрики 0.775, время решения составило 12.3 минуты. Таким образом, все три фреймворка — Optuna, HyperOpt и Globalizer — обеспечивают сопоставимое качество оптимизации, однако использование Globilizer характеризуется меньшим временем работы, что подтверждает его практическую эффективность в задачах подбора гиперпараметров нейросетевых моделей.

Таблица 1. Результаты масштабирования для задачи классификации ЭКГ

Число процессов	Синхронная схема		Асинхронная схема	
	F_1^{macro}	Ускорение	F_1^{macro}	Ускорение
1	0.78	1.00	0.78	1.00
3	0.75	1.98	0.78	1.94
5	0.80	3.91	0.78	3.65
9	0.78	7.32	0.78	10.11
17	0.78	12.81	0.78	22.91
33	0.78	18.07	0.83	15.94

На втором этапе экспериментов выполнялась оптимизация гиперпараметров модели сегментации ЭКГ-сигналов на основе одномерной U-Net. Варьируемыми параметрами являлись:

- Вероятность отключения нейронов: $p \in [0.0, 1.0]$;
- количество уровней энкодера/декодера: $depth \in [3, 5]$.

Для каждого класса c вычислялись precision и recall:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}, \qquad \text{Recall}_c = \frac{TP_c}{TP_c + FN_c},$$

а затем F1-мера:

$$F_{1,c} = \frac{2 \cdot \text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}.$$

Здесь TP_c, FP_c, FN_c — число истинно-положительных, ложноположительных и ложноотрицательных предсказаний класса c, подсчитанных по всем отсчетам времени t на всей выборке.

В рамках эксперимента макро-усреднённая F1-мера рассчитывался по точному совпадению меток на каждом отсчете (без временного допуска). Итоговая метрика для сравнения моделей вычислялась как среднее по классам без фона:

$$F_1^{\text{macro}} = \frac{1}{C_{\text{cls}} - 1} \sum_{c=1}^{C_{\text{cls}}-1} F_{1,c}.$$

В табл. 2 представлены результаты масштабирования параллельных алгоритмов при оптимизации гиперпараметров модели сегментации. При установленном ограничении на число испытаний $K_{max} = 320$ длительность последовательного запуска составила 5.5 часов.

Таблица 2. Результаты масштабирования для задачи сегментации ЭКГ

Число процессов	Синхронная схема		Асинхронная схема	
	F_1^{macro}	Ускорение	F_1^{macro}	Ускорение
1	0.80	1.00	0.80	1.00
3	0.79	1.87	0.79	1.85
5	0.80	3.10	0.79	3.38
9	0.80	5.69	0.80	6.34
17	0.79	9.46	0.79	15.68
33	0.80	13.88	0.79	14.29

Сравнение результатов для задач классификации и сегментации показывает, что эффективность асинхронной схемы вычислений выше для задачи сегментации, что связано с большей вычислительной трудоемкостью одного испытания (обучение модели сегментации требует больше времени вследствие большей глубины сети и большего объема обрабатываемых данных).

В рамках рассматриваемой диагностической задачи методы поиска глобального экстремума служат инструментом обеспечения стабильной и воспроизводимой работы модели в условиях ограниченных вычислительных ресурсов. Для обеих задач (классификации и сегментации) грубая оценка точки оптимума, полученная в условиях

ограниченного вычислительного ресурса, не приводит к критической деградации диагностических свойств модели. В то же время для задачи сегментации основной целью является точная локализация морфологических элементов кардиоцикла для последующего расчёта интервалов. В данном контексте подбор гиперпараметров сегментационной сети обеспечивает устойчивое выделение границ комплексов, что напрямую влияет на надежность последующей интерпретации.

Заключение

В работе рассмотрена задача оптимизации гиперпараметров нейросетевых моделей классификации и сегментации сигналов ЭКГ с использованием параллельного алгоритма глобального поиска, реализованного в программной системе Globalizer. Для задачи классификации (модель MobileNetV3 Small) варьировались число нейронов (80–200) и вероятность отключения нейронов (0.0–1.0), достигнуто значение целевой метрики F1-macro, равное 0.83. Для задачи сегментации (одномерная U-Net) варьировались глубина сети (3–5) и Dropout (0.0–1.0), достигнуто значение целевой метрики F1-macro равное 0.80. Исследована эффективность синхронной и асинхронной схем распараллеливания на суперкомпьютере «Лобачевский». Асинхронная схема показала более высокую эффективность: при 17 процессах ускорение составило 22.91 для классификации и 15.68 для сегментации. Полученные результаты подтверждают, что асинхронный алгоритм глобальной оптимизации, реализованный в системе Globalizer, является эффективным инструментом настройки гиперпараметров нейронных сетей для задач медицинской диагностики.

Список литературы

[1] Ansari Y., Mourad O., Qaraqe K., Serpedin E. Deep learning for ECG arrhythmia detection and classification: an overview of progress for period 2017–2023 // Frontiers in Physiology. 2023. Vol. 14. 1246746.

[2] Seenivasan D., Sakthivel K. Patch-Based ECG Segmentation for Arrhythmia Classification Using Hybrid Deep Learning // Information Technology and Control. 2025. Vol. 54, No. 4. P. 1189–1205.

[3] Xiao Q., Lee K., Mokhtar S.A., Ismail I., Pauzi A.L.B.M., Zhang Q., Lim P.Y. Deep learning-based ECG arrhythmia classification: A systematic review // Applied Sciences. 2023. Vol. 13, No. 8. 4964.

[4] Huang Y., Li H., Yu X. A novel time representation input based on deep learning for ECG classification // Biomedical Signal Processing and Control. 2023. Vol. 83. 104628.

[5] Akiba T., Sano S., Yanase T., Ohta T., Koyama M. Optuna: A Next-Generation Hyperparameter Optimization Framework // Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019. P. 2623—2631. DOI: 10.1145/3292500.3330701

[6] Bergstra J., Yamins D., Cox D.D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures // Proceedings of the 30th International Conference on Machine Learning. 2013. P. 115–123.

[7] Yang X.-S. Engineering optimization: An introduction with metaheuristic applications. Hoboken: John Wiley & Sons, 2013. 384 p.

[8] Gendreau M., Potvin J.-Y. (Eds.) Handbook of metaheuristics. 2nd ed. New York: Springer, 2010. 640 p.

[9] Eiben A.E., Smith J.E. Introduction to evolutionary computing. 2nd ed. Berlin: Springer, 2015. 294 p.

[10] Квасов Д.Е., Сергеев Я.Д. Методы липшицевой глобальной оптимизации в задачах управления // АиТ. 2013. № 9. С. 3–19. DOI: 10.1134/S0005117913090014

[11] Сергеев Я.Д., Квасов Д.Е. Диагональные методы глобальной оптимизации. М.: ФИЗМАТЛИТ, 2008. 352 с.

[12] Žilinskas A. Branch and bound algorithm for multidimensional scaling with city-block metric // Journal of Global Optimization. 2008. Vol. 43, No. 2–3. P. 357–372. DOI: 10.1007/s10898-008-9306-x.

[13] Strongin R.G., Sergeyev Y.D. Global optimization with non-convex constraints. Sequential and parallel algorithms. Dordrecht: Kluwer Academic Publishers, 2000. 416 p. DOI: 10.1007/978-1-4615-4677-1.

[14] Strongin R., Gergel V., Barkalov K., Sysoyev A. Generalized parallel computational schemes for time-consuming global optimization // Lobachevskii Journal of Mathematics. 2018. Vol. 39, No. 4 P. 576–586. DOI: 10.1134/S1995080218040133.

[15] Kalyakulina A.I., Yusipov I.I., Moskalenko V.A., Nikolskiy A.V., Kozlov K.A., Zudov K.A., Ivanchenko M.V. LUDB: a new open-access validation tool for electrocardiogram delineation algorithms // IEEE Access. 2020. Vol. 8. P. 186181–186190. DOI: 10.1109/ACCESS.2020.3029376.

[16] Евтушенко Ю.Г., Малкова В.У., Станевичюс А.-И.А. Параллельный поиск глобального экстремума функций многих переменных // Ж. вычисл. матем. и матем. физ. 2009. Т. 49, № 2. С. 255–269. DOI: 10.1134/S0005117907050062.

[17] Žilinskas A., Žilinskas J. Global optimization based on a statistical model and simplicial partitioning // Computers & Operations Research. 2010. Vol. 37, No. 10. P. 1759–1767. DOI: 10.1016/S0898-1221(02)00206-7.

[18] Pintér J.D. Global optimization in action: Continuous and Lipschitz optimization: Algorithms, implementations and applications. Dordrecht: Kluwer Academic Publishers, 1996. 480 p.

[19] Jones D.R. The DIRECT global optimization algorithm // Encyclopedia of Optimization. Boston: Springer, 2009. P. 725–735. DOI: 10.1007/978-0-387-74759-0_128

[20] Стронгин Р.Г., Гергель В.П., Гришагин В.А., Баркалов К.А. Параллельные вычисления в задачах глобальной оптимизации. М.: Изд-во МГУ, 2013.

[21] Barkalov K., Lebedev I., Karchkov D. Analysis and elimination of bottlenecks in parallel algorithm for solving global optimization problems // Lecture Notes in Computer Science. 2022. Vol. 13708. P. 18–32. DOI: 10.1007/978-3-031-22941-1_2.

[22] Barkalov K., Lebedev I. Solving multidimensional global optimization problems using graphics accelerators // Communications in Computer and Information Science. 2016. Vol. 687. P. 224–235. DOI: 10.1007/978-3-319-55669-7_18.

[23] Gergel V., Barkalov K., Sysoyev A. A novel supercomputer software system for solving time-consuming global optimization problems // Numerical Algebra, Control & Optimization. 2018. Vol. 8, No. 1. P. 47–62. DOI: 10.3934/naco.2018003.

Параллельная эффективность модели атмосферы на сетке кубическая сфера с подключенным блоком параметризаций физических процессов подсеточного масштаба

В.В. Шашкин[1,2,3], Г.С. Гойман[1], Д.А. Марханов[1,2], И.Д. Третьяк[1,2,3]

[1]Институт вычислительной математики им. Г.И. Марчука РАН
[2]Гидрометцентр России
[3]Московский физико-технический институт

В ИВМ РАН и Гидрометцентре России разрабатывается модель атмосферы ParCS на сетке кубическая сфера с квазиравномерным разрешением на сфере. Целевая область применения модели – глобальный прогноз погоды и моделирование климата с шагом сетки по горизонтали в широком диапазоне 100-3 км. В предыдущих исследованиях было показано, что блок численного решения уравнений динамики атмосферы ParCS соответствует передовым зарубежным моделям по точности решения общепринятых идеализированных задач. Следующим шагом к использованию ParCS в практических задачах является подключение описания физических процессов подсеточного масштаба. В данной работе рассматривается подключение блока параметризаций физических процессов подсеточного масштаба от модели атмосферы ПЛАВ, которая используется для оперативного прогноза погоды в Гидрометцентре. На примере эксперимента аквапланета обосновывается выбор схемы расщепления для совместного интегрирования системы «подсеточная физика» — динамика, и использование вычислений в одинарной точности в блоке динамики. Анализируется вычислительная эффективность и масштабируемость модели ParCS с подсеточной физикой. Рассматриваемая конфигурация модели – горизонтальное разрешение около 100 км, 96 уровней по вертикали, что соответствует версии модели ПЛАВ для внутрисезонного прогноза погоды. Эксперименты в данной конфигурации являются необходимым этапом на пути к запускам в высоком разрешении, так как позволяют проверить корректность подключения подсеточной физики, выполнить базовую настройку новой модели и исследовать ее вычислительные характеристики при умеренных вычислительных затратах.

Ключевые слова: Моделирование атмосферы, прогноз погоды, климат

1. Введение

Гидродинамические математические модели атмосферы являются основным инструментом прогноза погоды, исследований климата и физики атмосферы. Типичное разрешение современных глобальных моделей атмосферы для задачи прогноза погоды на 3-14 дней — 10 км [2], в области долгосрочного прогноза погоды и исследований климата типичный шаг сетки – около 100 км. Перспективным считается переход к вычислениям на сетках с шагом по горизонтали в единицы километров [35], [19], [25]. В задаче прогноза погоды это позволит воспроизводить на сетке мезомасштабные метеорологические процессы, отвечающие за значительную часть опасных явлений погоды. В области моделирования климата сетки с подобным разрешением нужны, чтобы снизить неопределенность проекций изменений климата, создаваемую

параметризациями облачных процессов и глубокой конвекции [34], а также получить возможность явно моделировать изменения частоты опасных явлений по мере потепления климата.

Оценки показывают, что размерность задачи моделирования глобальной атмосферы с шагом сетки 3-5 км составляет величину порядка 10^{10} (при 100 уровнях по вертикали и 10 переменных). Практические расчеты с моделью такого класса потребуют эффективных вычислений на $O(10^5)$ ядер [2] или массиве из примерно 100 графических ускорителей. Требования параллельной эффективности приводят к необходимости отказаться от традиционной широтно-долготной сетки, так как уменьшение шага по долготе вблизи полюсов приводит к жёстким ограничениям на шаг по времени для явных методов и низкой параллельной эффективности в случае неявных методов из-за интенсивных коммуникаций в полярных областях [24]. При высоком разрешении эти проблемы становятся практически непреодолимыми. Для моделей атмосферы нового поколения, ориентированных на моделирование с разрешением в единицы километров, стандартом является применение сферических сеток с квазиравномерным разрешением, таких как кубическая сфера [21], [20], икосаэдральная сетка [31] и сетка Инь-Янь [18]. В зарубежных метеорологических центрах активная разработка моделей на квазиравномерных сферических сетках велась в прошлом десятилетии. На данный момент модели такого класса уже используются оперативно (пока с шагом сетки по горизонтали около 10 км) [33], [16] или близки к оперативному внедрению [13]. При помощи разработанных моделей (и их прототипов) ведутся активные исследования о необходимости применения негидростатических уравнений динамики атмосферы в системах прогноза погоды высокого разрешения [35], в рамках таких исследований проводятся расчеты прогнозов погоды на несколько суток с разрешением вплоть до 1 км. В области моделирования климата проводятся первые эксперименты с моделями атмосферы на сетках с шагом 1-5 км [25], [8]. В России (в ИВМ РАН и Гидрометцентре) активная разработка модели атмосферы на сетке кубическая сфера [20, 21] с квазиравномерным разрешением началась в 2021 году [22, 23]. Особенностью разрабатываемой модели (далее ParCS — от parallel cubed-sphere) является применение устойчивой пространственной аппроксимации высокого порядка точности на основе метода конечных разностей со свойством суммирования по частям (SBP finite-differences [4]). Данный подход является разумной альтернативой спектрально-элементным и конечно-объёмным методам, часто оказывается более эффективным за счет меньшего спектрального радиуса дискретных операторов (что позволяет использовать больший шаг по времени при эквивалентном разрешении [3]). Программный комплекс ParCS включает алгоритмы решения системы уравнений динамики атмосферы (динамические ядра) в гидростатическом приближении по вертикали [9] и без него [14]. Допускается использование схем интегрирования по времени с различной степенью неявности, а также полулагранжевой аппроксимации адвективных слагаемых. В работах [9, 14, 22] приведены результаты испытания алгоритмов решения уравнений динамики ParCS на идеализированных тестовых задачах. В последней работе [9] подробно рассматривается гидростатическая версия динамического ядра ParCS, сохраняющая энергию. Общий вывод работ – точность ParCS на идеализированных задачах динамики [12, 30] соответствует передовым зарубежным моделям. Важной частью моделей атмосферы, применяемых для решения практических задач, таких как прогноз погоды и моделирование климата, является блок параметризованного описания процессов подсеточного масштаба (блок параметризаций) [6]. Блок параметризаций представляет собой набор локально-одномерных (по вертикали) полуэмпирических моделей, которые воспроизводят эффект мелкомасштабных процессов

на динамику атмосферных течений, разрешаемых на сетке. Основные рассматриваемые процессы – турбулентное перемешивание в пограничном слое атмосферы, влажная конвекция, перенос излучения, облачность, распространение и обрушение гравитационных волн. В качестве основного варианта блока подсеточных параметризаций для модели ParCS в задачах прогноза погоды рассматривается использование параметризаций глобальной модели атмосферы ПЛАВ (ИВМ РАН, Гидрометцентр России) [27, 28], которая применяется оперативно в Гидрометцентре России для среднесрочного и внутрисезонного прогноза погоды. Данный блок параметризаций, основанный на работах [6, 7] с усовершенствованиями, предназначен для моделей атмосферы с горизонтальным разрешением вплоть до единиц километров.

В данной работе мы рассматриваем вычислительные проблемы, возникающие при подключении блока параметризаций физических процессов подсеточного масштаба к базовой версии динамического блока ParCS – эйлеровой гидростатической. На примере эксперимента аквапланета (воспроизведение общей циркуляции атмосферы на планете, целиком покрытой океаном, при заданной температуре поверхности) в низком разрешении (1°) исследуются вопросы чувствительности воспроизводимого климата к методу расщепления, применяемому для совместного интегрирования по времени системы уравнений динамики атмосферы и моделей подсеточных процессов. Также мы исследуем масштабируемость программного комплекса ParCS с подключенными параметризациями подсеточных физических процессов и анализируем относительную вычислительную стоимость его компонентов с целью выявления точек для дальнейшей оптимизации. Изучаемая конфигурация модели ParCS соответствует самому низкому разрешению, в котором планируется ее применять. Подобные эксперименты – необходимый этап развития модели, так как позволяют выполнить базовую настройку параметризаций и изучить вычислительные характеристики модели при умеренных вычислительных затратах.

В разделе 2 изучается зависимость модельного климата от схемы интегрирования по времени для системы динамика — подсеточная физика. В разделе 3 исследуется масштабируемость программного комплекса на параллельной ЭВМ, анализируется относительная вычислительная стоимость компонентов. В разделе 4 формулируются основные результаты работы.

2. Совместное интегрирование уравнений динамики и параметризаций физических процессов

2.1. Уравнения гидротермодинамики атмосферы

Гидростатическая версия модели ParCS [9] решает примитивные уравнения динамики атмосферы [11] на сетке кубическая сфера в гибридной σ-p системе координат по вертикали [15]. В развитие работы [9] система уравнений была обобщена на случай динамики влажного воздуха и приобрела вид:

$$\frac{\partial \vec{v}}{\partial t} + (f+\zeta)\vec{k}\times\vec{v} + \dot{\eta}\frac{\partial \vec{v}}{\partial \eta} + \nabla_\eta(K+\Phi) + \frac{RT_v}{p}\nabla_\eta p = F_{\vec{v}}, \tag{1}$$

$$\frac{\partial T}{\partial t} + \vec{v}\cdot\nabla_\eta T + \dot{\eta}\frac{\partial T}{\partial \eta} - \frac{RT_v}{c_{pv}p}\omega = F_T, \tag{2}$$

$$\frac{\partial}{\partial t}\left(\frac{\partial p}{\partial \eta}\right) = \nabla\cdot\left(\frac{\partial p}{\partial \eta}\vec{v}\right) + \frac{\partial}{\partial \eta}\left(\frac{\partial p}{\partial \eta}\dot{\eta}\right), \tag{3}$$

$$\frac{\partial q_i}{\partial t} + \vec{v} \cdot \nabla_\eta q_i + \dot{\eta} \frac{\partial q_i}{\partial \eta} = F_{q_i}, \tag{4}$$

где $\vec{v}$ – вектор горизонтальной скорости ветра, f – параметр Кориолиса, $\zeta = \vec{k} \cdot (\nabla \times \vec{v})$ – вертикальная компонента относительной завихренности, $\vec{k}$ – внешняя нормаль к сфере, η – вертикальная координата, $\dot{\eta}$ – вертикальная скорость в η-координате, $K = \vec{v} \cdot \vec{v}/2$ – удельная кинетическая энергия горизонтального движения, Φ – геопотенциал, R – газовая постоянная сухого воздуха, R_v – газовая постоянная влажного воздуха, $T_v = \frac{R_v}{R} T$ – виртуальная температура, p – давление, ∇_η – градиент при постоянной η, c_{pv} – удельная теплоёмкость влажного воздуха при постоянном давлении, $\omega = Dp/Dt$ – вертикальная скорость в изобарической системе координат, q_i – удельная концентрация водяного пара ($i = 0$) или гидрометеоров ($i \in [1, 4]$). Величины F в правых частях уравнений обозначают тенденции, создаваемые физическими процессами подсеточного масштаба.

Газовая постоянная и теплоемкость влажного воздуха определяются как

$$R_v = R \left(1 - \sum_{i=0}^{4} q_i \right) + q_0 R_q, \tag{5}$$

$$c_{pv} = c_p \left(1 - \sum_{i=0}^{4} q_i \right) + q_0 c_{pq} + \sum_{i=1}^{4} c_i q_i, \tag{6}$$

где R_q и c_{pq} — газовая постоянная и удельная теплоёмкость при постоянном давлении водяного пара, c_p — удельная теплоёмкость сухого воздуха при постоянном давлении, c_i — удельная теплоемкость i-го гидрометеора. Формула для вычисления геопотенциала:

$$\Phi = \Phi_s - \int_0^\eta \frac{R T_v}{p} \frac{\partial p}{\partial \eta'} d\eta', \tag{7}$$

где Φ_s — геопотенциал поверхности Земли.

Дискретизация системы уравнений (1)–(4) на сетке кубическая сфера и алгоритм численного решения совпадают с работой [9]. Дискретная форма уравнения переноса водяного пара и гидрометеоров (4) совпадает с дискретной формой адвективной части уравнения термодинамики (2). Интегрирование уравнений динамики по времени производится классической явной схемой Рунге–Кутты 4-го порядка точности.

2.2. Совместное интегрирование уравнений динамики и «подсеточной физики»

Блок параметризаций процессов подсеточного масштаба представляет собой набор одномерных по вертикали статистико-эмпирических моделей отдельных процессов – распространения излучения, турбулентной диффузии в пограничном слое атмосферы, влажной конвекции, микрофизики облаков, обрушения внутренних гравитационных волн [1] и других. Большинство из этих моделей предполагают неявную схему интегрирования по времени, ввиду быстроты протекающих процессов. Совместное интегрирование по времени всех моделей подсеточных процессов, а также разрешаемой динамики не представляется разумным с точки зрения вычислительных затрат, из-за сложности и нелинейности получающейся системы уравнений [10]. Типичным является применение метода расщепления по физическим процессам [36], [29]. В рамках метода расщепления при данном начальном состоянии атмосферы ψ находится

состояние ψ^*, в которое пришла бы атмосфера за время τ_{phys} под действием исключительно процессов подсеточного масштаба. Тенденции, создаваемые подсеточными процессами, задаются как

$$F = (\psi^* - \psi)/\tau_{phys}. \tag{8}$$

В модели ПЛАВ состояние ψ^* используется как стартовое для интегрирования системы уравнений динамики (1)–(4) на один шаг по времени. При этом тенденции F в правых частях уже не должны учитываться, а шаг по времени динамики τ_{dyn} совпадает с шагом параметризаций τ_{phys}. Вычислительно дорогие параметризация радиации вызываются один раз в 4 шага, вычисленные тенденции «замораживаются» (в случае солнечной радиации обновляются с учетом зенитного угла солнца каждый шаг). Такая схема интегрирования по времени логична для полулагранжевых моделей атмосферы, к классу которых относится ПЛАВ, так как блок динамики устойчив при больших шагах по времени, $\tau_{dyn} = 1440$ с при разрешении $0.9^\circ \times 0.72^\circ$.

В эйлеровых моделях атмосферы шаг интегрирования уравнений динамики по времени меньше, чем в полулагранжевых. Для модели ParCS в эксперименте аквапланета характерный $\tau_{dyn} = 200$ с. Данный факт, а также возможность точного и устойчивого интегрирования физических параметризаций в модели ПЛАВ с шагом большим, чем шаг динамики ParCS, вызывает интерес к возможности производить расчеты физических параметризаций при $\tau_{phys} = n \cdot \tau_{dyn}$.

Подобные исследования проводились в работе [26] в рамках идеализированных экспериментов для модели CAM [5]. Рассматривалось два подхода к подключению физики. В рамках одного, подобно схеме модели ПЛАВ, вычислялось состояние ψ^*, в которое приходит атмосфера за τ_{phys} под действием только подсеточных процессов, затем с состояния ψ^* производится n шагов интегрирования уравнений динамики $n\tau_{dyn} = \tau_{phys}$ при $F = 0$. Второй подход заключался в вычислении тенденций по формуле (8) и интегрировании блока динамики на n шагов размера τ_{dyn} с использованием «замороженных» тенденций F в правой части. Первый подход приводил к возникновению артефактов решения, связанных с распространением циркулярных инерционно-гравитационных волн большой амплитуды от зон активной конденсации водяного пара, поэтому в данной работе не рассматривается. Второй подход был признан удовлетворительным [26], далее мы используем его для ускорения вычислений модели.

2.3. Численные эксперименты

Влияние схемы совместного интегрирования по времени уравнений динамики и моделей процессов подсеточного масштаба на особенности воспроизводимого климата производилось в рамках эксперимента аквапланета. Производился расчет двух конфигураций — «эталонной» и «базовой» — на 1865 дней (1500 дней, исключая период раскрутки). В обеих конфигурациях разрешение сетки кубическая сфера – около 1° (90 точек вдоль ребра куба), 96 уровней по вертикали (совпадает с вертикальной сеткой модели ПЛАВ в версии внутрисезонного прогноза погоды), шаг интегрирования по времени уравнений динамики – 200 с. В «эталонной» конфигурации расчет тенденции от блока параметризаций процессов подсеточного масштаба осуществлялся на каждом шаге блока динамики. В «базовой» конфигурации тенденции параметризаций вычислялись 1 раз в 1800 с модельного времени и использовались в течение 9 шагов блока динамики. Расчет параметризаций излучения в обеих конфигурациях производился 1 раз в 3600 с модельного времени. Отметим, что в «эталонной» конфигурации расчеты

блока динамики производились в двойной точности, в «базовой» использовалась одинарная точность. Чувствительность модели к точности машинной арифметики в блоке динамики оценивалась в отдельном эксперименте и находится ниже уровня статистической значимости. Мы рассматриваем «базовую» конфигурацию как основу для дальнейшей работы с моделью, так как она более чем в 2,5 раза быстрее «эталонной».

На Рис. 1 сравниваются средние за 1500 дней эксперимента поля интенсивности осадков по результатам расчета в двух конфигурациях. Поля осадков значимо отличаются в тропической зоне — в «базовой» конфигурации интенсивность осадков на

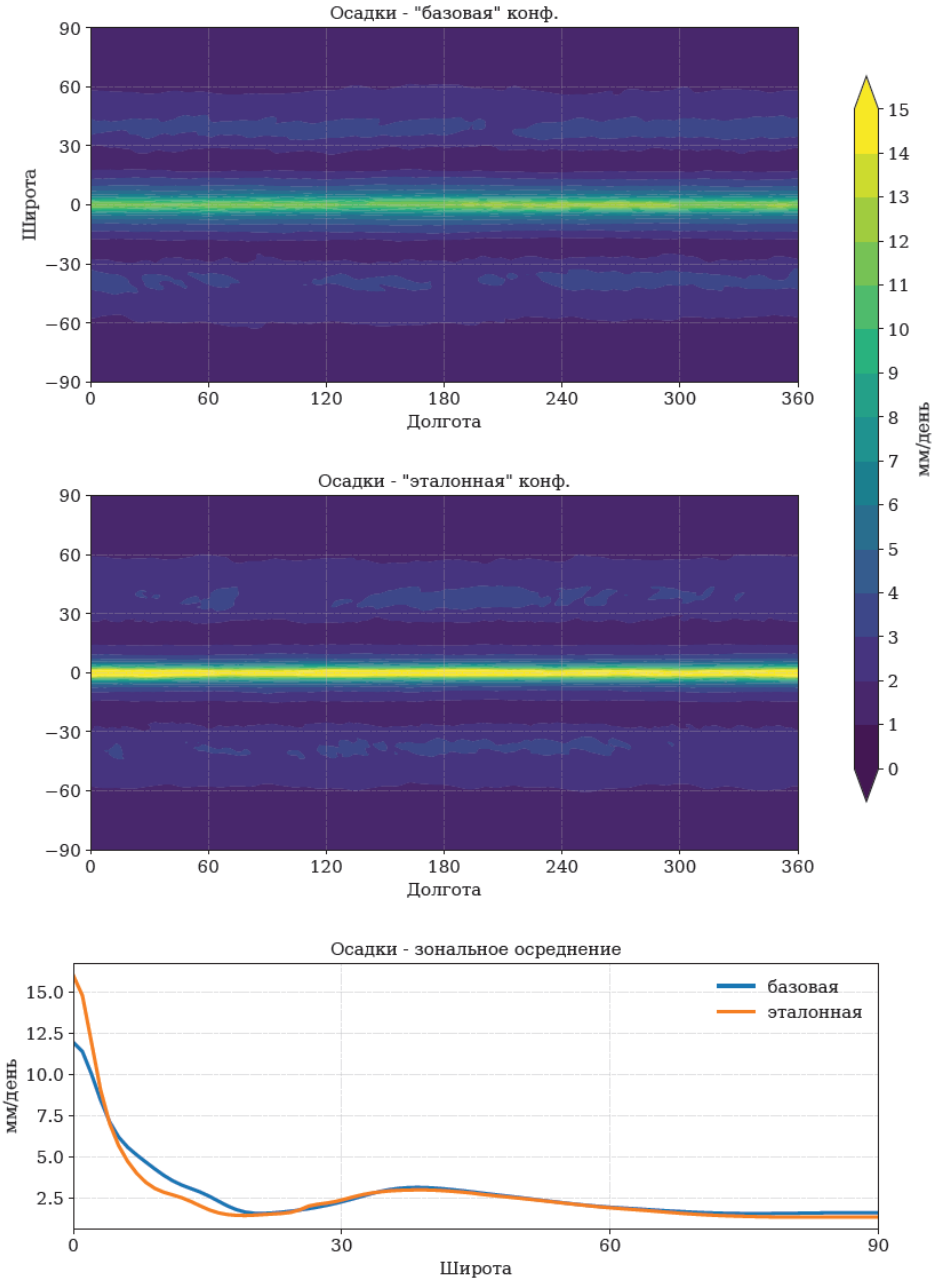


Рис. 1. Распределение осадков в расчетах с базовой и эталонной конфигурациями. Нижний график — осреднение вдоль круга широты и между полушариями.

экваторе ниже, чем в «эталонной». При этом результаты «базовой» конфигурации ближе к среднему мультимодельному ансамблю [32]. Последнее объясняется, по всей

видимости, тем, что настройки блока параметризаций модели ПЛАВ, который используется в данной работе, соответствуют версии модели ПЛАВ для долгосрочного прогноза погоды, производящей расчеты с шагом 1440 с (и параметризации, и динамика). Таким образом, режим работы параметризаций в «базовой» конфигурации ближе к режиму их работы в модели ПЛАВ. Анализ других полей (распределения температуры, скорости ветра) подтверждает сделанные выводы.

3. Эффективность расчетов на параллельных вычислительных системах

Параллельная реализация модели ParCS основана на технологии MPI с двумерной декомпозицией расчетной области по горизонтали. Применять декомпозицию по вертикали нецелесообразно, так как и в блоке численного решения уравнений динамики, и в блоке параметризаций физических процессов подсеточного масштаба используются алгоритмы с областью зависимости по данным, покрывающей весь вертикальный столбец сетки.

Оценка эффективности расчетов модели ParCS с подключенными параметризациями физических процессов подсеточного масштаба производилась в конфигурации эксперимента аквапланета [17], аналогичной рассматриваемой в разделе 2.3 (разрешение по горизонтали — около 1°, 96 уровней по вертикали). Шаг по времени алгоритма решения уравнений динамики $\Delta t = 200$ с, шаг по времени для подсеточных физических процессов – 1800 с, вызов расчета параметризаций радиации — 1 раз в 3600 с. Вычисления блока динамики производились в одинарной точности.

На Рис. 2 представлена скорость расчетов модели ParCS на ЭВМ Cray XC-40 Главного вычислительного центра Росгидромета при разном числе используемых процессоров. Для схемы 2-го порядка аппроксимации по горизонтали скорость расчетов при использовании 540 и 1620 ядер составляет 20,6 и 36,8 модельных лет за сутки соответственно. Ускорение относительно времени расчета на 36 ядрах – 12,1 (81% линейного) при 540 ядрах и 21,6 (48% линейного) при 1620 ядрах. Схема 4-го порядка аппроксимации по горизонтали несколько медленнее – 17,3 и 29,6 модельных года за сутки при использовании 540 и 1620 ядер соответственно. Ускорение относительно расчета на 36 ядрах было 13,1 (87% линейного) при 540 ядрах и 22,3 (49% линейного) при 1620 ядрах.

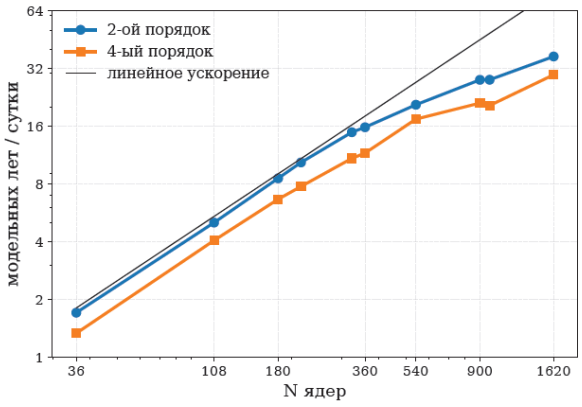


Рис. 2. Сильная масштабируемость при использовании схем второго и четвертого порядков аппроксимации.

Можно сделать заключение, что наблюдается вполне удовлетворительная эффективность параллельных расчетов для схем 2-го и 4-го порядка на 540 ядрах, что соответствует областям размером 9×10 точек по горизонтали, приходящимся на каждый MPI-процесс. При дальнейшем повышении числа ядер до 1620, что соответствует MPI-областям размером 5×6 точек по горизонтали, модель продолжает ускоряться, однако эффективность падает. Несмотря на большую ширину шаблона, метод 4-го порядка масштабируется несколько эффективней, чем метод 2-го порядка (эффективность 87% против 81% при расчете на 540 ядрах).

На Рис. 3 представлена доля вычислительного времени, приходящегося на каждый из крупных блоков модели при разном числе ядер. Можно отметить, что в целом все блоки масштабируются с примерно одинаковой эффективностью, не возникает ситуации, когда при большом числе ядер один из блоков начинает доминировать. Для схем 2-го и 4-го порядков доля вычислений блока численного решения уравнений динамики немного уменьшается при росте числа процессоров, доля же физики растет. Данный результат может показаться несколько парадоксальным, так как блок подсеточной физики, в отличие от блока динамики, не требует выполнения обменов данными. Однако блок динамики масштабируется сверхлинейно при 180-540 ядрах. В блоке физики вычисления производятся над последовательностью векторов из 16 вертикальных колонок сетки, что позволяет добиться хорошей эффективности использования кэш-памяти процессора при низком числе ядер, однако число 16 может быть не оптимальным при некоторых параллельных конфигурациях. Кроме того, в блоке физики существует неизбежный дисбаланс вычислительной нагрузки между MPI-подобластями, приходящимися на дневную и ночную стороны.

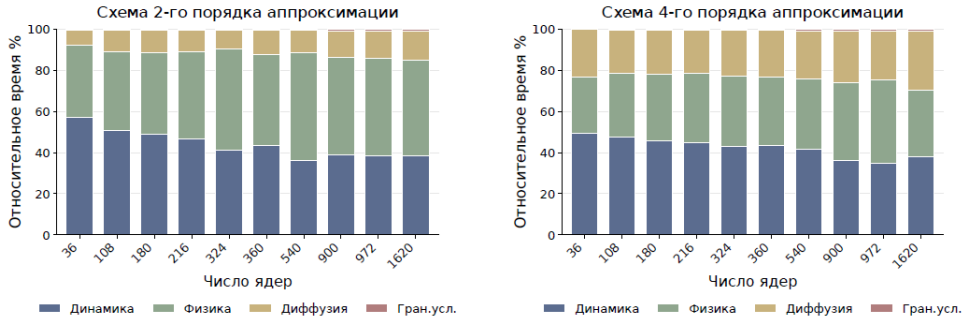


Рис. 3. Относительное время исполнения основных блоков при разном числе используемых CPU-ядер для схемы 2-го и 4-го порядков аппроксимации.

Заключение

Работа посвящена развитию глобальной модели атмосферы ParCS на сетке кубическая сфера. Метод численного решения уравнений динамики в гидростатическом приближении по вертикали [9] дополнен блоком параметризаций физических процессов подсеточного масштаба от модели ПЛАВ [28], которая применяется в Гидрометцентре для оперативного среднесрочного и внутрисезонного прогноза погоды.

Рассматриваемая версия блока динамики ParCS [9] использует эйлерову формулировку уравнений и, следовательно, требует значительно меньших шагов по времени, чем модель ПЛАВ. Таким образом, важным аспектом для вычислительной эффективности модели становится возможность производить вычисления блока подсеточной физики реже, чем блока динамики. В эксперименте аквапланета показано, что схема

расщепления, в которой тенденции подсеточной физики вычисляются один раз за 1800 с модельного времени и далее «замораживаются» в течение 9 шагов динамики, является приемлемой с точки зрения точности. При использовании такой схемы не наблюдается значительных артефактов численного решения, а распределение осадков оказывается ближе к результатам мультимодельного ансамбля [32], чем при вычислении тенденций подсеточных процессов на каждом шаге. Последнее, скорее всего, объясняется тем, что настройки параметризации взяты без изменений из версии модели ПЛАВ для внутрисезонного прогноза погоды, которая производит вычисления с шагом 1440 с.

Проведённый анализ параллельной эффективности показал, что модель ParCS с подключённой подсеточной физикой хорошо масштабируется на многопроцессорных вычислительных системах. При использовании 540 ядер эффективность остаётся высокой как для схемы второго, так и для схемы четвёртого порядка аппроксимации. Дальнейшее увеличение числа ядер до 1620 также приводит к ускорению расчётов, но с ожидаемым снижением эффективности. Анализ распределения вычислительного времени между основными блоками модели показал отсутствие выраженного «узкого места»: блоки динамики, подсеточной физики и диффузии масштабируются с сопоставимой эффективностью.

При разрешении около 1° (90 точек вдоль ребра куба) и 96-ти уровнях по вертикали скорость расчетов для схемы 4-го порядка на 540 вычислительных ядрах составляет 17,3 года/сутки, что близко к скорости расчета модели ПЛАВ в конфигурации внутрисезонного прогноза погоды — 16 лет/сутки (разрешение $0.9° \times 0.72°$ по долготе и широте соответственно, 96 вертикальных уровней). Следует однако отметить, что при таких параметрах число точек сетки ParCS примерно в два раза меньше, чем число точек сетки ПЛАВ. По формальному числу степеней свободы внутрисезонная версия модели ПЛАВ соответствует сетке кубическая сфера с 128-ю точками вдоль ребра. В терминах же эффективного разрешения (на основе не приведенного здесь анализа спектров кинетической энергии) аналог ПЛАВ $0.9° \times 0.72°$ – именно сетка кубическая сфера с 90 точками вдоль ребра.

Дальнейшее ускорение расчетов модели ParCS может быть достигнуто с помощью частичного перевода параметризаций на расчеты в одинарной точности (что уже сделано в модели ПЛАВ), подбора оптимальной длины вектора для вычислений в блоке подсеточной физики. Также все еще возможна оптимизация кода в блоке динамики и диффузии, которые в сумме занимают почти 2/3 времени вычислений.

Список литературы

[1] Doppler-spread parameterization of gravity-wave momentum deposition in the middle atmosphere. Part 1: Basic formulation. Journal of Atmospheric and Solar-Terrestrial Physics, 59(4):371–386, 1997. doi:10.1016/S1364-6826(96)00079-X.

[2] P. Bauer, A. Thorpe, and G. Brunet. The quiet revolution of numerical weather prediction. Nature, 535:47–55, 2015. doi:10.1038/nature14956.

[3] Pieter D Boom, David C Del Rey Fernández, and David W Zingg. Relative efficiency of finite-difference and discontinuous spectral-element summation-by-parts methods on distorted meshes. Journal of Scientific Computing, 102(1):28, 2025.

[4] David C. Del Rey Fernández, Jason E. Hicken, and David W. Zingg. Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations. Computers & Fluids, 95:171–196, 2014. doi:10.1016/j.compfluid.2014.02.016.

[5] A. Fournier, M. Taylor, and J. Tribbia. The spectral element atmospheric model: High-resolution parallel computation and response to regional forcing. Mon. Wea. Rev., 132:726–748, 2004. doi:10.1175/1520-0493(2004)132<0726:TSEAMS>2.0.CO;2.

[6] J.-F. Geleyn, E. Bazile, and P. Bougeault. Atmospheric parameterization schemes in Meteo-France's ARPEGE N.W.P. model. In Parameterization of subgrid-scale physical processes, ECMWF Seminar proceedings, pages 385–402, Reading, UK, 1994.

[7] L. Gerard, J.-M. Piriou, R. Brožková, J.-F. Geleyn, and D. Banciu. Cloud and precipitation parameterization in a meso-gamma-scale operational weather prediction model. Mon. Wea. Rev., 137:3960–3977, 2009. doi:10.1175/2009MWR2750.1.

[8] M. A. Giorgetta, W. Sawyer, and X. Lapillonne. The ICON-A model for direct QBO simulations on GPUs (version icon-cscs:baF28a514). Geoscientific Model Development, 15(18):6985–7016, 2022. doi:10.5194/gmd-15-6985-2022.

[9] G.S. Goyman, V.V. Shashkin, D.A. Markhanov, I.D. Tretyak, and Z.B. Khaidapov. Summation-by-Parts finite-difference hydrostatic dynamical core for atmospheric model on the cubed-sphere grid. Lobachevskii Journal of Mathematics, 47:1156–1175, 2026. doi:10.1134/S1995080226616255.

[10] Markus Gross, Hui Wan, and Philip J. Rasch. Physics–dynamics coupling in weather, climate, and earth system models: Challenges and recent progress. Monthly Weather Review, 146(11):3505–3544, 2018. doi:10.1175/MWR-D-17-0345.1.

[11] J. R. Holton. An introduction to dynamic meteorology, volume 88 of Int. Geophys. Ser. Elsevier Academic Press, fourth edition, 2004.

[12] C. Jablonowski, P. H. Lauritzen, M. Taylor, and R. D. Nair. Idealized test cases for the dynamical cores of atmospheric general circulation models. A proposal for the NCAR ASP 2008 summer colloquium, 2008. URL: http://www.cgd.ucar.edu/cms/pel/asp2008/idealized_testcases.pdf.

[13] J. Kent, T. Melvin, and G. A. Wimmer. A mixed finite element discretisation of the shallow water equations. Geoscientific Model Development Discussions, 2022:1–17, 2022. doi:10.5194/gmd-2022-225.

[14] D. Markhanov, V. Shashkin, and G. Goyman. An energy consistent summation-by-parts finite difference discretization for the non-hydrostatic atmospheric dynamics equations on the cubed-sphere grid. Russian Journal of Numerical Analysis and Mathematical Modelling, 40(2):121–140, 2025.

[15] A. McDonald and J.E. Haugen. A two time-level, three-dimensional, semi-Lagrangian, semi-implicit, limited-area gridpoint model of the primitive equations. Part II: extension to hybrid vertical coordinates. Mon. Wea. Rev., 121(7):2077–2087, 1993. doi:10.1175/1520-0493(1993)121<2077:ATTLTD>2.0.CO;2.

[16] J. Mouallem, L. Harris, and R. Benson. Multiple same-level and telescoping nesting in GFDL's dynamical core. Geoscientific Model Development, 15(11):4355–4371, 2022. doi:10.5194/gmd-15-4355-2022.

[17] R. B. Neale and B. J. Hoskins. A standard test for AGCMs including their physical parametrizations: I: the proposal. Atmospheric Science Letters, 1(2):101–107, 2000. doi:10.1006/asle.2000.0022.

[18] A. Quaddouri and V. Lee. The Canadian Global Environmental Multiscale model on the Yin-Yang grid system. Quart. J. Roy. Met. Soc., 137:1913–1926, 2011. doi:10.1002/qj.873.

[19] T. Rackow, X. Pedruzo-Bagazgoitia, and T. Becker. Multi-year simulations at kilometre scale with the Integrated Forecasting System coupled to FESOM2.5 and NEMOv3.4. Geoscientific Model Development, 18(1):33–69, 2025. doi:10.5194/gmd-18-33-2025.

[20] M. Rančić, R. J. Purser, and F. Mesinger. A global shallow-water model using an expanded spherical cube: Gnomonic versus conformal coordinates. Quarterly Journal of

the Royal Meteorological Society, 122(532):959–982, 1996. doi:10.1002/qj.49712253209.

[21] R. Sadourny. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. Monthly Weather Review, 100(2):136–144, 1972. doi:10.1175/1520-0493(1972)100<0136:CFAOTP>2.3.CO;2.

[22] V. Shashkin, Gordey Goyman, and Ilya Tretyak. Development of the next-generation atmosphere dynamics model in Russia: Current state and prospects. Lobachevskii Journal of Mathematics, 45:3159–3172, 10 2024. doi:10.1134/S1995080224603746.

[23] Vladimir V. Shashkin, Gordey S. Goyman, and Mikhail A. Tolstykh. Summation-by-parts finite-difference shallow water model on the cubed-sphere grid. Part I: Non-staggered grid. Journal of Computational Physics, 474:111797, 2023. doi:10.1016/j.jcp.2022.111797.

[24] A. Staniforth and J. Thuburn. Horizontal grids for global weather and climate prediction models: a review. Quart. J. Roy. Met. Soc., 138:1–26, 2012. doi:10.1002/qj.958.

[25] Christopher Terai, Noel Keen, and Peter Caldwell. Climate response to warming in Cess-Potter simulations using the global 3-km SCREAM. preprint, 01 2025. doi:10.22541/essoar.173655643.33295443/v1.

[26] D. R. Thatcher and C. Jablonowski. A moist aquaplanet variant of the Held-Suarez test for atmospheric model dynamical cores. Geosci. Model Dev., 9(4):1263–1292, 2016. doi:10.5194/gmd-9-1263-2016.

[27] M. Tolstykh, R. Fadeev, V. Shashkin, G. Goyman, R. Zaripov, V. Mizyak, V. Rogutov, K. Alipova, and E. Biryuchena. Global slav10 model for medium-range weather prediction. Russian Meteorology and Hydrology, 50(6):473–481, 2025.

[28] M. Tolstykh, V. Shashkin, R. Fadeev, and G. Goyman. Vorticity-divergence semi-Lagrangian global atmospheric model SL-AV20: dynamical core. Geoscientific Model Development, 10(5):1961–1983, 2017. doi:10.5194/gmd-10-1961-2017.

[29] Stefano Ubbiali, Christoph Schär, Linda Schlemmer, and Thomas C. Schulthess. A numerical analysis of six physics-dynamics coupling schemes for atmospheric models. Journal of Advances in Modeling Earth Systems, 13(11):e2020MS002377, 2021. doi:10.1029/2020MS002377.

[30] P. A. Ullrich, C. Jablonowski, J. Kent, P. H. Lauritzen, R. D. Nair, and M. A. Taylor. Dynamical core model intercomparison project (DCMIP) test case document. 2012. URL: http://earthsystemcog.org/site_media/docs/DCMIP-TestCaseDocument_v1.7.pdf.

[31] H. Wan, M. A. Giorgetta, G. Zangl, M. Restelli, D. Majewski, L. Bonaventura, K. Fruhlich, D. Reinert, P. Ripodas, L. Kornblueh, and J. Forstner. The ICON-1.2 hydrostatic atmospheric dynamical core on triangular grids - part 1: Formulation and performance of the baseline version. Geosci. Model Dev., 6:735–763, 2013.

[32] D. L. Williamson, M. Blackburn, and B. J. Hoskins. The APE atlas, NCAR technical note NCAR/TN-484+STR. 2012. doi:10.5065/D6FF3QBR.

[33] G. Zangl, D. Reinert, P. Ripodas, and M. Baldauf. The ICON (ICOshedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core. Quart. J. Roy. Met. Soc., 141(687):563–579, 2015. doi:10.1002/qj.2378.

[34] Mark Zelinka, Timothy Myers, and Daniel Mccoy. Causes of higher climate sensitivity in CMIP6 models. Geophysical Research Letters, 47, 01 2020. doi:10.1029/2019GL085782.

[35] C. Zeman, N. P. Wedi, P. D. Dueben, N. Ban, and C. Schär. Model intercomparison of COSMO 5.0 and IFS 45r1 at kilometer-scale grid spacing. Geoscientific Model Development, 14(7):4617–4639, 2021. doi:10.5194/gmd-14-4617-2021.

[36] Г.И. Марчук. Методы расщепления. М.: Наука, 1988.

Реализация многошагового варианта метода сопряженных градиентов в глобальной модели атмосферы на сетке кубическая сфера*

Г.С. Гойман[1,2,3], В.В. Шашкин[1,2,3]

[1]Институт вычислительной математики им. Г.И. Марчука РАН
[2]Гидрометцентр России
[3]Московский физико-технический институт

В работе исследуется применимость s-шагового метода сопряжённых градиентов с предобусловливанием (sPCG) для решения систем линейных алгебраических уравнений (СЛАУ), возникающих при использовании неявно-явных схем интегрирования по времени в глобальных моделях динамики атмосферы. В качестве модельной задачи рассматривается уравнение типа Гельмгольца в тонкой сферической оболочке на сетке типа кубическая сфера.
Проведено численное исследование сходимости sPCG в зависимости от размера блока s, типа полиномиального базиса, порядка пространственной аппроксимации и параметра, соответствующего числу Куранта для быстрых горизонтальных волн. Показано, что при умеренных значениях s метод воспроизводит сходимость стандартного метода сопряжённых градиентов, однако при увеличении размера блока возможны ухудшение сходимости, стагнация или расходимость. Использование базиса Чебышёва существенно повышает численную устойчивость по сравнению со степенным базисом и позволяет применять большие размеры блока. Параллельная эффективность метода исследована в составе программного комплекса глобальной модели динамики атмосферы на суперкомпьютере Cray XC40. Показано, что в коммуникационно-доминирующем режиме sPCG превосходит стандартный метод сопряжённых градиентов за счёт сокращения числа глобальных параллельных коммуникаций. В представленных экспериментах максимальное ускорение при фиксированном числе ядер составило около 1.6×, а наилучшее время решения уменьшилось примерно на 20%. Полученные результаты показывают, что sPCG может рассматриваться как эффективный метод решения СЛАУ в задачах численного моделирования атмосферы на массивно-параллельных вычислительных системах.

Ключевые слова: метод сопряжённых градиентов, избегающие-коммуникаций методы Крылова, высокопроизводительные вычисления, кубическая сфера, моделирование атмосферы, численный прогноз погоды, климатическое моделирование

1. Введение

Уравнения гидротермодинамики атмосферы представляют собой типичный пример жёсткой системы, описывающей широкий спектр процессов с существенно различающимися временными масштабами. В связи с этим в численных моделях глобальной динамики атмосферы широко применяются неявно-явные методы интегрирования

*Работа выполнена в Институте вычислительной математики им. Г. И. Марчука Российской академии наук при поддержке Российского научного фонда (РНФ), проект № 24-71-00123 (https://rscf.ru/project/24-71-00123/)

по времени [1, 23, 32]. В таких схемах слагаемые, отвечающие за динамику быстрых процессов, в частности звуковых и инерционно-гравитационных волн, аппроксимируются неявно, что позволяет существенно увеличить максимально допустимый шаг интегрирования по времени по сравнению с полностью явными методами. Однако использование неявных аппроксимаций приводит к необходимости решения больших разреженных систем линейных алгебраических уравнений (СЛАУ) на каждом временном шаге. Поэтому эффективность и параллельная масштабируемость линейного решателя становятся критически важными факторами, определяющими общую вычислительную эффективность модели [10, 18, 19].

Наиболее распространёнными подходами к решению СЛАУ в моделях динамики атмосферы являются методы крыловского типа, например метод сопряжённых градиентов, а также геометрические многосеточные методы [3, 10, 11, 13, 18, 19, 25]. Многосеточные методы [2, 8, 33] обладают высокой скоростью сходимости, однако их эффективная параллельная реализация затруднена из-за необходимости выполнения вычислений на последовательности всё более грубых сеток, где отношение времени коммуникаций к времени арифметических операций резко возрастает. Кроме того, на гибридных архитектурах CPU+GPU объём вычислительной работы на грубых уровнях может оказаться недостаточным для полной загрузки ускорителей [20]. Эффективность геометрического многосеточного метода также существенно зависит от особенностей пространственной аппроксимации и может требовать дополнительной адаптации отдельных компонент метода к конкретному оператору и типу сетки [3, 12, 16].

Методы крыловского типа также широко применяются в метеорологических приложениях. Это связано с тем, что они сравнительно просты в реализации, в том числе в рамках безматричного подхода, не требуют сложной настройки свободных параметров и, как правило, в меньшей степени зависят от особенностей используемой пространственной аппроксимации. Однако существенным ограничением таких методов является необходимость выполнения глобальных коммуникаций, прежде всего редукций, на каждой итерации для вычисления скалярных произведений и норм. На массивно-параллельных вычислительных системах эти операции становятся одним из основных факторов, ограничивающих масштабируемость итерационного решателя.

Одним из подходов к снижению затрат на глобальные коммуникации являются так называемые избегающие-коммуникаций (communication-avoiding) s-шаговые (s-step) варианты методов крыловского типа [5, 6, 14, 17, 31]. Такая формулировка предполагает расширение подпространства Крылова сразу на s базисных векторов за одну итерацию метода, что позволяет сократить количество глобальных коммуникаций примерно в s раз. В точной арифметике такие алгоритмы эквивалентны соответствующим классическим одношаговым формулировкам, однако в арифметике с плавающей точкой их поведение может существенно отличаться. С ростом s возможны ухудшение численной устойчивости, замедление или стагнация сходимости, а также снижение достижимой точности [5]. Кроме того, различные варианты s-шаговых алгоритмов имеют разную вычислительную стоимость: некоторые из них требуют заметно большего числа умножений матрицы на вектор по сравнению со стандартными методами, что может нивелировать выигрыш от сокращения количества глобальных коммуникаций [17]. Поэтому применимость данного класса методов и возможность получения практического ускорения требуют отдельного исследования для каждого конкретного класса задач.

В настоящей работе исследуется применимость s-шагового метода сопряжённых градиентов к эллиптическим задачам, характерным для глобальных моделей дина-

мики атмосферы с неявно-явными аппроксимациями по времени. Насколько нам известно, ранее исследование применимости данных методов в контексте численного моделирования динамики атмосферы не проводилось.

В рамках исследования реализован s-шаговый метод сопряжённых градиентов с предобусловливанием (sPCG) [17] в параллельном программном комплексе глобальной модели атмосферы на сетке типа кубическая сфера, разрабатываемой в Институте вычислительной математики им. Г.И. Марчука РАН совместно с Гидрометцентром России [28]. Исследование свойств метода производится на модельных эллиптических уравнениях на сфере, характерных для задач, возникающих при использовании неявно-явных аппроксимаций по времени. Цель работы состоит в оценке применимости данного класса методов к рассматриваемым задачам и анализе их параллельной эффективности.

2. Модельная задача

2.1. Непрерывная постановка

Сначала проиллюстрируем основную идею применения неявно-явных методов интегрирования по времени на примере упрощённой системы уравнений динамики атмосферы в вертикальной плоскости, а затем сформулируем трёхмерную задачу в сферической оболочке, используемую в численных экспериментах.

Рассмотрим линеаризованные уравнения Эйлера в двумерной плоскости (x, z):

$$\frac{\partial u'}{\partial t} + u_0 \frac{\partial u'}{\partial x} + c_p \theta_0 \frac{\partial \pi'}{\partial x} = 0, \tag{1}$$

$$\frac{\partial w'}{\partial t} + u_0 \frac{\partial w'}{\partial x} + c_p \theta_0 \frac{\partial \pi'}{\partial z} + g \frac{\theta'}{\theta_0} = 0, \tag{2}$$

$$\frac{\partial \pi'}{\partial t} + u_0 \frac{\partial \pi'}{\partial x} + \frac{R_d}{c_v \pi_0} \left(\frac{\partial u'}{\partial x} + \frac{\partial w'}{\partial z} \right) = 0, \tag{3}$$

где u' и w' – возмущения горизонтальной и вертикальной компонент скорости, а π' – возмущение функции давления Экснера. Величины u_0, θ_0 и π_0 задают фоновое состояние, а c_p, c_v и R_d обозначают теплоёмкости и газовую постоянную сухого воздуха.

Поскольку звуковые волны распространяются существенно быстрее, чем происходит адвективный перенос, естественно аппроксимировать слагаемые, отвечающие за распространение звука, неявно. В рамках неявно-явного подхода градиенты давления в уравнениях импульса и дивергенция скорости в уравнении для давления учитываются неявно, тогда как адвективные слагаемые аппроксимируются явно или с использованием полулагранжевого подхода:

$$\frac{u'^{n+1} - u'^{n}}{\Delta t} = -\gamma c_p \theta_0 \frac{\partial \pi'^{n+1}}{\partial x} + f_u^n, \tag{4}$$

$$\frac{w'^{n+1} - w'^{n}}{\Delta t} = -\gamma c_p \theta_0 \frac{\partial \pi'^{n+1}}{\partial z} + f_w^n, \tag{5}$$

$$\frac{\pi'^{n+1} - \pi'^{n}}{\Delta t} = -\gamma \frac{R_d}{c_v \pi_0} \left(\frac{\partial u'^{n+1}}{\partial x} + \frac{\partial w'^{n+1}}{\partial z} \right) + f_\pi^n, \tag{6}$$

где f_u^n, f_w^n и f_π^n содержат все явно вычисляемые слагаемые, а γ – параметр схемы интегрирования по времени.

Вычисляя дивергенцию по формулам (4)–(5) и подставляя полученное выражение в (6), получаем эллиптическое уравнение типа Гельмгольца для функции давления Экснера на новом временном слое:

$$\pi'^{n+1} - \omega^2 \left(\frac{\partial^2 \pi'^{n+1}}{\partial x^2} + \frac{\partial^2 \pi'^{n+1}}{\partial z^2} \right) = f, \tag{7}$$

где

$$\omega^2 = \gamma^2 c_p \theta_0 \frac{R_d}{c_v \pi_0} \Delta t^2. \tag{8}$$

Вводя обозначение

$$c_s^2 = c_p \theta_0 \frac{R_d}{c_v \pi_0}, \tag{9}$$

где c_s имеет смысл скорости звука в рассматриваемой линеаризованной системе, уравнение (7) можно записать в компактном виде:

$$\psi - \omega^2 \left(\Delta_S \psi + \frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial \psi}{\partial r} \right) \right) = f. \tag{10}$$

Здесь ψ — искомая функция, имеющая смысл коррекции давления, f — заданная правая часть, а Δ_S — горизонтальный оператор Лапласа на сфере. Расчетная область:

$$\Omega = S \times [a, a + H], \tag{11}$$

где S – единичная сфера, a – радиус Земли, а $H \ll a$ – характерная толщина атмосферы. Параметр ω определяется шагом по времени и характерной скоростью быстрых волн c_h:

$$\omega^2 = (\gamma c_h \Delta t)^2. \tag{12}$$

На нижней и верхней границах сферической оболочки задаются однородные граничные условия типа Неймана:

$$\left. \frac{\partial \psi}{\partial r} \right|_{r=a} = 0, \qquad \left. \frac{\partial \psi}{\partial r} \right|_{r=a+H} = 0. \tag{13}$$

Уравнение (11) широко используется как модельная задача для исследования итерационных методов решения эллиптических уравнений, возникающих в численных моделях атмосферы [19, 20].

2.2. Пространственная аппроксимация

Для пространственной аппроксимации по горизонтали используется равноугольная гномоническая сетка типа кубическая сфера [24], а по вертикали – равномерная сетка. Горизонтальная сетка строится путём центрального проектирования граней куба на единичную сферу. Для опорной грани отображение задаётся формулой

$$\vec{r} = \frac{1}{\sqrt{1 + \tan^2 \alpha + \tan^2 \beta}} \begin{pmatrix} \tan \alpha & \tan \beta & 1 \end{pmatrix}^T,$$

где r — радиус-вектор в декартовых координатах, а $\alpha, \beta \in [-45°, 45°]$ — локальные угловые координаты на грани куба. Такая сетка обеспечивает квазиравномерное разрешение на сфере [29].

Горизонтальная аппроксимация строится на основе конечно-разностного метода со свойством суммирования по частям (SBP-FD) [7, 15, 21, 30]. Подробное описание соответствующей аппроксимации на сетке кубическая сфера приведено в [27]. Методы SBP-FD аппроксимируют производные таким образом, что на дискретном уровне сохраняется аналог формулы интегрирования по частям. На одномерных равномерных сетках стандартные SBP-операторы совпадают с центральными разностями во внутренних узлах и используют специальные односторонние аппроксимации вблизи границы, обеспечивающие выполнение SBP-свойства. На двумерных многоблочных логически прямоугольных сетках соответствующие операторы строятся с использованием тензорных произведений одномерных SBP-операторов.

В данной работе используются горизонтальные операторы дискретного градиента G и дивергенции D, построенные в [27]. В качестве базовых одномерных операторов рассматриваются SBP-FD аппроксимации порядков точности $2/1$ и $4/2$, где первая цифра соответствует порядку точности во внутренних узлах, а вторая — порядку точности вблизи границы блока. Дискретный горизонтальный оператор Лапласа задаётся в виде

$$L_h = DG.$$

Благодаря SBP-свойству оператор L_h является симметричным относительно дискретного скалярного произведения, индуцированного квадратурной матрицей конечно-разностного метода и метрическими коэффициентами криволинейных координат, и отрицательно полуопределённым.

По вертикали используется стандартная конечно-разностная аппроксимация второго порядка для радиальной части оператора в (11) с учётом однородных граничных условий типа Неймана (13). Получаемый вертикальный оператор L_z также является симметричным и отрицательно полуопределённым. При выбранном разбиении области он имеет блочно-диагональную структуру: каждому вертикальному столбцу соответствует независимый трёхдиагональный блок.

После дискретизации уравнения (11) получаем систему линейных алгебраических уравнений

$$(E - \omega^2(L_h + L_z))\psi = f, \tag{14}$$

где E – единичная матрица, а ψ и f – сеточные функции. Обозначим матрицу системы через

$$A = E - \omega^2(L_h + L_z).$$

Поскольку операторы L_h и L_z симметричны и отрицательно полуопределены в соответствующем дискретном скалярном произведении, матрица A является симметричной положительно определённой.

Так как $H \ll a$, вертикальный шаг сетки Δz существенно меньше характерного горизонтального шага Δx. Поэтому дискретный оператор обладает выраженной сеточной анизотропией с доминирующими связями по вертикали. Оценка степени анизотропии имеет вид

$$\beta \approx \left(\frac{\Delta x}{\Delta z}\right)^2 \gg 1,$$

где Δx и Δz — характерные горизонтальный и вертикальный шаги сетки соответственно. Такая анизотропия приводит к ухудшению обусловленности системы (14) и является важным структурным свойством, которое должен учитывать эффективный итерационный решатель [19].

В качестве предобуславливателя используется блочный метод Якоби с матрицей

$$M = E - \omega^2 L_z.$$

Применение M^{-1} сводится к решению набора независимых трёхдиагональных систем, соответствующих отдельным вертикальным столбцам. Эти системы решаются методом прогонки. Поскольку вертикальное направление не распределяется между MPI-процессами, применение предобуславливателя не требует межпроцессорных обменов.

3. Методы решения СЛАУ

В этом разделе кратко описываются стандартный метод сопряжённых градиентов с предобусловливанием (PCG) и s-шаговый аналог (sPCG) в формулировке [17].

Рассматривается решение большой разреженной симметричной положительно определённой системы

$$Ax = b, \tag{15}$$

где $A \in \mathbb{R}^{n \times n}$ и задан симметричный положительно определённый предобуславливатель M.

3.1. Метод сопряжённых градиентов с предобусловливанием

Алгоритм 1 описывает стандартную формулировку PCG. Каждая итерация требует одного умножения разреженной матрицы на вектор, одного применения предобуславливателя и двух скалярных произведений.

Алгоритм 1 Метод сопряжённых градиентов с предобуславливанием (PCG)

Вход: матрица A, правая часть b, предобуславливатель M^{-1}, начальное приближение $x^{(0)}$
$r^{(0)} \leftarrow b - Ax^{(0)}$
$u^{(0)} \leftarrow M^{-1} r^{(0)}$
$p^{(0)} \leftarrow u^{(0)}$
for $i = 0, 1, 2, \ldots,$ до выполнения критерия остановки **do**
    $\alpha^{(i)} \leftarrow \dfrac{(r^{(i)})^T u^{(i)}}{(p^{(i)})^T A p^{(i)}}$ // Глобальная редукция
    $x^{(i+1)} \leftarrow x^{(i)} + \alpha^{(i)} p^{(i)}$
    $r^{(i+1)} \leftarrow r^{(i)} - \alpha^{(i)} A p^{(i)}$
    $u^{(i+1)} \leftarrow M^{-1} r^{(i+1)}$
    $\beta^{(i+1)} \leftarrow \dfrac{(r^{(i+1)})^T u^{(i+1)}}{(r^{(i)})^T u^{(i)}}$ // Глобальная редукция
    $p^{(i+1)} \leftarrow u^{(i+1)} + \beta^{(i+1)} p^{(i)}$
end for

3.2. s-шаговый метод сопряжённых градиентов (sPCG)

В методе sPCG s последовательных итераций объединяются в одну внешнюю итерацию. Обозначим через k номер внешней итерации.

В начале внешней итерации k строятся блочные базисные матрицы:

$$S^{(k)} = \left[\phi_0(AM^{-1})\, r^{(k)},\ \phi_1(AM^{-1})\, r^{(k)},\ \ldots,\ \phi_s(AM^{-1})\, r^{(k)}\right] \in \mathbb{R}^{n\times(s+1)}, \tag{16}$$

$$U^{(k)} = \left[\phi_0(M^{-1}A)\, u^{(k)},\ \phi_1(M^{-1}A)\, u^{(k)},\ \ldots,\ \phi_{s-1}(M^{-1}A)\, u^{(k)}\right] \in \mathbb{R}^{n\times s}, \tag{17}$$

где $u^{(k)} = M^{-1}r^{(k)}$, а $\{\phi_\ell\}$ — выбранный полиномиальный базис (например, степенной, Ньютона или Чебышёва), причём $\phi_\ell(A)$ обозначает соответствующий матричный полином. Через $R^{(k)} \in \mathbb{R}^{n\times s}$ обозначим матрицу, составленную из первых s столбцов $S^{(k)}$.

Для общего полиномиального базиса полиномы удовлетворяют трёхчленному рекуррентному соотношению:

$$\phi_0(z) = 1, \qquad \phi_1(z) = \frac{(z-\theta_0)\phi_0(z)}{\gamma_0}, \tag{18}$$

а при $\ell \geq 2$

$$\phi_\ell(z) = \frac{(z-\theta_{\ell-1})\phi_{\ell-1}(z) + \mu_{\ell-2}\phi_{\ell-2}(z)}{\gamma_{\ell-1}}. \tag{19}$$

Коэффициенты θ_j, γ_j и μ_j определяются типом базиса. Степенной базис получается как частный случай при $\theta_j = 0$, $\gamma_j = 1$, $\mu_j = 0$.

Выбор базиса критичен для численной устойчивости. Степенной базис наиболее прост в реализации, однако при умеренно больших значениях s (обычно $s >= 5$) может приводить к плохой обусловленности базисных матриц и, как следствие, к замедлению или стагнации сходимости [5,6,37]. Альтернативные базисы, такие как полиномы Чебышёва или Ньютона, существенно улучшают устойчивость, но требуют информации о границах спектра матрицы. В данной работе рассматриваются степенной базис и базис Чебышёва.

Внутри каждой внешней итерации блочные обновления параметризуются вектором $a^{(k)}$ длины s и матрицей $B^{(k)}$ размера $s \times s$, связывающей текущие поисковые направления с предыдущими. Для вычисления этих величин требуется компактное представление произведения $AU^{(k)}$. Для произвольного полиномиального базиса оно записывается через матрицу перехода $\mathcal{B} \in \mathbb{R}^{(s+1)\times s}$:

$$AU^{(k)} = S^{(k)}\mathcal{B}. \tag{20}$$

Описание метода sPCG приведено в алгоритме 2.

3.3. Сравнение вычислительной работы

Сравним вычислительную стоимость s итераций метода PCG и одной внешней итерации sPCG. Оценки приведены в соответствии с [17].

За s итераций стандартный PCG выполняет s умножений разреженной матрицы на вектор и s применений предобусловливателя. Остальные векторные операции имеют стоимость порядка $8sn$ операций с плавающей точкой, где n — размер системы. При этом в базовой реализации PCG на каждой итерации необходимо вычислять скалярные произведения для коэффициентов метода, что приводит к $\mathcal{O}(s)$ глобальным редукциям за блок из s итераций.

Алгоритм 2 s-шаговый PCG

Вход: A, b, M^{-1}, $x^{(0)}$, размер блока s, допуск ε
$r^{(0)} \leftarrow b - Ax^{(0)}$
$P^{(-1)} \leftarrow 0 \in \mathbb{R}^{n\times s}, \quad AP^{(-1)} \leftarrow 0 \in \mathbb{R}^{n\times s}$
for $k = 0, 1, 2, \ldots,$ до выполнения критерия остановки **do**
    $u^{(k)} \leftarrow M^{-1}r^{(k)}$
    Построить $S^{(k)}$ по формуле (16) с использованием рекуррентных соотношений (18)–(19)
    Построить $U^{(k)}$ по формуле (17) с использованием рекуррентных соотношений (18)–(19)
    $R^{(k)} \leftarrow S^{(k)}(:, 1:s)$
    $AU^{(k)} \leftarrow S^{(k)}\mathcal{B}$ // Формула (20)
    $m^{(k)} \leftarrow (R^{(k)})^T u^{(k)}$ // Первый столбец $(U^{(k)})^T S^{(k)}$
    if $k \neq 0$ **then**
        $C^{(k)} \leftarrow -\mathcal{B}^T (S^{(k)})^T P^{(k-1)}$
        Решить систему $W^{(k-1)}B^{(k)} = C^{(k)}$ относительно $B^{(k)}$
    else
        $C^{(k)} \leftarrow 0, \quad B^{(k)} \leftarrow 0$
    end if
    $W^{(k)} \leftarrow (U^{(k)})^T S^{(k)}\mathcal{B} - C^{(k)}B^{(k)}$
    Решить систему $W^{(k)}a^{(k)} = m^{(k)}$ относительно $a^{(k)}$
    $P^{(k)} \leftarrow U^{(k)} + P^{(k-1)}\, B^{(k)}$
    $AP^{(k)} \leftarrow AU^{(k)} + AP^{(k-1)}\, B^{(k)}$
    $x^{(k+1)} \leftarrow x^{(k)} + P^{(k)}a^{(k)}$
    $r^{(k+1)} \leftarrow r^{(k)} - AP^{(k)}a^{(k)}$
end for

Метод sPCG за одну внешнюю итерацию также выполняет s умножений матрицы на вектор и s применений предобусловливателя. Основной дополнительный вклад в вычислительную работу sPCG связан с локальными операциями сложность которых порядка $\mathcal{O}(s^2 n)$. Для степенного базиса стоимость оставшейся части алгоритма за блок из s шагов составляет

$$(6s^2 + 6s)n$$

операций с плавающей точкой, а для произвольного полиномиального базиса —

$$(6s^2 + 16s - 4)n.$$

3.4. Параллельная реализация

Рассматриваемые решатели реализованы и протестированы в рамках программного комплекса динамического ядра глобальной модели атмосферы нового поколения, разрабатываемого в ИВМ РАН совместно с Гидрометцентром России [28]. Основные принципы архитектуры программного комплекса описаны в [28], детали параллельной реализации – в [26], а организация инфраструктуры линейных решателей – в [11].

Модель реализована на языке Fortran и использует MPI для распараллеливания на системах с распределённой памятью. Расчётная область декомпозируется только по горизонтали и разбивается на логически прямоугольный набор подобластей на сетке кубическая сфера, распределяемых между MPI-процессами.

4. Численные эксперименты

4.1. Исследование сходимости

Сначала исследуем сходимость метода sPCG и сравним её со сходимостью стандартного PCG. Анализируется влияние следующих факторов:

- размера блока s;
- типа полиномиального базиса: степенного или чебышёвского;
- порядка пространственной аппроксимации: SBP 2/1 или SBP 4/2;
- значения параметра ω^2 в уравнении (11).

Число Куранта для быстрых волн, распространяющихся в горизонтальном направлении, определяется как

$$\mathrm{CFL} = \frac{c_h \Delta t}{\Delta x}, \tag{21}$$

где c_h — характерная скорость быстрых горизонтальных волн, Δt — шаг по времени, а Δx — характерный горизонтальный шаг сетки. Из вывода в разделе 2 и определения параметра ω следует связь

$$\omega = \gamma c_h \Delta t = \gamma \mathrm{CFL} \Delta x. \tag{22}$$

Для атмосферных моделей с явным эйлеровым описанием адвекции шаг по времени обычно ограничивается условием устойчивости по адвективному переносу, поэтому число Куранта для быстрых волн, как правило, не превышает 2–3. В качестве характерного значения для такого режима в экспериментах используется $\mathrm{CFL} = 3$. При $\gamma = 1/2$ это даёт $\omega = 1.5\Delta x$. Далее этот режим обозначается как EU.

Полулагранжевы схемы переноса допускают существенно большие шаги по времени, при которых число Куранта для быстрых волн может достигать значений порядка 10–15. В качестве характерного значения для такого режима используется $\mathrm{CFL} = 15$, что при $\gamma = 1/2$ соответствует $\omega = 7.5\Delta x$. Далее этот режим обозначается как SL.

Во всех экспериментах ниже сравнивается эволюция относительной нормы невязки

$$\frac{\|b - Ax^{(k)}\|_2}{\|b\|_2}$$

для стандартного PCG и sPCG при различных значениях размера блока s. Правая часть b выбирается случайным образом. Для каждой конфигурации выполняется 100 запусков с различными правыми частями. Далее приводятся средняя история сходимости и наихудшее поведение среди этих запусков. Во всех расчётах начальное приближение полагается равным нулю: $x^{(0)} = 0$.

Тесты сходимости выполняются на сетке с $N_x = 64$ узлами вдоль каждой грани и $N_z = 32$ вертикальными уровнями. Результаты расчетов представлены на рис. 1. Сплошные линии соответствуют среднему значению относительной невязки по 100 запускам, а верхняя граница заштрихованной области показывает наибольшую за 100 запусков норму невязки. Видно несколько характерных тенденций. Во-первых, увеличение параметра ω в режиме SL приводит к более медленной сходимости стандартного PCG, что согласуется с ростом числа обусловленности системы. Во-вторых, для PCG и sPCG аппроксимация SBP 4/2 обычно приводит к более сложной для итерационного решателя системе, чем аппроксимация SBP 2/1. В-третьих, при малых и умеренных значениях s метод sPCG хорошо воспроизводит сходимость PCG, однако после достижения некоторого зависящего от задачи уровня невязки скорость

сходимости замедляется, а при слишком больших s может наблюдаться расходимость метода.

В реальных атмосферных расчётах, как правило, достаточно уменьшения относительной невязки до уровня порядка 10^{-4}–10^{-5} [3, 18, 19]. Из результатов для степенного базиса видно, что в режиме EU можно использовать блоки вплоть до $s \approx 9$ без заметного ухудшения сходимости на практически значимых уровнях точности. В более жёстком режиме SL пригодные значения обычно ограничены диапазоном $s \approx 5$–6. Для базиса Чебышёва представлены результаты для наиболее сложной конфигурации (SL, 4/2); видно, что этот базис позволяет устойчиво использовать существенно большие значения s, примерно вдвое увеличивая допустимый размер блока по сравнению со степенным базисом.

4.2. Параллельная эффективность

В данном разделе сравнивается параллельная и вычислительная эффективность sPCG в сравнении с PCG.

В качестве правых частей используются случайные векторы, рассматривается среднее за 100 последовательных запусков время решения. Во всех расчётах начальное приближение полагается равным нулю. Критерием остановки служит уменьшение относительной нормы невязки в 10^{-5} раз.

Рассматриваются задачи на сетках с числом узлов вдоль ребра куба $N_x \in \{96, 180\}$ и числом вертикальных уровней $N_z \in \{32, 64\}$. Конфигурацию сетки будем обозначать как CN_xLN_z; например, запись C96L64 соответствует $N_x = 96$ и $N_z = 64$. Все тесты выполняются для режима SL, то есть при $\omega = 7.5\Delta x$, и для SBP-FD аппроксимаций порядков 2/1 и 4/2. Для sPCG рассматриваются размеры блока $s = 2, \ldots, 5$; большие значения s в данном наборе экспериментов не дают преимущества согласно исследованию сходимости из раздела 4.1.

Все тесты выполнены на суперкомпьютере Cray XC40 Главного вычислительного центра Росгидромета. Система использует интерконнект Cray Aries. Каждый вычислительный узел оснащён двумя процессорами Intel Xeon E5-2697 v4: по 18 ядер на сокет, 36 ядер на узел, с 45 MB Intel Smart Cache на процессор. В представленных экспериментах использовалось до 6480 CPU-ядер. Код компилировался с использованием Intel Fortran Compiler версии 19.1.3.304.

На рис. 2–3 показано ускорение PCG и sPCG относительно времени работы PCG на базовом числе MPI-процессов для соответствующей задачи и выбранной дискретизации. Для всех исследованных разрешений и порядков аппроксимации наблюдается одна и та же качественная картина. При малом числе процессов стандартный PCG оказывается быстрее sPCG, поскольку стоимость глобальных коммуникаций ещё не доминирует, а дополнительные локальные операции sPCG приводят к накладным расходам, возрастающим с увеличением s. Однако при переходе в режим сильного масштабирования sPCG начинает превосходить стандартный PCG. В этом режиме sPCG расширяет область эффективного масштабирования, уменьшает наилучшее достижимое время решения и ослабляет деградацию производительности после достижения предела сильной масштабируемости, когда дальнейшее увеличение числа ядер уже не сокращает, а увеличивает время решения.

Во всех приведённых случаях наилучшие времена для sPCG достигаются при $s = 2$ или $s = 3$. Для $s = 4$ и $s = 5$ отставание от оптимальных значений s уменьшается с ростом числа ядер, что указывает на потенциальную пользу больших размеров блока при более высоких разрешениях и большем числе вычислительных узлов. Максимальное уменьшение лучшего времени решения составляет примерно 20%,

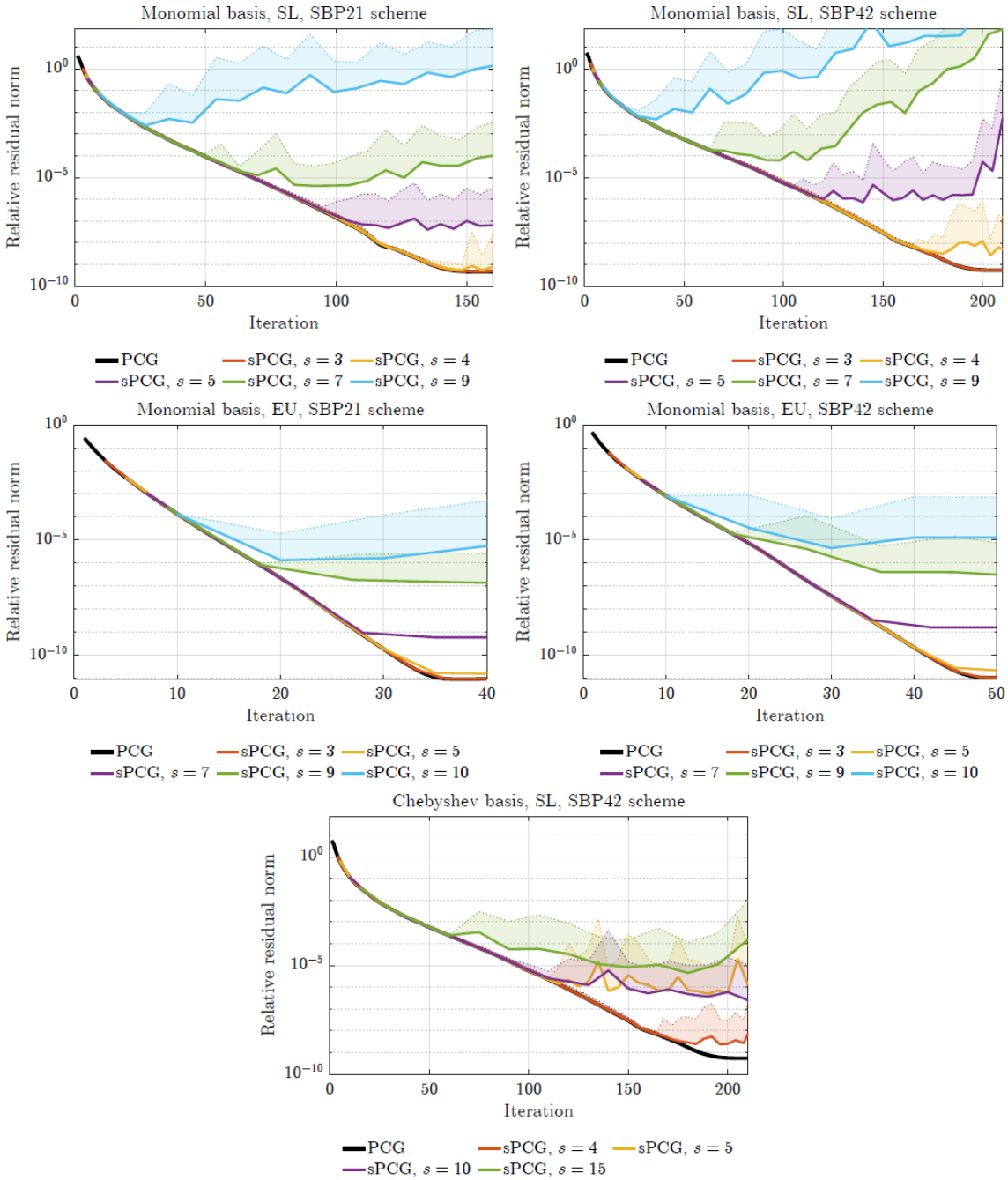


Рис. 1. Сходимость PCG и sPCG при различных размерах блока s. Верхний ряд: степенной базис, режим SL, SBP 2/1 (слева) и 4/2 (справа). Средний ряд: степенной базис, режим EU, SBP 2/1 (слева) и 4/2 (справа). Нижний ряд: базис Чебышёва, режим SL, SBP 4/2. Сплошные линии показывают среднюю истинную относительную невязку по 100 случайным правым частям; заштрихованная область соответствует наихудшей невязке среди этих запусков.

а максимальное ускорение при фиксированном числе ядер — около 1.6×.

Следует также отметить, что выигрыш от сокращения числа глобальных редукций частично маскируется снижением эффективности параллельного умножения матрицы на вектор в режиме сильного масштабирования. Поскольку умножение матрицы на вектор обычно хорошо масштабируется в режиме слабого масштабирования, тогда как стоимость глобальных редукций существенно возрастает с увеличением числа

процессов, можно ожидать, что преимущество sPCG будет более выраженным для задач большего размера и при использовании большего числа вычислительных ядер.

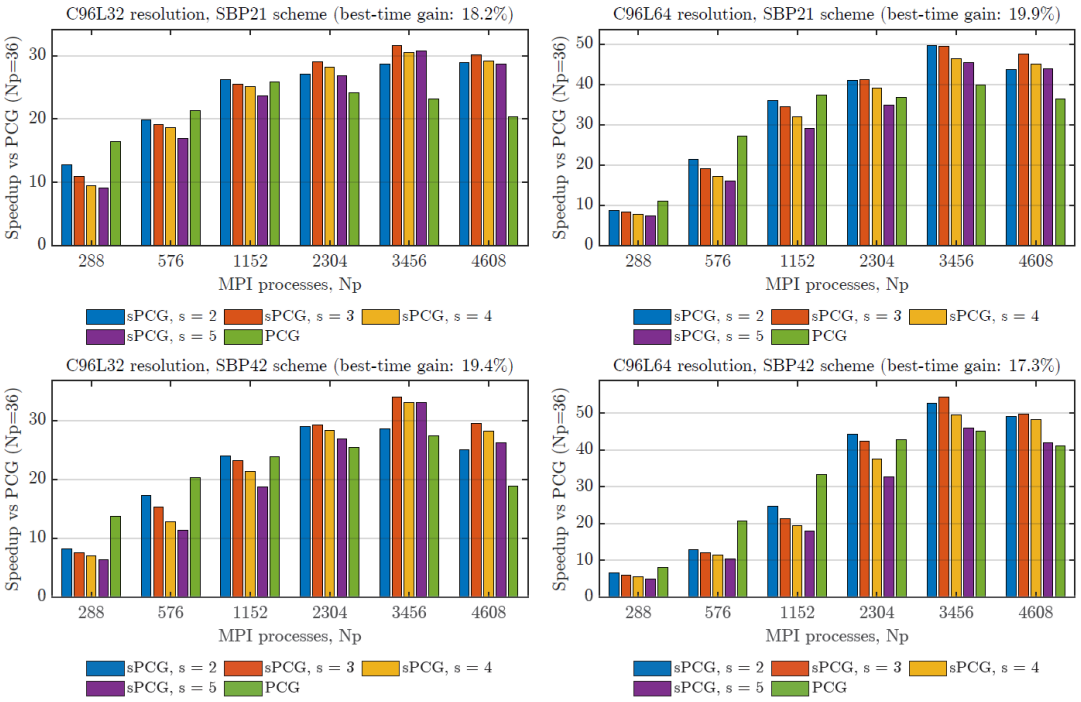


Рис. 2. Ускорение PCG и sPCG (размеры блока $s = 2$–5) относительно PCG на базовом числе процессов для сеток C96L32 и C96L64: сверху — дискретизация SBP 2/1, снизу — SBP 4/2. Режим SL.

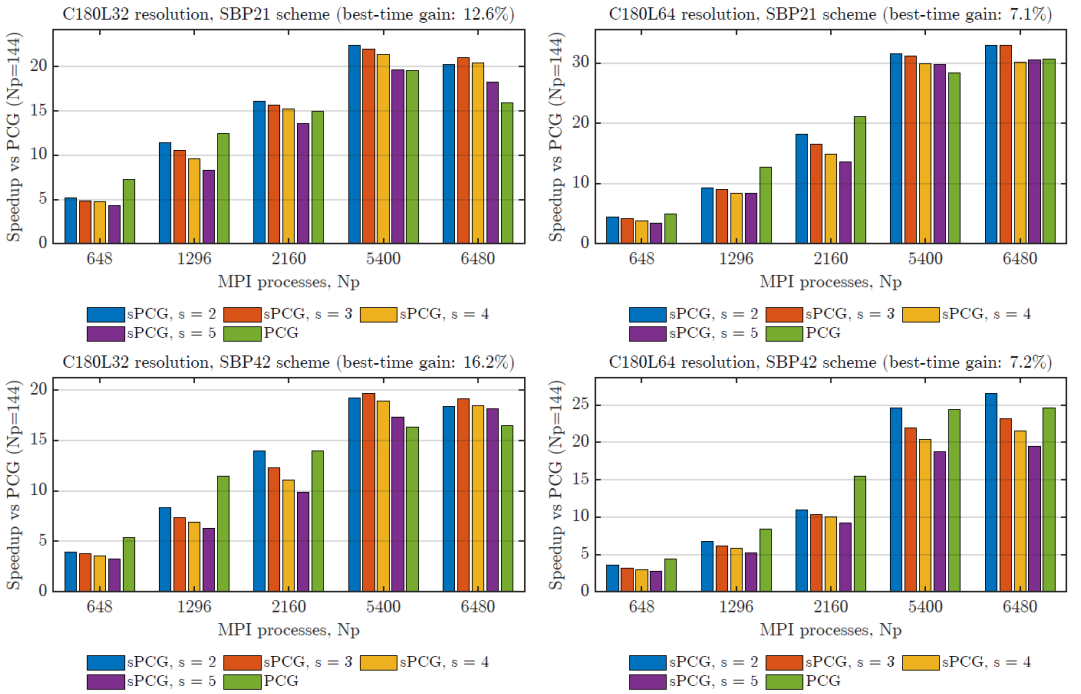


Рис. 3. Ускорение PCG и sPCG (размеры блока $s = 2$–5) относительно PCG на базовом числе процессов для сеток C180L32 и C180L64: сверху — дискретизация SBP 2/1, снизу — SBP 4/2. Режим SL.

Заключение

В работе исследована применимость s-шагового метода сопряжённых градиентов (sPCG) к СЛАУ, возникающим при неявно-явной аппроксимации по времени уравнений глобальной динамики атмосферы. Исследование включало анализ сходимости в режимах, характерных для атмосферного моделирования, а также оценку параллельной эффективности метода.

Численные эксперименты показали, что практическая применимость sPCG определяется размером блока s, типом полиномиального базиса и режимом задачи, задаваемым параметром ω или, эквивалентно, числом Куранта для быстрых горизонтальных волн. Для всех протестированных конфигураций sPCG воспроизводит сходимость стандартного метода сопряжённых градиентов (PCG) до некоторого зависящего от задачи уровня невязки, после чего при достаточно больших s может наблюдаться ухудшение сходимости, стагнация или расходимость. Допустимый диапазон значений s уменьшается в более жёстких режимах, соответствующих большим значениям ω и более высокому порядку пространственной аппроксимации. Использование базиса Чебышёва существенно повышает численную устойчивость и позволяет применять примерно в два раза большие размеры блока по сравнению со степенным базисом.

Измерения параллельной эффективности на системе Cray XC40 показали, что при малом числе MPI-процессов стандартный PCG оказывается быстрее, поскольку вклад глобальных коммуникаций в общее время решения ещё невелик, а дополнительные локальные операции sPCG приводят к заметным накладным расходам. Однако в коммуникационно-доминирующем режиме sPCG превосходит PCG. В представленных экспериментах наилучшие времена решения для sPCG достигались при $s = 2$ или $s = 3$; максимальное ускорение по сравнению с PCG при фиксированном числе ядер составило около $1.6\times$, а наилучшее время решения уменьшилось примерно на 20%.

Таким образом, рассмотренный метод расширяет область эффективной масштабируемости в режимах, где стандартный PCG перестаёт эффективно ускоряться, и может рассматриваться как дополнение к PCG при проведении расчётов на массивно-параллельных вычислительных системах.

Список литературы

[1] Ascher U. M., Ruuth S. J., Spiteri R. J. Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations // Appl. Numer. Math. 1997. Vol. 25, No. 2–3. P. 151–167.

[2] Briggs W. L., Henson V. E., McCormick S. F. A Multigrid Tutorial. Philadelphia: SIAM, 2000.

[3] Buckeridge S., Scheichl R. Parallel geometric multigrid for global weather prediction // Numer. Linear Algebra Appl. 2010. Vol. 17, No. 2-3. P. 325–342.

[4] Carson E., Gergelits T., Yamazaki I. Mixed precision s-step Lanczos and conjugate gradient algorithms // Numer. Linear Algebra Appl. 2022. Vol. 29, No. 3. Art. e2425.

[5] Carson E. C. Communication-Avoiding Krylov Subspace Methods in Theory and Practice: PhD Dissertation. University of California, Berkeley, 2015.

[6] Chronopoulos A. T., Gear C. W. On the efficient implementation of preconditioned s-step conjugate gradient methods on multiprocessors with memory hierarchy // Parallel Comput. 1989. Vol. 11, No. 1. P. 37–53.

[7] Del Rey Fernández D. C., Hicken J. E., Zingg D. W. Review of summation-by-parts

operators with simultaneous approximation terms for the numerical solution of partial differential equations // Computers and Fluids. 2014. Vol. 95. P. 171–196.

[8] Fedorenko R. P. A relaxation method for solving elliptic difference equations // USSR Comput. Math. and Math. Phys. 1962. Vol. 1, No. 4. P. 1092–1096.

[9] Geoffroy O., Saint-Martin D. The ARP-GEM1 global atmosphere model: Description, speedup analysis, and multiscale evaluation up to 6 km // J. Climate. 2025. Vol. 38, No. 18. P. 4739–4762.

[10] Gillard M., Szmelter J., Coccetta F. Preconditioning elliptic operators in high-performance all-scale atmospheric models on unstructured meshes // J. Comput. Phys. 2025. Vol. 520. Art. 113503.

[11] Goyman G., Shashkin V. Implementation of elliptic solvers within ParCS parallel framework // Commun. Comput. Inf. Sci. 2021. Vol. 1510. P. 137–147.

[12] Goyman G. S., Shashkin V. V. A geometric multigrid solver for collocated discretizations in atmospheric dynamics on a cubed-sphere grid // Lobachevskii J. Math. 2025. Vol. 46, No. 8. P. 3662–3677.

[13] Heikes R. P., Randall D. A., Konor C. S. Optimized icosahedral grids: Performance of finite-difference operators and multigrid solver // Mon. Weather Rev. 2013. Vol. 141, No. 12. P. 4450–4469.

[14] Hoemmen M. Communication-Avoiding Krylov Subspace Methods: PhD Dissertation. University of California, Berkeley, 2010.

[15] Kreiss H.-O., Scherer G. Finite element and finite difference methods for hyperbolic partial differential equations // In: Mathematical Aspects of Finite Elements in Partial Differential Equations. Elsevier, 1974. P. 195–212.

[16] Larsson J., Lien F. S., Yee E. Conditional semicoarsening multigrid algorithm for the Poisson equation on anisotropic grids // J. Comput. Phys. 2005. Vol. 208, No. 1. P. 368–383.

[17] Mayer V., Gansterer W. N. Numerical properties and scalability of s-step preconditioned conjugate gradient methods // In: Proceedings of the SC'25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2025. P. 1779–1789.

[18] Maynard C., Melvin T., Müller E. H. Multigrid preconditioners for the mixed finite element dynamical core of the LFRic atmospheric model // Quart. J. Roy. Met. Soc. 2020. Vol. 146, No. 733. P. 3917–3936.

[19] Müller E., Scheichl R. Massively parallel solvers for elliptic partial differential equations in numerical weather and climate prediction // Quart. J. Roy. Met. Soc. 2014. Vol. 140, No. 685. P. 2608–2624.

[20] Müller E. H., Scheichl R., Vainikko E. Petascale solvers for anisotropic PDEs in atmospheric modelling on GPU clusters // Parallel Comput. 2015. Vol. 50. P. 53–69.

[21] Olsson P. Summation by parts, projections, and stability I // Math. Comp. 1995. Vol. 64, No. 211. P. 1035–1065.

[22] Palmer T. N. More reliable forecasts with less precise computations: a fast-track route to cloud-resolved weather and climate simulators? // Phil. Trans. R. Soc. A. 2014. Vol. 372, No. 2018.

[23] Robert A. Integration of a spectral model of the atmosphere by the implicit method // In: Procs. of WMO/IUGG Symp. on Numerical Weather Prediction. 1969.

[24] Sadourny R. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids // Mon. Weather Rev. 1972. Vol. 100, No. 2. P. 136–144.

[25] Sandbach S., Thuburn J., Vassilev D., Duda M. G. A semi-implicit version of the MPAS-Atmosphere dynamical core // Mon. Weather Rev. 2015. Vol. 143, No. 9. P.

3838–3855.

[26] Shashkin V., Goyman G. Parallel efficiency of time-integration strategies for the next generation global weather prediction model // In: Russian Supercomputing Days. Springer, 2020. P. 285–296.

[27] Shashkin V. V., Goyman G. S., Tolstykh M. A. Summation-by-parts finite-difference shallow water model on the cubed-sphere grid. Part I: Non-staggered grid // J. Comput. Phys. 2023. Vol. 474. Art. 111797.

[28] Shashkin V. V., Goyman G. S., Tretyak I. D. Development of the next-generation atmosphere dynamics model in Russia: Current state and prospects // Lobachevskii J. Math. 2024. Vol. 45, No. 7. P. 3159–3172.

[29] Staniforth A., Thuburn J. Horizontal grids for global weather and climate prediction models: a review // Quart. J. Roy. Met. Soc. 2012. Vol. 138. P. 1–26.

[30] Strand B. Summation by parts for finite difference approximations for d/dx // J. Comput. Phys. 1994. Vol. 110, No. 1. P. 47–67.

[31] Toledo S. A. Quantitative Performance Modeling of Scientific Computations: PhD Dissertation. Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1995.

[32] Mengaldo G., Wyszogrodzki A., Diamantakis M., Lock S.-J., Giraldo F. X., Wedi N. P. Current and emerging time-integration strategies in global numerical weather and climate prediction // Arch. Comput. Methods Eng. 2019. Vol. 26. P. 663–684.

[33] Trottenberg U., Oosterlee C. W., Schuller A. Multigrid. Academic Press, 2000.

[34] Vana F., Duben P., Lang S., Palmer T., Leutbecher M., Salmond D., Carver G. Single precision in weather forecasting models: An evaluation with the IFS // Mon. Weather Rev. 2017. Vol. 145, No. 2. P. 495–502.

[35] Xu Z., Alonso J. J., Darve E. A numerically stable communication-avoiding s-step GMRES algorithm // SIAM J. Matrix Anal. Appl. 2024. Vol. 45, No. 4. P. 2039–2074.

[36] Yang X., Zhou S., Zhou S., Song Z., Liu W. A barotropic solver for high-resolution ocean general circulation models // J. Mar. Sci. Eng. 2021. Vol. 9, No. 4. Art. 421.

[37] Zamarashkin N. L., Tyrtyshnikov E. E. Eigenvalue estimates for Hankel matrices // Sb. Math. 2001. Vol. 192, No. 4. P. 537–550.

Суперкомпьютерное моделирование напряжений в тонких оптических пленках при различных температурах подложки

Ф.В. Григорьев[1,2,a], В.Б. Сулимов[1,b], А.В. Тихонравов[1,2,c]

[1]Московский государственный университет имени М.В. Ломоносова, Научно-исследовательский вычислительный центр, Россия, 119991, Москва

[2]Московский центр фундаментальной и прикладной математики, Россия, 119234, Москва

[a]afedor.grigoriev@rcc.msu.ru
[b]bvs@dimonta.com
[c]ctikh@srcc.msu.ru

Методом молекулярной динамики проведен суперкомпьютерный расчет механических напряжений в тонких пленках диоксида кремния и диоксида титана при различных температурах процесса напыления. Расчет напряжений требует атомистических кластеров больших размеров, что обуславливает необходимость использования технологий параллельных вычислений. Показано, что характер и величины напряжений существенно зависят от пленкообразующего материала, температуры подложки и энергии напыляемых атомов. Проведен анализ численной эффективности суперкомпьютерных вычислений с учетом больших размеров области атомистического моделирования, и определены способы повышения численной эффективности.

Ключевые слова: Атомистическое моделирование, тонкие пленки, высокопроизводительные расчеты, напряжения в тонких пленках

1. Введение

Многослойные оптические покрытия, изготавливаемые для различных целей, состоят из нескольких десятков слоев с чередующимися величинами показателя преломления [1]. Структуру покрытий схематически можно представить следующим образом: S|HLH...L|A, где S, H, L и A — подложка, слой с высоким показателем преломления (например, Ta_2O_5), слой с низким показателем преломления (например, SiO_2) и воздух. В качестве материала с низким показателем преломления часто используют SiO_2, а в качестве материала с высоким — TiO_2.

Одним из ключевых факторов, влияющих на свойства покрытий и ограничивающих их использование в лазерном оборудовании сверхвысокой мощности, являются механические напряжения. Такие напряжения возникают из-за различий в физических свойствах пленкообразующих материалов. Напряжения могут приводить к деформации покрытий [2], частичному расслоению соседних слоев [3], увеличению концентрации дефектов, влияющих на оптические свойства покрытий, и другим эффектам, ухудшающим характеристики многослойных покрытий [4]. В связи с этим

определение технологических условий, при которых могут быть изготовлены многослойные покрытия с низкими абсолютными значениями механических напряжений, является актуальной задачей в оптической и оптоэлектронной промышленности.

Для теоретического изучения зависимости напряжений в растущей пленке от условий напыления можно использовать методы атомистического моделирования – такие, как метод молекулярной динамики (МД), метод Монте Карло (МК), и другие. При использовании этих методов тензор напряжений рассчитывается на основе траектории моделирования по величинам скоростей и координат каждого атома в кластере. В настоящее время такой подход широко используется при расчете напряжений в различных материалах, включая тонкие пленки. В [5] метод МД использовался для исследования дефектов и напряжений, вызванных бомбардировкой пленки Mo высокоэнергетическими ионами Ar. Было выявлено, что размеры кристаллических зерен в пленках уменьшаются с ростом энергии ионов, что приводит к увеличению концентрации междоузельных дефектов. Также увеличивается значение сжимающего напряжения (compressive stress). При этом изменение угла падения осаждаемых атомов влияет на напряжения и структуру пленок незначительно [5]. Структура и механические свойства тонких пленок DLC (Diamond-like carbon)/Ni-DLC были изучены в [6] с использованием метода МД. Показано, что легирование Ni и специально подобранные параметры напыления снижают значение напряжений в пленках DLC. В [7] было проведено моделирование напыления методом МД покрытий из тантала и нитрида тантала, изготовленных химическим осаждением из газовой фазы. На основе экспериментальных результатов и моделирования сделан вывод, что межфазное напряжение в покрытии Ta/TaN существенно ниже, чем в покрытии Ta/(нержавеющая сталь). Зависимость напряжений пленки диоксида кремния от толщины пленок изучалась в [8] с использованием метода МД с эмпирическим силовым полем DESIL [9]. В случае нормального осаждения (атомы падают перпендикулярно подложке) напряжение является сжимающим и незначительно изменяется с увеличением толщины пленки. Осаждение под большими углами (glancing angle deposition) приводит к уменьшению абсолютных значений напряжений в несколько раз [9]. Напряжения в алмазоподобных углеродных пленках были рассчитаны в [10] с использованием МД моделирования. Показано, что напряжения уменьшаются с увеличением угла напыления. Авторы [10] объяснили это уменьшение релаксацией напряженных связей C–C. Расчет механических напряжений в пленках TiO_2, SiO_2 и TiO_2–SiO_2 выполнен в [11] с использованием МД моделирования. Найдено, что при нормальном осаждении, когда угол падения не превышает 30°, главные компоненты тензора напряжений отрицательны, что соответствует сжимающему напряжению. Абсолютные значения напряжений незначительно изменяются в указанных интервалах угла напыления. Увеличение угла от 30° приводит к заметному уменьшению абсолютных значений тензора напряжений, а при напылении под скользящим углом напряжения уменьшаются в несколько раз. Кроме того, в [11] наблюдалась значительная анизотропия напряжений в плоскости подложки.

Таким образом, к настоящему времени накоплен значительный опыт в области расчета механических напряжений в тонких пленках различного состава, напыленных в различных условиях, с использованием методов атомистического моделирования. В данной работе мы используем МД моделирование для изучения зависимости напряжений в растущих пленках диоксида кремния и диоксида титана от температуры подложки. Также исследуется влияние охлаждения подложки после осаждения. Важно отметить, что для расчета тензора напряжений с использованием МД моделирования необходимы кластеры больших размеров, состоящие из десятков и сотен

тысяч атомов. Это связано с зависимостью величин напряжений от толщины пленок. Чтобы учесть этот эффект, характерные размеры кластеров моделирования должны достигать нескольких десятков нанометров [9]. Такое моделирование может быть выполнено только с использованием технологии параллельных вычислений. Эффективность суперкомпьютерного моделирования, проведенного в настоящей работе, также анализируется.

2. Метод моделирования

Процедура МД моделирования с целью расчета напряжений в напыляемой пленке при различных температурах состоит из следующих этапов:

1. Моделирование напыления тонкой пленки диоксида кремния (диоксида титана) на горячей подложке. Компоненты тензора напряжений рассчитываются по мере роста толщины пленки с шагом по толщине, приблизительно равным 3 нм;
2. Конечный кластер пленки толщиной около 18 нм охлаждается от температуры напыления до комнатной температуры (300 K) в течение 200 пс. Скорость охлаждения постоянна;
3. Проводится МД моделирование напыленного кластера при температуре 300 K в течение 200 пс. Затем выполняется МД моделирование в течение 200 пс, и компоненты тензора напряжений рассчитываются на основе полученной траектории моделирования.

Описанная процедура схематически показана на Рис. 1. Моделирование напыления (Рис. 1, левая сторона) воспроизводит технологические условия метода физического напыления из газовой фазы (Physical Vapour Deposition, PVD) [1]. Пленка диоксида кремния (диоксида титана) формируется в вакуумной камере при низком давлении атомами, извлеченными из мишени [12]. Эти атомы движутся в камере от мишени к подложке, реагируют с молекулами кислорода и формируют новые слои пленки. Как правило, характерный размер подложки существенно меньше расстояния от мишени до подложки. При таком условии МД моделирование процесса напыления проходит следующим образом:

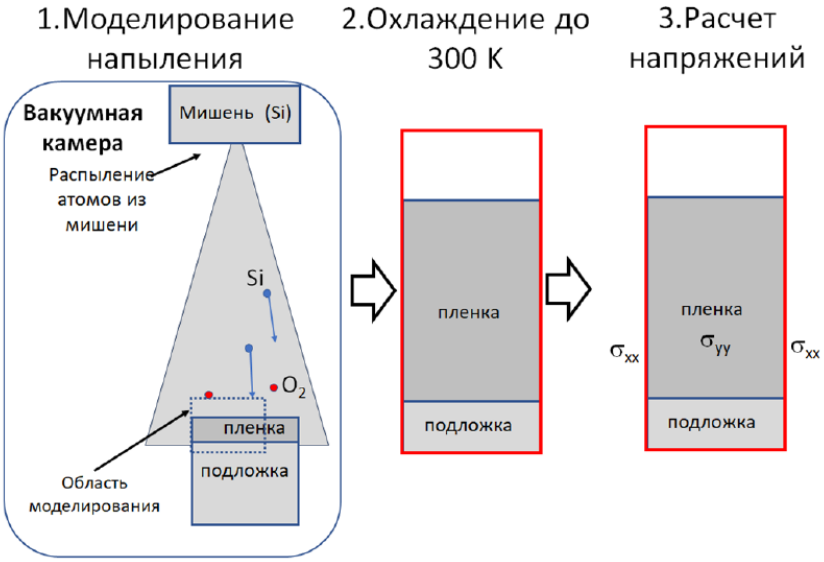


Рис. 1. Схема расчета напряжений по результатам моделирования напыления пленки методом молекулярной динамики.

1. Подготовка структуры подложки. Ее горизонтальные размеры определяются, исходя из имеющихся вычислительных ресурсов и масштабируемости рассчитываемых характеристик пленки. Например, плотность пленки можно рассчитать, используя относительно небольшие кластеры с характерным размером в несколько нанометров. В то же время для расчета пористости требуются кластеры с размерами в десятки нанометров. Толщина подложки обычно выбирается равной 4-6 нм. Это значение превышает глубину проникновения высокоэнергетических атомов и ассистирующих ионов (в случае напыления с ассистированием) в подложку и осаждаемую пленку.
2. Проводятся циклы напыления, как описано в [9]. В каждом цикле атомы кремния (титана) и кислорода вводятся в верхнюю часть области моделирования над подложкой. Начальные скорости атомов кремния (титана) и кислорода соответствуют энергии и температуре напыляемых частиц в вакуумной камере соответственно. Длительность одного цикла составляет 6 пс [9].
3. Напыление останавливается, когда толщина пленки достигает заданных значений.

МД моделирование выполняется в NVT ансамбле (постоянное число частиц, объем и температура). После каждого цикла осаждения вертикальный размер области моделирования увеличивается, чтобы учесть рост толщины пленки. Температура области моделирования контролируется с помощью термостата Берендсена [13]. Шаг по времени в уравнениях движения составляет 0,5 фс. Потенциальная энергия межатомного взаимодействия рассчитывается с использованием эмпирического силового поля DESIL со следующей функциональной формой:

$$U = q_i q_j / r_{ij} + A_{ij}/r_{ij}^{12} - B_{ij}/r_{ij}^{6}, \tag{1}$$

где q_i – заряд i-ого атома. Для атомов кислорода и кремния в диоксиде кремния $q_O = -0.65e$; $q_{Si} = 1.3e$. Для атомов кислорода в диоксиде титана $q_O = -1.098e$; $q_{Ti} = 2.196e$; e – величина элементарного заряда. Параметры потенциала Леннарда-Джонса (последние два слагаемых в Ур. (1)) приведены в Таблице 1.

Таблица 1. Параметры потенциала Леннарда-Джонса для SiO_2 [14] и TiO_2 [15], Ур. (1).

Взаим.	A_{ij} (кДж·нм^{12}/моль)	B_{ij} (кДж·нм^{6}/моль)
Si-Si	$1.5 \cdot 10^{-6}$	$5.0 \cdot 10^{-4}$
O-O	$1.5 \cdot 10^{-6}$	$5.0 \cdot 10^{-4}$
Si-O	$4.6 \cdot 10^{-8}$	$4.2 \cdot 10^{-3}$
Ti-Ti	$5.06 \cdot 10^{-4}$	$3.12 \cdot 10^{-8}$
O-O	$3.00 \cdot 10^{-3}$	$1.735 \cdot 10^{-6}$
Ti-O	$1.44 \cdot 10^{-3}$	$2.909 \cdot 10^{-7}$

Радиус обрезания неэлектростатической части потенциала равен 1 нм, как и в наших предыдущих работах [14, 11]. Электростатическая составляющая энергии взаимодействия рассчитывается без обрезания с использованием метода Particle Mesh Ewald (PME) [16].

Моделирование выполнено с использованием оборудования центра коллективного пользования для высокопроизводительных вычислений Московского государственного университета им. М. В. Ломоносова [17]. Молекулярно-динамическое моделирование выполняется с использованием программного пакета GROMACS [18], установленного на суперкомпьютере «Ломоносов-2». Визуализация напыленных атомистических структур выполнена с помощью программы Visual Molecular Dynamics (VMD) [19].

3. Результаты и обсуждение

3.1. Результаты моделирования

В настоящей работе моделируется высокоэнергетическое и низкоэнергетическое напыление оптических пленок из газовой фазы. При высокоэнергетическом напылении атомы кремния (титана) выбиваются из мишени (левая часть Рис. 1) ионами благородных элементов, ускоренными электрическим полем. В результате распыленные атомы Si(Ti) имеют начальную энергию порядка нескольких эВ [20]. Как правило, при моделировании высокоэнергетического напыления методом МД начальную энергию осаждаемых атомов выбирают равной 10 эВ [21, 22, 23]. Такая же величина используется и в настоящей работе. При низкоэнергетическом напылении (методы термического испарения, электронно-лучевого испарения) кинетическая энергия вылетающих из мишени частиц значительно ниже и не превышает нескольких десятых эВ [1]. В нашей работе начальная энергия атомов кремния (титана) при низкоэнергетическом напылении равна 0.1 эВ. Энергия атомов кислорода равна 0.1 эВ как при низкоэнергетическом напылении, так и при высокоэнергетическом.

Для температуры горячей подложки выбраны следующие значения: 400 K, 500 K и 600 K. Эти величины соответствуют экспериментальным условиям в [24, 25]. Моделирование напыления пленок диоксида кремния проводилось, как описано в предыдущем разделе. Толщина напыленного атомистического кластера достигла приблизительно 18 нм, на что потребовалось около 1500 циклов осаждения. Горизонтальные размеры подложки составляли 30 на 10 нм.

Результаты расчета напряжений представлены на Рис. 2, 3. При высокоэнергетическом напылении главные компоненты тензора напряжений имеют отрицательный знак (Рис. 2, левая сторона), что соответствует сжимающему напряжению (compressive stress), при котором напыленная пленка стремится увеличить свой объем, а подложка препятствует этому. Этот результат согласуется с экспериментальными данными по напряжениям в пленках, полученным высокоэнергетическим напылением [4, 26]. С ростом толщины пленки абсолютная величина компонент тензора напряжений растет. Такое поведение сходно с тем, что наблюдалось ранее при моделировании на холодной подложке [8] и на качественном уровне может объясняться следующим образом. Осаждаемые атомы, обладая большой – около 10 эВ – кинетической энергией, при столкновении с атомами пленки и подложки отталкивают их в стороны. Этот эффект сопровождается увеличением давления на границы области моделирования, что и выражается в росте напряжений. Тенденция к росту напряжений наблюдается при всех выбранных величинах температуры подложки.

Охлаждение подложки до комнатной температуры ведет к существенному уменьшению абсолютных величин напряжений приблизительно на 200-300 атм, причем эффект растет с увеличением температуры подложки в фазе нагрева. Этот результат может на качественном уровне быть объяснен тем, что снижение температуры приводит к снижению давления атомов пленки на границы области моделирования, что приводит к уменьшению абсолютных величин напряжений.

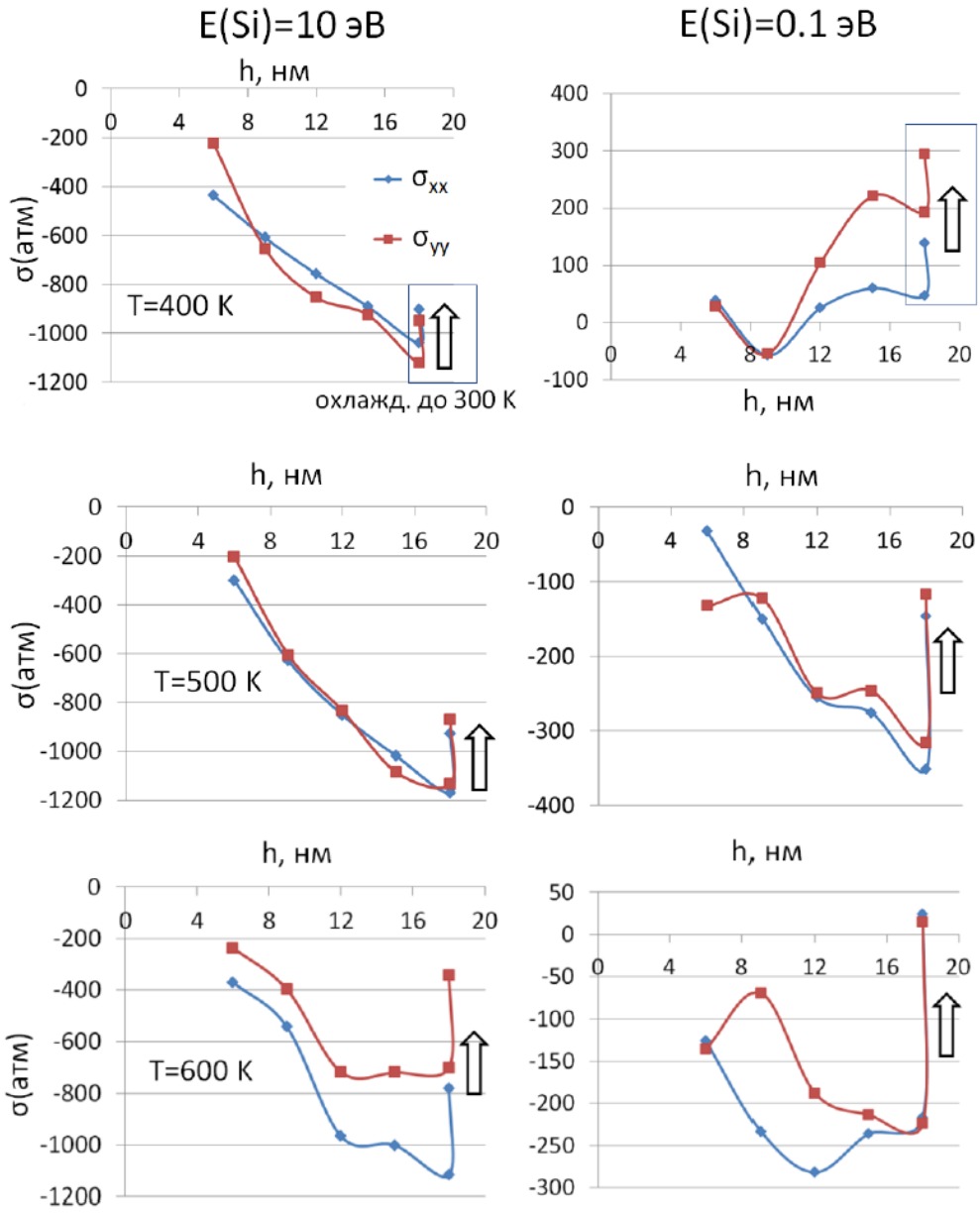


Рис. 2. Зависимость главных компонент тензора напряжений σ_{xx} и σ_{yy} в плоскости подложки от толщины пленки h при различных температурах T горячей подложки в пленках диоксида кремния. Стрелками показано изменение напряжений при охлаждении подложки до комнатной температуры после напыления (300 K).

При низкоэнергетическом напылении поведение напряжений с ростом толщины пленки, изменением температуры подложки и последующим охлаждением существенно отличается от случая высокоэнергетического напыления. Абсолютные значения главных компонент тензора напряжений существенно ниже. Этот результат согласуется с экспериментальными данными [27]. Отметим, что при $T = 400$ K напряжения в основном имеют положительный знак (tensile stress), то есть напыленная пленка стремится сжаться, чему препятствует подложка. При росте температуры подложки напряжения становятся отрицательными, как и при высокоэнергетическом напылении, однако их абсолютные величины в несколько раз меньше. Охлаждение подложки приводит к росту абсолютных величин напряжений в случае $T = 400$ K, и к умень-

шению при $T = 500$ K и 600 K. Эти различия объясняются тем, что при $T = 400$ K напряжения в ходе напыления имеют положительный знак, а при $T = 500$ K и 600 K – отрицательный. Как и при высокоэнергетическом напылении, рост температуры подложки сопровождается ростом абсолютной величины изменения напряжений при последующем охлаждении.

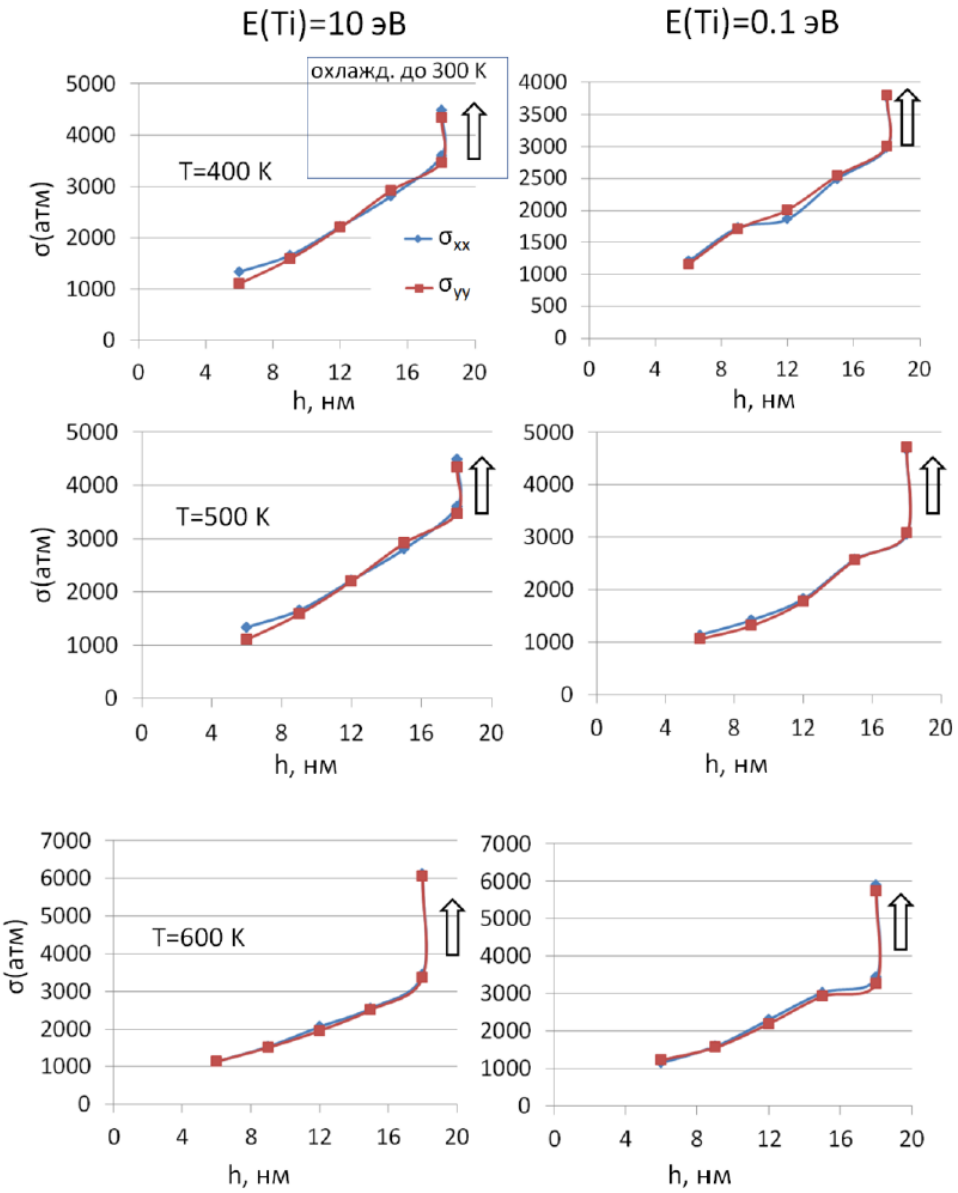


Рис. 3. Зависимость главных компонент тензора напряжений σ_{xx} и σ_{yy} от толщины пленки h при различных температурах T горячей подложки в пленках диоксида титана. Стрелками показано изменение напряжений при охлаждении подложки до комнатной температуры (300 K).

Расчет напряжений по описанной выше методике также был выполнен и для диоксида титана. Максимальная толщина напыленной пленки составляет около 18 нм, горизонтальные размеры подложки 14 нм на 14 нм. Напыление проводилось по той же схеме, как и в случае диоксида кремния. Результаты показаны на Рис. 3. В отличие от диоксида кремния, главные компоненты тензора напряжений имеют положительный знак (tensile stress) как при высокоэнергетическом, так и при низкоэнергетическом напылении. Такой результат соответствует экспериментальным

данным [28]. По абсолютной величине напряжения растут с увеличением толщины пленки. Вероятно, эффект «расталкивания» атомов пленки осаждаемыми атомами титана не сказывается в такой мере, как в диоксиде кремния. Охлаждение подложки после завершения напыления приводит к увеличению напряжений, которое на качественном уровне объясняется так же, как и в случае диоксида кремния. С ростом температуры подложки эффект от ее охлаждения несколько возрастает.

Таким образом, можно сделать следующие выводы. Охлаждение подложки после напыления на горячую подложку приводит к уменьшению абсолютных величин компонент тензора напряжений в случае, если напряжения имеют отрицательный знак. Напряжения, имеющие положительный знак, возрастают по величине при охлаждении. Количественно изменение напряжений при охлаждении зависит от пленкообразующего материала, температуры горячей подложки и толщины пленки. Возможно, разница в поведении напряжений в оксидах кремния и титана обусловлена различием коэффициентов температурного расширения.

3.2. Эффективность моделирования

Все расчеты проведены в разделе «pascal» суперкомпьютера «Ломоносов-2» со следующими основными характеристиками: общее число узлов 160, центральный процессор Intel Xeon Gold 6126 (12 ядер, 2.6 ГГц), графический ускоритель 2 x Nvidia P100 (3584 cuda ядер, 16 ГБ памяти), объем оперативной памяти на узле 96 ГБ.

Эффективность параллельных расчетов a в зависимости от числа ядер N_c определялась следующим образом:

$$a(N_c) = \frac{N_{at}(N_c)}{N_{at}(4)} \cdot \frac{4}{N_c},$$

где N_{at} – число напыленных атомов; эффективность при $N_c = 4$ принята равной 100%. Результаты показаны на Рис. 4. Видно, что с ростом числа ядер эффективность моделирования заметно уменьшается и снижается приблизительно до $1/3$ от исходного значения при числе ядер, равным 32.

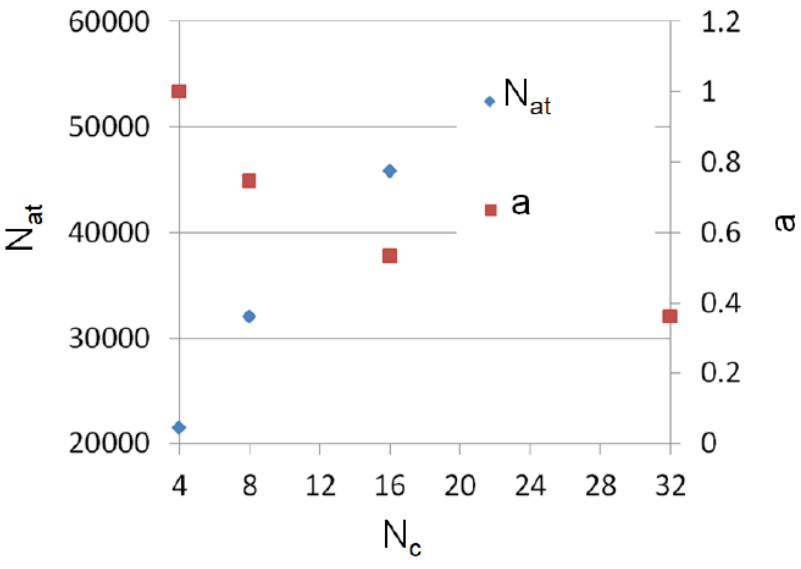


Рис. 4. Зависимость числа напыленных атомов N_{at} и эффективности параллельных расчетов a от числа ядер N_c. Время моделирования во всех случаях равно 500 минут.

Анализ снижения численной эффективности проведен средствами, имеющимися в рабочем кабинете пользователя суперкомпьютерного комплекса МГУ им. М.В. Ломоносова. Интенсивность передачи данных по MPI находится в зеленой зоне (существенных проблем нет). В то же время интенсивность чтения из файловой системы и интенсивность записи в файловую систему низкие (красная зона). Загрузка GPU находится либо в желтой, либо в красной зоне.

Таким образом, наиболее узкое место с точки зрения эффективности параллельных вычислений — запись и чтение с использованием файловой системы. Вероятно, наиболее простой путь увеличения эффективности заключается в сокращении объема сохраняемой в выходных файлах GROMACS информации, в частности, координат атомов и их скоростей. Такой подход может быть использован в задачах, в которых характеристики пленок могут быть рассчитаны по конечной геометрии или по ограниченному набору снимков траектории. Дальнейшее повышение эффективности потребует перестройки описанной выше схемы моделирования. В настоящий момент в каждом цикле напыления требуется запуск GROMACS, при котором программа считывает заново данные о предыдущей конфигурации кластера, а также проводит подготовительные процедуры для начала МД моделирования. В условиях, когда длительность цикла напыления мала – всего 6 пс, – а размер файлов с геометрией атомистического моделирования велик, относительный вклад этих процедур в общее время моделирования возрастает с увеличением числа ядер, используемых в параллельных расчетах. Именно с этим связано снижение эффективности (Рис. 4). Для решения проблемы необходимо собрать исполняемый модуль GROMACS, в котором предусмотрено моделирование напыления – например, так, как это сделано в программе Lammps [29].

Заключение

В настоящей работе проведено суперкомпьютерное моделирование напыления тонких пленок диоксида кремния и диоксида титана на горячей подложке. Рассчитаны напряжения при различных толщинах пленки, а также изменение напряжений при охлаждении подложки после завершения напыления.

В пленках диоксида кремния напряжения в основном сжимающие (compressive stress), а их абсолютная величина существенно зависит от энергии напыления. При высокоэнергетическом напылении охлаждение подложки приводит к уменьшению напряжений по абсолютной величине, при низкоэнергетическом напылении абсолютные величины главных компонент тензора напряжений могут как увеличиваться, так и уменьшаться. В диоксиде титана как при высокоэнергетическом, так и при низкоэнергетическом напылении наблюдаются растягивающие напряжения (tensile stress). При охлаждении подложки их абсолютная величина растет.

Проведен анализ численной эффективности суперкомпьютерных расчетов с учетом больших размеров области атомистического моделирования. Найдено, что запись и чтение в/из файловой системы в наибольшей степени снижает эффективность расчетов. Предложены способы решения проблемы.

Статья подготовлена при финансовой поддержке Министерства науки и высшего образования в рамках программы Московского центра фундаментальной и прикладной математики по Соглашению № 075-15-2025-345.

Список литературы

[1] Optical Thin Films and Coatings; Angela Piegari, F.F., Ed.; Elsevier, 2018.

[2] Qiao, G.; Hu, H.; Zhang, X.; Luo, X.; Xue, D.; Zhang, G.; Hu, H.; Yi, L.; Yang, Y.; Deng, W. Stress-Induced Deformation of the Coating on Large Lightweight Freeform Optics. Opt. Express 2021, 29, 4755–4769, doi:10.1364/OE.414953.

[3] Forschelen, P.J.J.; Suiker, A.S.J.; van der Sluis, O. Effect of Residual Stress on the Delamination Response of Film-Substrate Systems under Bending. Int. J. Solids Struct. 2016, 97–98, 284–299, doi:https://doi.org/10.1016/j.ijsolstr.2016.07.020.

[4] Abadias, G.; Chason, E.; Keckes, J.; Sebastiani, M.; Thompson, G.B.; Barthel, E.; Doll, G.L.; Murray, C.E.; Stoessel, C.H.; Martinu, L. Review Article: Stress in Thin Films and Coatings: Current Status, Challenges, and Prospects. J. Vac. Sci. Technol. A 2018, 36, 20801, doi:10.1116/1.5011790.

[5] Zhang, M.; Rao, Z.; Kim, K.-S.; Qi, Y.; Fang, L.; Sun, K.; Chason, E. Molecular Dynamics Simulation of Stress Induced by Energetic Particle Bombardment in Mo Thin Films. Materialia 2021, 16, 101043, doi:https://doi.org/10.1016/j.mtla.2021.101043.

[6] Xiaoqiang, W.; Xu, Z.; Xiangyi, H.; Yingjian, T.; Haojie, W.; Haoran, F.; Huimin, L. Molecular Dynamics Simulation of Hybrid Structure and Mechanical Properties of DLC/Ni-DLC Thin Films. Sci. Rep. 2024, 14, 18885, doi:10.1038/s41598-024-69759-9.

[7] Ghorbani, H.; Poladi, A. MD-Simulation of Duty Cycle and TaN Interlayer Effects on the Surface Properties of Ta Coatings Deposited by Pulsed-DC Plasma Assisted Chemical Vapor Deposition. Int. J. Eng. 2020, 33, 861–869, doi:10.5829/ije.2020.33.05b.18.

[8] Grigoriev, F. V.; Sulimov, V.B.; Tikhonravov, A. V. Atomistic Simulation of Stresses in Growing Silicon Dioxide Films. Coatings 2020, 10, 220, doi:10.3390/coatings10030220.

[9] Grigoriev, F. V; Sulimov, A. V; Kochikov, I. V; Kondakova, O.A.; Sulimov, V.B.; Tikhonravov, A. V Supercomputer Modeling of the Ion Beam Sputtering Process: Full-Atomistic Level. In Proceedings of the Proceedings of SPIE - The International Society for Optical Engineering; Bellingham, WA, United States, 2015; Vol. 9627, pp. 962701–962708.

[10] Li, X.; Ke, P.; Lee, K.-R.; Wang, A. Molecular Dynamics Simulation for the Influence of Incident Angles of Energetic Carbon Atoms on the Structure and Properties of Diamond-like Carbon Films. Thin Solid Films 2014, 552, 136–140, doi:https://doi.org/10.1016/j.tsf.2013.12.012.

[11] Grigoriev, F. V; Sulimov, V.B.; Tikhonravov, A. V Study of Thin Optical Films Stress Dependence on Deposition Conditions by Large-Scale Molecular Dynamics. Coatings 2022, 12, 750, doi:10.3390/coatings12060750.

[12] Baptista, A.; Silva, F.; Porteiro, J.; Miguez, J.; Pinto, G. Sputtering Physical Vapour Deposition (PVD) Coatings: A Critical Review on Process Improvement and Market Trend Demands. Coatings 2018, 8.

[13] Berendsen, H.J.C.; Postma, J.P.M.; Van Gunsteren, W.F.; Dinola, A.; Haak, J.R. Molecular Dynamics with Coupling to an External Bath. J. Chem. Phys. 1984, 81, 3684–3690, doi:10.1063/1.448118.

[14] Grigoriev, F. V.; Sulimov, A. V.; Kochikov, I. V.; Kondakova, O.A.; Sulimov, V.B.; Tikhonravov, A. V. High-Performance Atomistic Modeling of Optical Thin Films Deposited by Energetic Processes. Int. J. High Perform. Comput. Appl. 2015, 29, 184–192, doi:10.1177/1094342014560591.

[15] Grigoriev, F. V.; Sulimov, V.B.; Tikhonravov, A. V. Application of a Large-Scale Molecular Dynamics Approach to Modelling the Deposition of TiO2 Thin Films. Comput. Mater. Sci. 2021, 188, 110202, doi:10.1016/j.commatsci.2020.110202.

[16] Darden, T.; York, D.; Pedersen, L. Particle Mesh Ewald: An N-log(N) Method for Ewald Sums in Large Systems. J. Chem. Phys. 1993, 98, 10089–10092, doi:10.1063/1.464397.

[17] Voevodin, V. V.; Antonov, A.S.; Nikitenko, D.A.; Shvets, P.A.; Sobolev, S.I.; Sidorov, I.Y.; Stefanov, K.S.; Voevodin, V. V.; Zhumatiy, S.A. Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. Supercomput. Front. Innov. Vol 6, No 2 2019.

[18] Abraham, M.J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J.C.; Hess, B.; Lindahl, E. GROMACS: High Performance Molecular Simulations through Multi-Level Parallelism from Laptops to Supercomputers. SoftwareX 2015, 1-2, 19–25, doi:10.1016/j.softx.2015.06.001.

[19] Humphrey, W.; Dalke, A.; Schulten, K. VMD: Visual Molecular Dynamics. J. Mol. Graph. 1996, 14, 33–38, doi:https://doi.org/10.1016/0263-7855(96)00018-5.

[20] WINTERS, H.F. Physical Sputtering: A Discussion of Experiment and Theory. In Radiation Effects on Solid Surfaces; Advances in Chemistry; AMERICAN CHEMICAL SOCIETY, 1976; Vol. 158, p. 1 ISBN 9780841203310.

[21] Badorcek, H.; Steinecke, M.; Jensen, L.; Ristau, D.; Jupe, M.; Müller, J.; Tonneau, R.; Moskovkin, P.; Lucas, S.; Pflug, A.; et al. Correlation of Structural and Optical Properties Using Virtual Materials Analysis. Opt. Express 2019, 27, 22209–22225, doi:10.1364/OE.27.022209.

[22] Taguchi, M.; Hamaguchi, S. MD Simulations of Amorphous SiO2 Thin Film Formation in Reactive Sputtering Deposition Processes. Thin Solid Films 2007, 515, 4879–4882, doi:https://doi.org/10.1016/j.tsf.2006.10.097.

[23] Grigoriev, F. V.; Sulimov, A. V.; Katkova, E. V.; Kochikov, I. V.; Kondakova, O.A.; Sulimov, V.B.; Tikhonravov, A. V. Full-Atomistic Nanoscale Modeling of the Ion Beam Sputtering Deposition of SiO2 Thin Films. J. Non. Cryst. Solids 2016, 448, 1–8, doi:10.1016/j.jnoncrysol.2016.06.015.

[24] Amirzada, M.R.; Tatzel, A.; Viereck, V.; Hillmer, H. Surface Roughness Analysis of SiO2 for PECVD, PVD and IBD on Different Substrates. Appl. Nanosci. 2016, 6, 215–222, doi:10.1007/s13204-015-0432-8.

[25] Juárez, H.; Pacio, M.; Díaz, T.; Rosendo, E.; Garcia, G.; García, A.; Mora, F.; Escalante, G. Low Temperature Deposition: Properties of SiO2 Films from TEOS and Ozone by APCVD System. J. Phys. Conf. Ser. 2009, 167, 12020, doi:10.1088/1742-6596/167/1/012020.

[26] Bischoff, M.; Nowitzki, T.; Voß, O.; Wilbrandt, S.; Stenzel, O. Postdeposition Treatment of IBS Coatings for UV Applications with Optimized Thin-Film Stress Properties. Appl. Opt 2014, 53, A212–A220.

[27] Tolenis, T.; Grinevičiūtė, L.; Smalakys, L.; Ščiuka, M.; Drazdys, R.; Mažulė, L.; Buzelis, R.; Melinkaitis, A. Next Generation Highly Resistant Mirrors Featuring All-Silica Layers. Sci. Rep. 2017, 7, 10898, doi:10.1038/s41598-017-11275-0.

[28] Chen, T.; Luo, C.; Wang, D.; Xiong, Y. Effect of Ion Beam Bombarding on Stress in TiO2 Thin Films. Phys. Procedia 2011, 18, 136–142, doi:https://doi.org/10.1016/j.phpro.2011.06.071.

[29] Aktulga, H.M.; Fogarty, J.C.; Pandit, S.A.; Grama, A.Y. Parallel Reactive Molecular Dynamics: Numerical Methods and Algorithmic Techniques. Parallel Comput. 2012, 38, 245–259, doi:https://doi.org/10.1016/j.parco.2011.08.005.

Управление рабочими процессами в распределенной среде с использованием HTCondor*

М.Л. Воскобойников[1], А.Г. Феоктистов[1], И.В. Бычков[1]

[1]Институт динамики систем и теории управления им. В.М. Матросова СО РАН, 664033, Иркутск, ул. Лермонтова, 134, а/я 29, Россия

При решении ресурсоемких задач на распределенных и облачных ресурсах, как правило, используются системы управления прохождением заданий. Одним из популярных средств подобного назначения является система HTCondor. В ней схема решения задачи, включающая набор взаимосвязанных подзадач, представляется рабочим процессом в виде ориентированного ациклического графа. К сожалению, в подобных системах информация о времени выполнения модулей рабочего процесса, реализующих его операции, обслуживания заданий в очереди, подготовки и запуска контейнеров, а также сведения о размерах передаваемых данных между ними и пропускной способности сети зачастую не учитывается при назначении ресурсов. В работе предложены алгоритмы планирования вычислений и назначения ресурсов на основе тестирования модулей на специальных испытательных стендах, а также механизмы передачи данных между модулями, позволяющие уплотнить расписание выполнения рабочего процесса и ускорить вычисления в сравнении с соответствующими алгоритмами и механизмами HTCondor.

Ключевые слова: рабочий процесс, тестирование, передача данных, управление вычислениями, HTCondor

1. Введение

Сегодня развитие средств и методов организации высокопроизводительных вычислений позволяет разрабатывать и применять все более сложные приложения, основанные на использовании рабочих процессов (РП) [1]. Как правило, сложность управления такими приложениями в гетерогенной распределенной вычислительной среде (ГРВС) обуславливается следующими факторами: большим числом компонентов приложений и связей между ними; востребованностью параллельных и распределенных вычислений; возможной информационной и программно-аппаратной избыточностью при планировании вычислений, а также вероятным ограничением параллелизма при назначении ресурсов; потенциальной необходимостью динамического выделения ресурсов, развертывания на них программного обеспечения (ПО) и управления им при выполнении РП; требованием эффективного использования ресурсов.

Для разработки и применения приложений, основанных на РП, зачастую используются системы управления рабочими процессами [1]. Как правило, управление вычислениями в ГРВС осуществляется на двух уровнях. На уровне среды управление осуществляется метапланировщиками, такими как GridWay [2] и Condor DAGMan [3] или средствами систем управления рабочими процессами. На уровне отдельных ресурсов среды (например, на вычислительных кластерах) управление поддерживается

*Исследование выполнено при поддержке Министерства науки и высшего образования Российской Федерации, проект № FWEW-2026-0012 «Методы и технологии облачной сервис-ориентированной цифровой плат-формы сбора, хранения и обработки больших объёмов разноформатных междисциплинарных данных и знаний, основанные на применении искусственного интеллекта, компонентного и модельно-управляе-мого подходов и машинного обучения» (рег. № 126021217141-8).

системами управления прохождением заданий (СУПЗ) Slurm [4], TORQUE/Maui [5], HTCondor [6] и др. Кроме того, дополнительное управление может осуществляться на уровне приложений путем выбора конкретных ресурсов и узлов среды.

HTCondor является одной из известных и широко используемых на практике СУПЗ [7, 8, 9]. Преимущества HTCondor по сравнению с другими СУПЗ, такими как Slurm и TORQUE/Maui, обусловлены рядом важных факторов. В их числе наличие гибкого и расширяемого языка описания заданий, поддержка контейнеризации и возможность определения сложных зависимостей заданий, которые могут быть представлены в виде ориентированного ациклического графа (англ., Directed Acyclic Graph – DAG).

Однако при его использовании возникает ряд проблем, связанных с эффективностью назначения ресурсов и передачи данных [10]. При помещении задания в очередь HTCondor выполняет поиск наиболее подходящего для него ресурса в соответствии с заданными требованиями к ресурсу в описании задания. При этом HTCondor пытается одновременно запустить как можно больше готовых к выполнению заданий на всех доступных ресурсах в соответствии с порядком их выполнения, описанном в файле DAG. Передача данных между программными модулями заданий осуществляется через управляющий узел, что приводит к большому числу пересылок данных. Как правило, в известных работах данная проблема решается либо увеличением пропускной способности каналов связи между управляющим узлом и вычислительными узлами [7, 8], либо кэшированием данных на вычислительных узлах [9].

В рамках данного исследования предложен новый подход к управлению РП в инструментальном комплексе Framework for Development and Execution of Scientific WorkFlows (FDE-SWFs) [10] с использованием системы HTCondor. В частности, разработаны алгоритмы управления вычислениями и реализован механизм прямой передачи данных между модулями, что в совокупности обеспечивает существенное ускорение выполнения РП относительно времени вычислений с использованием базовых алгоритмов и механизмов HTCondor.

2. Алгоритмы

В работе применяется подход к управлению вычислениями на уровне приложений [11]. В отличие от подходов к управлению на уровнях ГРВС и СУПЗ, суть данного подхода заключается в рациональном выборе ресурсов, наиболее подходящих для эффективного выполнения конкретного приложения. Выбор ресурсов выполняется на основе оценок времени выполнения компонентов РП. Организация вычислительного процесса в FDE-SWFs характеризуется следующими особенностями:

- схема решения задачи (частично-упорядоченная последовательность абстрактных операций, реализуемых программными модулями) представляется в виде РП, выполняемого в динамически развертываемой ГРВС на выделенных ресурсах;
- требования по выполнению операций РП описываются в заданиях метапланировщику и СУПЗ, используемых на ресурсах среды;
- метапланировщик Condor DAGMan распределяет задания между ресурсами;
- СУПЗ HTCondor управляет заданиями на ресурсах;
- ПО внешних библиотек задействуется при выполнении РП;
- динамическое развертывание и выполнение прикладной и системной составляющих СОП на выделенных ресурсах требует автоматизации методов и средств интеграции, тестирования и контейнеризации ПО;
- контейнеры используются для создания изолированной, воспроизводимой и портативной среды для выполнения приложений, включающей все необходимое системное и прикладное ПО;

- в качестве средства управления контейнерами используется Docker [12];
- автоматизация развертывания и конфигурирования ГРВС на выделенных ресурсах производится с помощью системы Ansible [13];
- РП включает фрагменты вычислений в асинхронном и синхронном режимах: базовые операции РП выполняются в асинхронном режиме, а операции обработки параллельных списков (вариантов) данных осуществляются в синхронном режиме;
- не предполагается взаимодействия по данным и управлению между экземплярами операций, обрабатывающих разные варианты данных;
- разрабатываемые приложения ориентированы на классы задач, в которых среднее время обработки любого варианта данных имеет незначительное отклонение от среднего времени обработки всех вариантов данных.

Схема работы диспетчера РП с метапланировщиком Condor DAGMan и СУПЗ HTCondor представлена на рис. 1. Алгоритм планирования вычислений (построения частично-упорядоченной последовательности выполнения операций) по непроцедурной постановке задачи реализует последовательное применение методов прямой и обратной волн [14] с целью построения РП, который в общем случае включает все множество сценариев решения задачи. Для выбора единственного сценария применяется алгоритм конкретизации РП путем удаления операций, выполняющих избыточные вычисления. Выбранный сценарий решения задачи представляет собой ярусно-параллельную форму графа алгоритма. Алгоритмы генерации заданий формируют описания выполнения РП в целом и выполнения каждого его отдельного компонента с использованием форматов Condor DAGMan и HTCondor. Диспетчер РП осуществляет контейнеризацию прикладного и системного ПО, необходимого для выполнения РП, и развертывает ГРВС. РП запускаются и выполняются в вычислительной среде средствами Condor DAGMan и HTCondor. Диспетчер РП мониторит процесс их выполнения. По завершении выполнения РП результаты расчетов сохраняются в расчетной базе данных (БД) и при необходимости визуализируются.

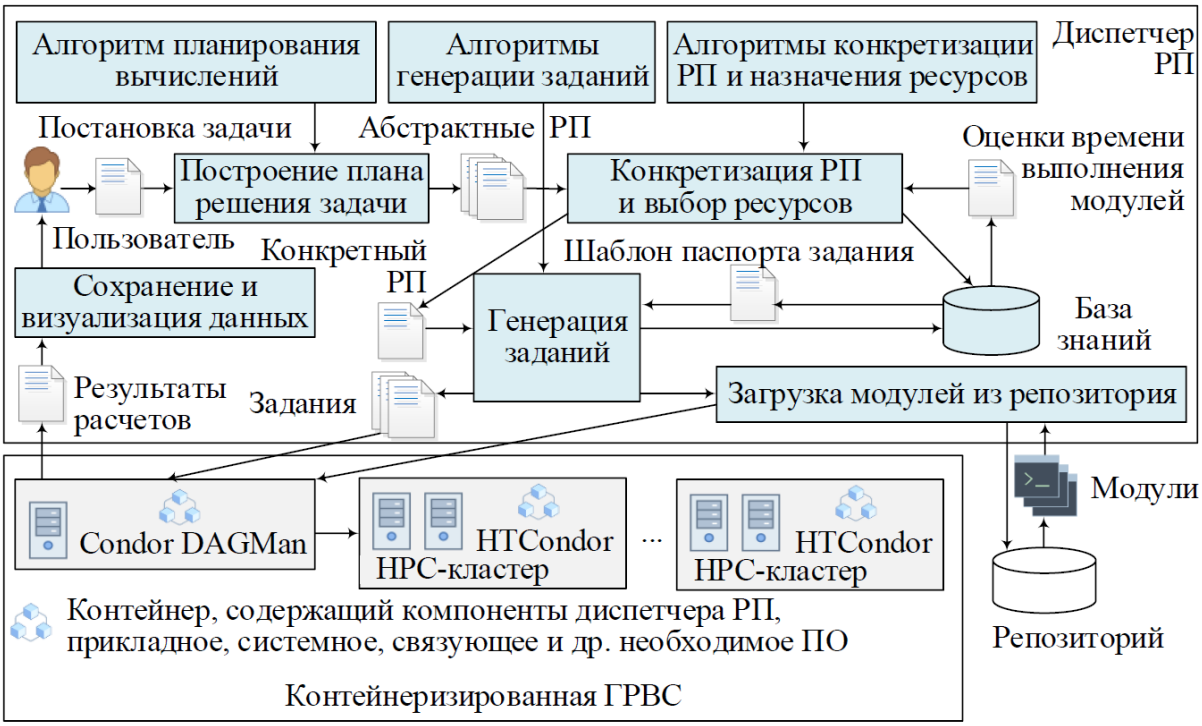


Рис. 1. Взаимодействие диспетчера с Condor DAGMan и HTCondor.

3. Алгоритм назначения ресурсов

Назначение операций на узлы реализуется с помощью специально разработанного эвристического алгоритма. Как правило, РП имеют сложную структуру, состоящую из большого числа операций, экземпляры которых могут выполняться последовательно и параллельно. В СУПЗ HTCondor передача данных между вычислительными узлами производится через управляющий узел, что приводит в общем случае к возрастанию числа пересылок данных по сети почти в два раза в сравнении с передачей данных непосредственно между операциями. Это обуславливает избыточный объем пересылаемых данных, который существенно увеличивает время выполнения РП и вычислительного эксперимента в целом. Для сокращения времени выполнения РП предлагается пересылать данные между вычислительными узлами напрямую, минуя управляющий узел. При запуске РП на узлах создаются временные папки, в которые перемещаются данные и модули. Задание выполняется от имени пользователя nobody. Данный пользователь не имеет разрешений на выполнение модулей вне созданной временной папки и команд на перемещение данных между вычислительными узлами. Организация прямого перемещения данных между вычислительными узлами и выполнения на них модулей включает следующие этапы:

- задание в конфигурационном файле HTCondor пользователя с разрешением на чтение/запись в папки вне временной папки, создаваемой в ОС при выполнении задания;
- планирование очередности выполнения модулей для обеспечения минимизации времени выполнения РП с учетом перемещения данных между операциями;
- генерация скриптов-оболочек, в которых прописываются команды для выполнения модулей и передачи данных между вычислительными узлами в соответствии с РП;
- формирование заданий, в которых в качестве исполняемых модулей указываются скрипты-оболочки и адреса вычислительных узлов.

На рис. 2 показана схема работы механизма передачи данных через управляющий узел. Можно заметить, что при запуске и завершении выполнения каждого прикладного модуля РП происходит передача входных и выходных данных между вычислительными узлами и управляющим узлом. На рис. 2 изображены схемы работы механизма перемещения данных через управляющий узел (рис. 2 а) и передачи данных между узлами (рис. 2 б).

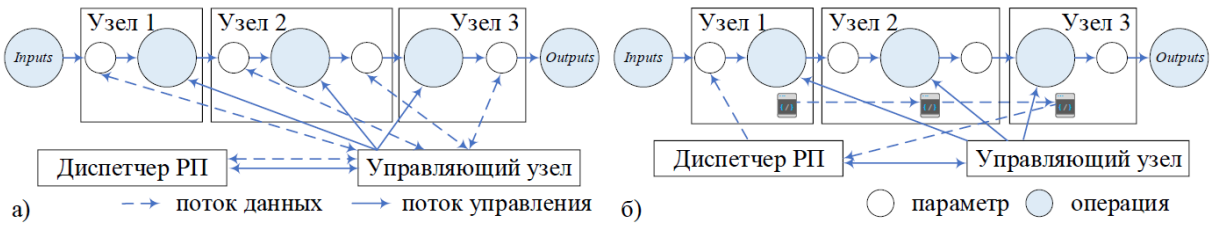


Рис. 2. Передача данных: через управляющий узел (а) и прямая (б)

Пусть имеются m узлов ГРВС (включая управляющий узел), на которых должны быть выполнены n операций РП, где b – индекс управляющего узла. Введем следующие обозначения:

- O – множество операций РП;
- X – булева матрица размерности $m \times n$ (элемент $x_{li} = 1$ показывает, что i-я операция выполняется на l-м узле);

- τ_{ir} (τ_{jl}) – оценка времени, прошедшего с начала выполнения РП до завершения операции o_i (o_j) на r-м (l-м) узле;
- q_{il} – оценка времени нахождения модуля, реализующего операцию o_i в очереди l-го узла;
- c_{il} – оценка временных затрат на контейнеризацию ПО на l-м узле для выполнения i-й операции;
- t_{il} – время выполнения i-й операции на l-м узле с учетом чтения и записи данных;
- P – булева матрица предшествования операций размерности $n \times n$ (элемент матрицы $p_{ij} = 1$ означает, что j-я операция предшествует i-й операции);
- D – матрица оценки размеров передаваемых данных размерности $n \times n$ (элемент матрицы $d_{ij} > 0$ показывает размер данных, передаваемых между i-й и j-й операциями);
- W – матрица пропускной способности интерконнекта размерности $m \times m$ (элемент матрицы $w_{lr} \geq 0$ демонстрирует пропускную способность интерконнекта между l-м и r-м узлами);
- $\omega_{lr}(t)$ – коэффициент снижения пропускной способности в момент времени t между l-м и r-м узлами;
- $m \leq \delta$ – число выделенных узлов;
- δ – квота на число узлов;
- n – число операций;
- b – индекс управляющего узла;
- λ_l – квота на время использования l-го узла, $\tau_{il} \leq \lambda_l$.

Матрица X должна удовлетворять следующим условиям:

$$\bigvee_{i=1}^{n} \bigvee_{l=1}^{m-1} \bigvee_{k=l+1}^{m} (x_{li} \wedge x_{ki}) = 0, \quad \bigvee_{i=1}^{n} \bigwedge_{l=1}^{m} \bar{x}_{li} = 0. \tag{1}$$

В случае выполнения РП средствами СУПЗ HTCondor, где выбор и назначение узлов для запуска операций осуществляется главным менеджером этой системы, а обмен данными между узлами осуществляется через управляющий узел, оценка $T(X)$ времени выполнения РП в асинхронном режиме по готовности данных определяется по формулам:

$$T(X) = \max_{i=\overline{1,n},\ l\in\overline{1,m}} \tau_{il}, \tag{2}$$

$$\tau_{il} = q_{il} + c_{il} + t_{il} + \max_{\forall j\in\overline{1,n}: p_{ij}=1, i\neq j, r\in\overline{1,m}} \left(\tau_{jr} + \frac{d_{ij}}{\omega_{rb}(t) w_{rb}} + \frac{d_{ij}}{\omega_{bl}(t) w_{bl}} \right). \tag{3}$$

В диспетчере РП реализован дополнительный механизм передачи данных напрямую с узла выполнения предшествующей операции на узел выполнения последующей операции. В этом случае, если операции выполняются на одном узле, то размер передаваемых данных между ними считается равным нулю. Тогда оценка $Y(X)$ времени выполнения РП в асинхронном режиме по готовности данных с помощью диспетчера РП определяется следующим образом:

$$Y(X) = \max_{i=\overline{1,n},\ l\in\overline{1,m}} \tau_{il}, \tag{4}$$

$$\tau_{il} = q_{il} + c_{il} + t_{il} + \max_{\forall j\in\overline{1,n}: p_{ij}=1, i\neq j, r\in\overline{1,m}} \left(\tau_{jr} + \frac{s_{lr}}{\omega_{lr}(t) w_{lr}} \right), \tag{5}$$

$$s_{lr} = \begin{cases} d_{ij}, & \text{если } l \neq r, \\ 0 & \text{в противном случае.} \end{cases} \tag{6}$$

В (4)–(6) $Y(X)$ является целевой функцией, которую требуется минимизировать. В общем случае эта задача является NP-трудной задачей, частным случаем которой является построение m-процессорного расписания с условиями предшествования [15], оптимальное решение которой сложно получить за время, приемлемое для управления вычислениями в реальном времени. Поэтому в работе предлагается эвристический алгоритм решения этой задачи.

Суть алгоритма назначения ресурсов состоит в следующем: каждому узлу назначаются такие операции из очереди операций, чтобы общее время выполнения РП было минимальным. Для этого выполняется оценка времени выполнения операции на каждом узле с учетом оценок размера передаваемых данных между операциями и скоростью передачи данных между узлами ГРВС. При этом, если на очередном шаге работы алгоритма одному и тому же узлу назначаются две и более операции, то выбирается та операция, которая предшествует большему числу операций РП. Учет структуры РП в алгоритме позволяет уплотнить расписание выполнения заданий в сравнении с алгоритмом HTCondor и тем самым сократить время вычислений.

При реализации в РП многовариантных расчетов с помощью экземпляров операции обработки параллельного списка данных выполнение РП разделяется на фрагменты вычислений в асинхронном и синхронном режимах. К моменту синхронных вычислений все асинхронные расчеты должны быть завершены, и все узлы свободны. Реализация операции включает дезагрегирование данных на k элементов параллельного списка, синхронную обработку элементов экземплярами операции и последующее агрегирование результатов расчетов. При назначении узлов в синхронном режиме решается задача распределения нагрузки в узлах и ее балансировки:

$$\max_{l=\overline{1,m}} \frac{v_l}{\pi_l} \downarrow \min, \tag{7}$$

$$\sum_{l=1}^{m} v_l = k, \quad 0 \leq \frac{v_l}{\pi_l} \leq \lambda_l, \tag{8}$$

где v_l – переменная, представляющая искомое число экземпляров операции для l-го узла, k – общее число экземпляров, π_l – производительность l-го узла.

Ниже приведен алгоритм назначения ресурсов. В алгоритме функция `GetReady Operations` возвращает список операций $R \subseteq O$, готовых к выполнению. Функция `NextOperation` возвращает номер очередной назначаемой операции. Функция `GetOperationType` возвращает тип операции. Функция `BalanceOperationsOnNodes` предназначена для распределения нагрузки в узлах и ее балансировки при многовариантных расчетах, T – вектор размерности m времени освобождения узлов. Функция `ShiftNodeTimeToMin` продвигает время на узлах после назначения операций. Функция `GetSubsequent`
`Operations` возвращает число последующих операций для каждой операции из O, S – вектор размерности n, содержащий число последующих операций для каждой из n операций. Функция `EstimateOperationsTime` производит расчет времени выполнения каждой операции из R для каждого свободного узла. Функция `AssignOperationsOnNodes` возвращает результат назначения операций на узлы, F – вектор размерности m состояния узлов, где $F_l = 0$ ($F_l = 1$) означает, что узел

свободен (занят), C – вектор размерности m, представляющий число операций, назначенных на каждый узел. Функция `GetSuitableOperation` возвращает номер операции, предшествующей большему числу операций РП. Функция `RemoveOperation` удаляет операцию из списка операций. Функция `ReleaseNodes(F, T)` освобождает ресурсы, на которых выполнились операции. Вычислительная сложность алгоритма в худшем случае составляет $O(m^2 \times n^2)$. Поэтому он вполне применим на практике.

Алгоритм 1 Алгоритм назначения ресурсов.

Require: $inputs_1, outputs_1, inputs_2, outputs_2, \ldots, inputs_n, outputs_n$, t, P, D, O, n, m, q, W, X, ω, c
Ensure: X

$R = \text{GetReadyOperations}(O, X, inputs, outputs)$
$i = \text{NextOperation}(R)$
if $\text{GetOperationType}(i) ==' parallel'$ **then**
    $T = \text{BalanceOperationsOnNodes}(i, t, P, T, D, W, X, q, c, \omega, n, m)$
    $F = \text{ReleaseNodes}(F, T)$
    $R = \text{RemoveOperation}(R, i)$
end if
$S = \text{GetSubsequentOperations}(O)$
while $length(R) > 0$ **do**
    $R = \text{GetReadyOperations}(O, X, inputs, outputs)$
    $T = \text{EstimateOperationsTime}(R, t, P, D, W, q, c, \omega, n, m)$
    $C = \text{AssignOperationsOnNodes}(R, T, F)$
    for $l = 1$ To m **do**
        if $length(C[l]) == 1$ **then**
            $i = C[l]$
        else if $length(C[l]) > 1$ **then**
            $i = \text{GetSuitableOperation}(C[l], S)$
        end if
        $T[l] = \text{EstimateNodeTime}(i, t, P, D, W, q, c, \omega, n, m)$
        $X[l, i] = 1$
        $R = \text{RemoveOperation}(R, i)$
        $F[l] = 1$
    end for
    $T = \text{ShiftNodeTimeToMin}(T)$
    $F = \text{ReleaseNodes}(F, T)$
    $R = \text{GetReadyOperations}(O, X, inputs, outputs)$
    $i = \text{NextOperation}(R)$
    if $\text{GetOperationType}(i) ==' parallel'$ **then**
        $T = \text{BalanceOperationsOnNodes}(i, t, P, T, D, W, X, q, c, \omega, n, m)$
        $F = \text{ReleaseNodes}(F, T)$
        $R = \text{RemoveOperation}(R, i)$
    end if
end while

4. Вычислительный эксперимент

Разработано приложение для решения важной научной задачи исследования эффективности методов структурно-параметрической оптимизации инфраструктуры энергетической системы (ЭС) относительно ее живучести. Создан стенд для тестирования, оценки и последующего выбора наиболее подходящих методов из 60 методов известной библиотеки PaGMO относительно модели определенного уровня с использованием оператора неявного многометодного анализа. Стенд успешно применен при выборе методов для исследования живучести локальной ЭС населенного пункта, расположенного на Байкальской природной территории (БПТ) [16].

На рис. 3 представлен рабочий прикладной рабочий процесс. Операция o_1 реализует структурно-параметрическую оптимизацию ЭС с помощью наиболее подходящего метода оптимизации из библиотеки PaGMO, определяемого с помощью испытательного стенда. Далее операция o_2 обрабатывает результаты оптимизации. Наконец, операция o_3 создает отчет о результирующей конфигурации локальной ЭС. Операции используют следующие параметры: z_1 – файл, содержащий настройки метода оптимизации; z_2 – входная БД, содержащая исходные данные для модели ЭС; z_3 – исходная БД для хранения результатов оптимизации локальной ЭС; z_4 – SQL-скрипт для обработки результатов оптимизации; z_5 – SQL-скрипт для создания отчетов; z_6 – результат выполнения операции o_2; z_7 – результат выполнения операции o_3; z_8 – отчет о предпочтительных типах оборудования, их составе, характеристиках и местах расположения; z_9 – БД с результатами оптимизации; z_{10} – БД с обработанными и агрегированными результатами оптимизации для создания z_8.

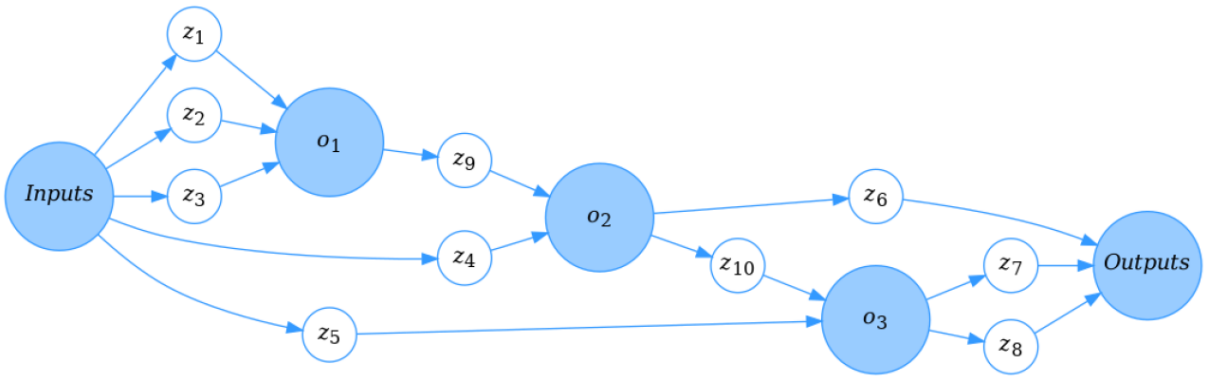


Рис. 3. Прикладной РП.

Испытательный стенд разработан в виде системного РП (рис. 4). Он включает следующие операции: o_1 – операция для генерации списка файлов с именами методов; o_2 – системная операция для дезагрегирования списка файлов методов по отдельным директориям; o_3 – системная операция для дезагрегирования списка экземпляров модулей по отдельным директориям; o_4 – системная операция для изменения имени метода в файле конфигурации; o_5 – системная операция для изменения числа поколений в файле конфигурации; o_6 – системная операция для изменения размера популяции в файле конфигурации; o_7 – операция структурно-параметрической оптимизации; o_8 – системная операция для агрегирования результатов структурно-параметрической оптимизации; o_9 – системная операция многокритериального выбора метода с помощью Парето-оптимального правила; o_{10} – системная операция назначения приоритетов критериев; o_{11} – системная операция многокритериального выбора метода с помощью лексикографического правила. Операции используют

следующие параметры: z_1 – шаблон пути для генерации файлов с именами методов структурно-параметрической оптимизации; z_2 – список спецификаций методов; z_3 – список методов; z_4 – число поколений для методов; z_5 – размер популяции методов; z_6 – конфигурация структурно-параметрической оптимизации; z_7 и z_8 – расчетные БД; z_9 – список результатов структурно-параметрической оптимизации; z_{10} – наилучший метод, выбранный Парето-оптимальным правилом; z_{11} и z_{12} – журнал выполнения операций o_2 и o_3 соответственно; z_{13} содержит список экземпляров модулей и БД; z_{14} содержит агрегированные результаты структурно-параметрической оптимизации; z_{15}, z_{16} и z_{17} – конфигурация структурно-параметрической оптимизации после изменения метода, числа поколений и размера популяции; z_{18} – модель многокритериального выбора; z_{19} – критерии многокритериального выбора; z_{20} – приоритеты критериев; z_{21} – наилучший метод, выбранный лексикографическим правилом.

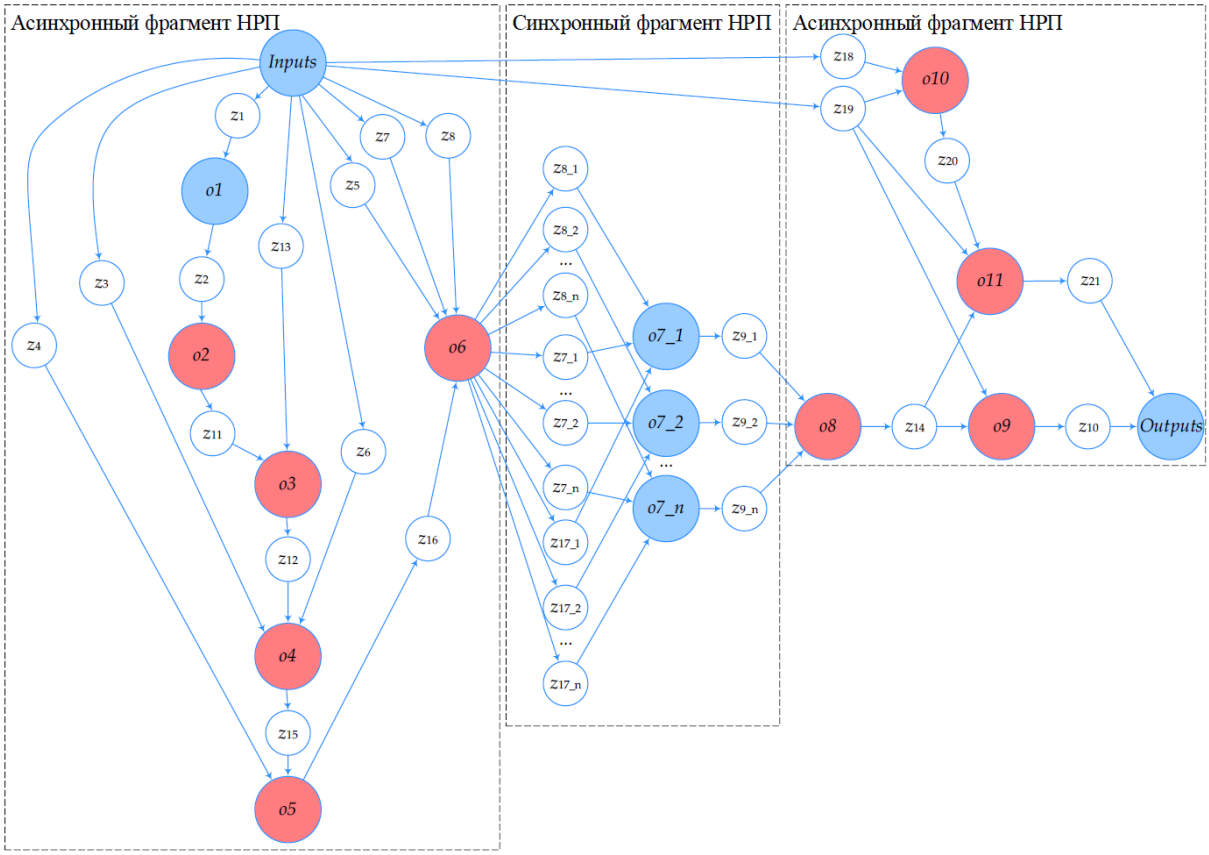


Рис. 4. Испытательный стенд (системный РП).

Экспериментальная среда развернута на облачных ресурсах провайдера Cloud.ru. Она включает 4 узла с 2-х ядерными процессорами и 2 ГБ оперативной памяти. Пропускная способность сети – 1 Гб/сек. Для эксперимента было проведено сравнение методов и алгоритмов управления вычислениями по двум схемам:

1) управление вычислениями и передача данных осуществляется с помощью алгоритмов и механизмов HTCondor;
2) управление вычислениями осуществляется с помощью разработанного алгоритма назначения ресурсов, а передача данных – с помощью предложенного механизма прямой передачи данных.

Распределение операций РП и их экземпляров по узлам приведено на рис. 5. В отличие от схемы 1, при распределении операций по схеме 2, на ярусе 7 обеспечивается баланси-ровка числа экземпляров операции o7, назначенных на узлы.

Ярус сценария решения задачи	Узел 1	Узел 2	Узел 3	Узел 4
1	o1 - 1	o10 - 1		
2	o2 - 1			
3	o3 - 1			
4	o4 - 1			
5	o5 - 1			
6	o6 - 1			
7	o7 - 18	o7 - 13	o7 - 14	o7 - 15
8	o8 - 1			
9	o9 - 1			
10	o11 - 1			

а)

Ярус сценария решения задачи	Узел 1	Узел 2	Узел 3	Узел 4
1	o1 - 1	o10 - 1		
2	o2 - 1			
3	o3 - 1			
4	o4 - 1			
5	o5 - 1			
6	o6 - 1			
7	o7 - 15	o7 - 15	o7 - 15	o7 - 15
8	o8 - 1			
9	o9 - 1			
10	o11 - 1			

б)

Рис. 5. Распределение операций: а) схема 1; б) схема 2

На рис. 6 (а) и 6 (б) приведены временные диаграммы выполнения заданий на узлах среды по схемам 1 и 2 соответственно. За счет предложенных в работе алгоритмов и механизмов удалось существенно сократить время выполнения РП.

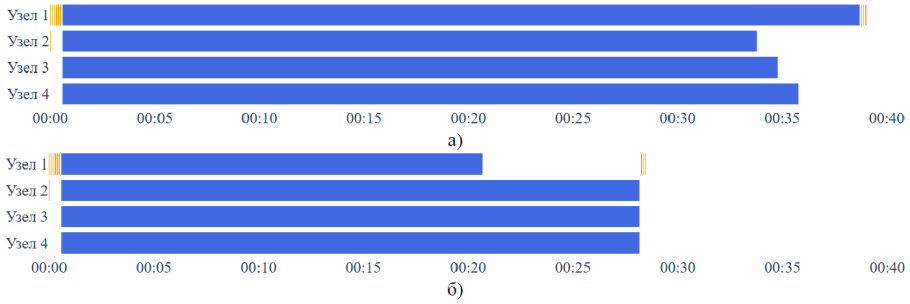


Рис. 6. Время выполнения заданий на узлах среды: а) схема 1; б) схема 2

На рис. 7 приведены показатели эффективности использования ресурсов. На рис. 7(а) и рис. 7(b) показано, что механизм прямой передачи данных позволил сократить долю времени на передачу данных относительно общего времени выполнения РП и средние накладные расходы на передачу данных соответственно. Алгоритм назначения ресурсов в схеме 2 обеспечил увеличение загрузки процессора в среднем более чем на 19% и ее балансировку в сравнении с результатами на рис. 7(в), полученными для схемы 1. Как видно из рис. 7(г), при использовании схемы 2 общее время выполнения РП сократилось более чем на 27%. Полученные результаты демонстрируют повышение эффективности использования ресурсов при использовании предложенного алгоритма назначения ресурсов и механизма прямой передачи данных. Время работы алгоритма при назначении ресурсов в данном эксперименте не превышает 2 мс.

Заключение

В рамках исследования получены следующие основные результаты:

- предложен механизм прямой передачи данных для системы управления прохождением заданий HTCondor, позволяющий сократить затраты на передачу данных за счет уменьшения числа их пересылок между вычислительными узлами;

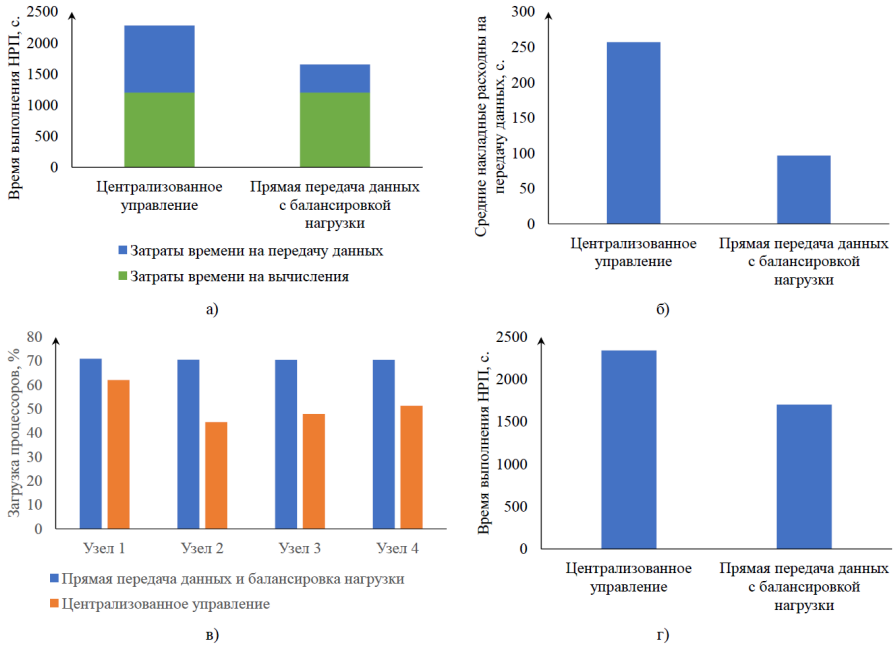


Рис. 7. Показатели эффективности использования ресурсов: а) затраты времени на вычисления и передачу данных; б) накладные расходы на передачу данных; в) загрузка процессоров; г) время выполнения РП.

- разработан эвристический алгоритм назначения ресурсов, который позволяет уплотнить расписание выполнения заданий и сбалансировать нагрузку на ресурсах в сравнении с алгоритмом HTCondor за счет учета структуры рабочего процесса и тестирования программных модулей на испытательных стендах;
- преимущества предложенного подхода продемонстрированы на примере выполнения рабочих процессов для исследования живучести локальной ЭС, расположенной на БПТ.

Дальнейшее направление исследований связано с дополнительным учетом стоимости и надежности выполнения заданий на узлах ГРВС при назначении ресурсов.

Список литературы

[1] Воскобойников М.Л., Феоктистов А.Г. Сравнительный анализ систем управления научными рабочими процессами // Информационные и математические технологии в науке и управлении. 2024. № 3(35). С. 102–111. DOI: 10.25729/ESI.2024.35.3.009.
[2] GridWay. URL: https://gridway.ucm.es/doku.php (дата обращения: 14.04.2026).
[3] Condor DAGMan. URL: https://htcondor.readthedocs.io/en/lts/automated-workflows/dagman-introduction.html (дата обращения: 14.04.2026).
[4] Slurm Workload Manager. URL: https://slurm.schedmd.com (дата обращения: 14.04.2026).
[5] TORQUE/Maui. URL: http://adaptivecomputing.com/cherry-services/torque-resource-manager (дата обращения: 14.04.2026).
[6] HTCondor. URL: https://htcondor.org (дата обращения: 14.04.2026).
[7] Fajardo E., Würthwein F., Jones R., Philpott S., Strosahl K. Limits of the HTCondor Transfer System // EPJ Web of Conferences. EDP Sciences, 2019. Vol. 214. 03008 p. DOI: 10.1051/epjconf/201921403008.
[8] Sfiligoi I. et al. HTCondor data movement at 100 Gbps // Proceedings of the 2021

IEEE 17th International Conference on eScience (eScience). IEEE, 2021. P. 239-240. DOI: 10.48550/arXiv.2107.03947.

[9] Weitzel D. J., Bockelman B., Swanson D. Distributed caching using the HTCondor CacheD // Parallel and Distrib. Process. Techn. and Appl. 2015.

[10] Феоктистов А.Г., Воскобойников М.Л., Черных А.Н. Разработка и применение сервис-ориентированных научных приложений в инструментальном комплексе FDE-SWFs // Труды ИСП РАН. 2024. Т. 36. № 6. С. 195–214. DOI: 10.15514/ISPRAS-2024-36(6)-11.

[11] Топорков В.В. Модели распределенных вычислений. М.: Физматлит, 2004. 320 с.

[12] Docker. URL: https://www.docker.com/ (дата обращения: 14.04.2026).

[13] Ansible. URL: https://www.redhat.com/en/ansible-collaborative (дата обращения: 14.04.2026).

[14] Горбунов–Посадов М.М., Корягин Д.А., Мартынюк В.В. Системное обеспечение пакетов прикладных программ. М.: Наука, 1990. 208 с.

[15] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Изд-во Мир, 1982. 420 с.

[16] Bychkov I.V., Feoktistov A.G., Voskoboinikov M.L., Edelev A., Beresneva N., Edeleva O. Optimization of Integrated Energy System Resilience // Информатика и автоматизация. 2025. Т. 24, № 3. С. 951–981. DOI: 10.15622/ia.24.3.8.

Учебный курс «Основы компьютерного зрения»*

В.Д. Кустикова

Нижегородский государственный университет им. Н.И. Лобачевского

Задачи компьютерного зрения возникают повсеместно при разработке интеллектуальных систем видеоанализа. Поэтому знание теоретических основ и развитие практических навыков решения задач компьютерного зрения являются неотъемлемой частью подготовки студентов ИТ-специальностей. В лекционной части курса «Основы компьютерного зрения» рассматриваются различные виды цифровых изображений и базовые алгоритмы их обработки. Дается основная информация о библиотеке OpenCV, которая предоставляет необходимый функционал для построения прототипов приложений компьютерного зрения. Вводятся основные задачи машинного обучения и широко известные алгоритмы их решения. Рассматриваются базовые принципы построения глубоких нейросетевых моделей, приводятся примеры моделей, обеспечивающих решение таких задач компьютерного зрения, как склеивание перекрывающихся изображений, классификация изображений, детектирование объектов, семантическая сегментация изображений, перенос стилей. Практика включает решение классических задач компьютерного зрения на реальных изображениях и видео.

Ключевые слова: учебный курс, компьютерное зрение, OpenCV, глубокое обучение

1. Введение

Компьютерное зрение [1, 2] – активно развивающаяся область, отвечающая за разработку алгоритмов и технологий, которые способны извлекать информацию из изображений и анализировать ее. Сложность задач компьютерного зрения обусловлена разным восприятием визуальной информации человеком и компьютером, а также неоднозначностью ее интерпретации. В настоящее время системы анализа видео и изображений занимают неотъемлемое место во многих сферах жизни каждого человека, упрощая значительное число обыденных операций таких, как взвешивание товаров с помощью «умных» весов, оплата квитанций с использованием сканера QR-кодов, аутентификация в различных сервисах (Интернет-банк, электронная почта). Многие задачи обработки и анализа изображений внутри подобных систем требуется решать в режиме реального времени, вследствие чего оптимизация и распараллеливание алгоритмов являются неотъемлемыми этапами разработки. Поэтому знание и понимание основных задач компьютерного зрения и базовых алгоритмов их решения необходимо в процессе подготовки ИТ-специалистов.

Цель разрабатываемого учебного курса «Основы компьютерного зрения» состоит в том, чтобы сформировать общее представление о спектре прикладных задач анализа изображений/видео и методов их решения, а также обучить базовым навыкам разработки приложений компьютерного зрения, изучить основы работы с открытой библиотекой OpenCV [3, 4], которая является стандартом де-факто в данной области.

*Работа выполнена при поддержке Аналитического Центра при Правительстве Российской Федерации (договор № 70-2025-000821 от 04.06.2025).

Статья построена следующим образом. Вначале дается обзор существующих открытых курсов на русском языке по компьютерному зрению. Далее описывается структура теоретической и практической частей курса, поясняется логика изложения материала. В заключении приводятся результаты реализации и перспективы развития курса.

2. Обзор существующих курсов

К настоящему моменту разработано значительное число образовательных курсов по тематике компьютерного зрения. Ниже (таблица 1) приведена краткая информация о релевантных курсах последних нескольких лет на русском языке, которые читаются в рамках образовательных программ бакалавриата и магистратуры ВУЗов, а также доступные онлайн-курсы для исследователей и разработчиков, ведущих деятельность в данной области. Следует отметить, что перечень ВУЗов, где читаются курсы, охватывающие различные разделы компьютерного зрения, достаточно широкий (МГУ им. М.В. Ломоносова, ИТМО, МФТИ, НИУ ВШЭ и другие), в таблице приведены лишь некоторые из них. В целом во всех курсах отражаются тенденции успешного применения глубокого обучения в задачах компьютерного зрения. Поэтому в материалах рассматриваются как широко известные нейросетевые модели, так и описывается подход **переноса обучения (transfer learning)** [5, 6], предполагающий использование архитектур моделей и обученных весов для решения схожих по смыслу задач.

Многие курсы [7, 8, 10–14] построены на последовательном рассмотрении типовых нейросетевых моделей для решения таких задач, как классификация изображений, детектирование объектов на изображениях, сегментация объектов (instance segmentation) и изображений (semantic segmentation), сопровождение (tracking) объектов. В курсах [7, 8, 10, 13, 14] также рассматривается применение генеративных моделей (автокодировщики, генеративно-состязательные сети, диффузионные модели) для генерации синтетических изображений. Указанная задача является не менее актуальной и востребованной, поскольку существует множество прикладных областей, в которых подготовка разметки реальных наборов данных для обучения глубоких моделей является затруднительной. Классические алгоритмы компьютерного зрения рассматриваются только в курсах [8, 9, 12]. При этом [9] ориентирован на прикладную область обработки и анализа аэрокосмических изображений.

Если обратиться к практической части приведенного перечня образовательных курсов, то наблюдается использование двух подходов. Первый подход [7, 9, 11, 12] состоит в выполнении последовательности небольших заданий, обеспечивающих освоение инструментария и решение классических задач компьютерного зрения. При этом предполагается, что слушатели как используют существующие глубокие модели, так и обучают собственные с целью решения конкретных проблем. Второй подход [10, 13, 14] предусматривает проектную работу, целью которой является максимальное покрытие тем и вопросов, рассматриваемых в теоретической части курса. Следует отметить, что в [9, 12] наряду с практическими задачами также предусмотрен итоговый проект.

Таблица 1. Структура доступных учебных курсов по компьютерному зрению. В таблице используются наименования, приведенные в оригинальных программах курсов. Сокращения: CNN (Convolutional Neural Network) – сверточная нейронная сеть, CV (Computer Vision) – компьютерное зрение, ML (Machine Learning) – машинное обучение.

Учебный курс, организация, лекции/практика (ч) (продолжительность)	**Теоретическая часть**	**Практическая часть**
«Компьютерное зрение» [7], МФТИ, ШАД, МГУ (ВМК), ИТМО и другие ВУЗы. Часы не указаны (зависят от ВУЗа)	1. Введение в предмет. Цифровое изображение, свет и цвет. 2. Основы обработки изображений. 3. Сжатие изображений, преобразование Фурье. 4. Классификация изображений. Введение в нейросети. 5. Сверточные нейросетевые архитектуры. 6. Трансформеры и сверточные нейронные сети с большими ядрами. 7. Метрическое обучение. 8. Детекторы объектов. 9. Сегментация изображений. 10. Основы обработки видео. 11. Обучение без разметки. Фундаментальные модели. 12. Перенос стиля. Синтез изображений. 13. Вариационные автоэнкодеры. Генеративно-состязательные сети. 14. Чтение статей CV/ML.	Семинарские занятия: 1. Работа с NumPy. 2. Основы обработки изображений. 3. Преобразование Фурье. 4. Обучение нейросетей: NumPy, PyTorch. 5. Обучение нейросетей. Базовые приемы. 6. Обучение нейросетей. Продвинутые приемы. 7. Метрическое обучение. 8. Практические аспекты CV/ML в индустрии. 9. Сегментация изображений. 10. Работы с видео. Оптический поток. 11. Обучение без разметки. Метод MoCo. 12. Вариационные автоэнкодеры. Генеративно-состязательные сети. 13. Диффузионные генеративные модели. Самостоятельная работа предполагает решение последовательности небольших заданий. Задания являются закрытыми.

Продолжение на следующей странице

Таблица 1 – продолжение

Учебный курс, организация, лекции/практика (ч) (продолжительность)	**Теоретическая часть**	**Практическая часть**
«Компьютерное зрение» [8], НИУ ВШЭ 18/18	1. Основные задачи компьютерного зрения. 2. Введение в компьютерное зрение. Цифровое изображение и тональная коррекция. 3. Сверточные нейросети для классификации и поиска похожих изображений. 4. Детекторы объектов. 5. Операция свертки, сверточные нейронные сети. 6. Сегментация изображений. 7. Методы машинного обучения в задачах классификации изображений. 8. Генеративно-состязательные сети и вариационные автокодировщики.	Информация отсутствует на странице курса.
«Компьютерное зрение и машинное обучение в анализе изображений» [9], НИУ ВШЭ 14/18	1. Компьютерное зрение в задачах обработки и анализа данных дистанционного зондирования. 2. Машинное обучение – смежная дисциплина, позволяющая выполнять автоматизированное дешифрирование аэрокосмических снимков. 3. Методы машинного обучения в задачах классификации изображений. 4. Компьютерное зрение в задачах распознавания движущихся объектов.	Практическая работа 1. Алгоритмы компьютерного зрения в анализе изображений. Практическая работа 2. Тематическая обработка видеоизображений и распознавание движущихся объектов. Практическая работа 3. Алгоритмы классического и глубокого обучения в анализе изображений. Проект. Машинное обучение в задачах классификации изображений.

Продолжение на следующей странице

Таблица 1 – продолжение

Учебный курс, организация, лекции/практика (ч) (продолжительность)	**Теоретическая часть**	**Практическая часть**
«Компьютерное зрение – CV» [10], Яндекс Практикум 8 недель	1. Нейросетевые решения на практике. 2. Детекция объектов (YOLO-модели, SSD, Faster R-CNN). Постобработка и визуализация результатов. 3. Сегментация изображений (архитектуры типа «кодировщик-декодировщик»). 4. Генерация изображений (автокодировщик, генеративно-состязательные сети, диффузионные модели). 5. Трансформеры и мультимодальные модели.	Проект 1. Использование предобученных моделей для анализа текста и изображений, интерпретация результатов их работы. Проект 2. Создание высокоточного детектора на кастомном наборе данных. Проект 3. Разработка модели сегментации для решения бизнес-задачи. Проект 4. Кастомизация генеративной сети под конкретный запрос. Проект 5. Обучение классификационной модели на трансформерной архитектуре.
«Нейронные сети и компьютерное зрение» [11], Stepik. Продолжительность не указана	1. Нейрон и нейронная сеть. 2. Строим первую нейронную сеть (компоненты нейронной сети, настройка нейронной сети). 3. Задачи, решаемые при помощи нейронных сетей. 4. Методы оптимизации. 5. Сверточные нейронные сети. 6. Регуляризация и нормализация. 7. Метод максимального правдоподобия и большой финал.	Семинар 1. Базовая работа в PyTorch. Семинар 2. Реализация градиентного спуска. Семинар 3. Строим первую нейронную сеть. Семинар 4. Классификация в PyTorch. Семинар 5. Классификация рукописных чисел полносвязанной сетью. Семинар 6. Распознавание рукописных чисел сверточной нейросетью. Семинар 7. Решаем задачу классификации на датасете CIFAR.

Продолжение на следующей странице

Таблица 1 – продолжение

Учебный курс, организация, лекции/практика (ч) (продолжительность)	**Теоретическая часть**	**Практическая часть**
«Современное компьютерное зрение» [12], Stepik. Продолжительность не указана	1. Классические методы Computer Vision (основы обработки изображений, операторы градиента, бинаризация и трешхолдинг, другое). 2. CNN и первые архитектуры (основы сверточных сетей, сеть LeNet, AlexNet, VGG, ResNet). 3. Современные архитектуры CV (Visual Transformer, MobileNet, EfficientNet). 4. Детекция (задача, модель R-CNN и ее развитие, модели YOLO). 5. Сегментация (U-Net, Deeplab, Mask R-CNN, Segment Anything Model). 6. Трекинг и работа с видеопотоками (задача, оценивание качества, алгоритм трекинга и другое).	Практика 1. Аугментация и MBConv. Практика 2. Детекция на практике. Практика 3. Сегментация. Практика 4. Захват видео с камер в реальном времени. Итоговый проект курса.
«Компьютерное зрение» [13], Otus 4 месяца	1. Нейронные сети и инструменты CV. 2. Архитектуры нейронных сетей (сверточные сети, трансформеры в задачах компьютерного зрения, CLIP-модели). 3. Стандартные задачи CV (детектирование объектов, сегментация, детектирование и распознавание лиц, оценка позы, геометрические методы компьютерного зрения). 4. Генеративные модели (автокодировщики, генеративно-состязательные сети, диффузионные модели).	Выполнение проектов по тематике компьютерного зрения. Возможные темы проектов: 1. Удаление объектов с фото. 2. Выделение описания фото из текста. 3. Поиск/удаление брендов на фото/видео. 4. Генерация персонального аватара в заданном стиле. 5. Озвучивание видео. 6. Проект на выбор слушателя курса.

Продолжение на следующей странице

Таблица 1 – продолжение

Учебный курс, организация, лекции/практика (ч) (продолжительность)	**Теоретическая часть**	**Практическая часть**
«Компьютерное зрение. Advanced» [14], Otus 4 месяца	1. Рабочее окружение и библиотеки для CV. 2. Нейронные сети и глубокое обучение. 3. Стандартные задачи CV. 4. Генеративные модели. 5. Продвинутые методы CV. 6. Оптимизация, инференс и подготовка к продакшену.	Выполнение проектов по тематике компьютерного зрения. Возможные темы проектов не опубликованы в открытом доступе.

Обзор курсов показывает, что они в основном ориентированы на изучение моделей глубокого обучения для решения четырех классических задач компьютерного зрения. При этом на практике множество задач не ограничивается классификацией изображений, детектированием объектов, сегментацией и сопровождением объектов. Существует значительное количество задач, где успешно работают традиционные подходы, в которые интегрированы элементы нейросетевых. Отметим, что изучение приведенных курсов предполагает наличие теоретической базы в области машинного и глубокого обучения, что ограничивает круг слушателей, способных успешно их освоить. При разработке курса «Основы компьютерного зрения» планируется учесть эти два важных момента, поскольку курс предполагается читать для студентов бакалавриата технических и естественно-научных специальностей.

3. Структура курса

3.1. Описание курса

В курсе «Основы компьютерного зрения» излагаются следующие ключевые моменты:
1. Виды цифровых изображений. Основные операции над ними.
2. Возможности библиотеки OpenCV, типовой цикл ее разработки и использования, а также базовые функции работы с изображениями.
3. Классификация задач машинного обучения и примеры приложений в компьютерном зрении.
4. Общая информация о моделях и методах глубокого обучения, результаты решения классических задач компьютерного зрения (классификация изображений, детектирование объектов на изображениях, семантическая сегментация изображений).
5. Классические и современные нейросетевые подходы к решению различных задач компьютерного зрения.

Курс включает в себя 16 часов лекций, 16 часов практических занятий и 16 часов самостоятельной работы (таблица 2). Теоретическая часть курса включает 12 тем: 8 основных тем; 4 темы для самостоятельного изучения (таблица 2, выделены темно-серым), первые три из которых обеспечивают выравнивание базовых знаний студентов разных направлений обучения, и одна является дополнительной. Лекции проводятся

в классической форме с элементами мастер-класса и сопровождаются примерами решения практических задач с использованием библиотеки OpenCV. В процессе выполнения практических работ осуществляется последовательное решение задач обработки, классификации изображений и детектирования объектов.

Промежуточный контроль знаний предполагает тестирование по теоретической части курса, представление результатов выполнения практических заданий и анализ достигнутых показателей качества решения задач. Итоговый контроль знаний предусматривает собеседование с преподавателем по теоретическим вопросам курса. Предполагаемая форма отчетности – зачет.

Курс ориентирован на студентов и аспирантов технических и естественно-научных специальностей, а также инженеров ИТ-компаний, которые меняют сферу деятельности, преподавателей вузов и исследователей. Предполагается, что слушатели имеют базовые навыки разработки программ на языках C++ и Python. Наряду с этим, изучение курса требует наличия теоретических знаний и практических навыков в следующих областях: дискретная математика, линейная алгебра, методы оптимизации, алгоритмы и структуры данных. Наличие базовых знаний в области машинного и глубокого обучения ускорит освоение материала, но не является обязательным ввиду наличия необходимых материалов для самостоятельного изучения.

Таблица 2. Структура учебного курса «Основы компьютерного зрения».

Тема	**Содержание**	**Часы**
Лекция 1. Введение в компьютерное зрение.	1. Что такое компьютерное зрение? 2. В чем сложность задач компьютерного зрения? 3. Примеры задач компьютерного зрения. 4. Цифровые изображения: бинарные, полутоновые и мультиспектральные изображения. Основные операции. 5. Цветные изображения. Цветовые пространства. 6. Классические задачи компьютерного зрения: классификация изображений, детектирование объектов на изображениях, семантическая сегментация изображений, сопровождение объектов на кадрах видеопотока. 7. Литература и программные инструменты.	2

Продолжение на следующей странице

Таблица 2 – продолжение

Тема	**Содержание**	**Часы**
Лекция 2.* Библиотека OpenCV.	1. Краткая информация о библиотеке. 2. Схема разработки и состав библиотеки OpenCV. 3. Типовая схема использования библиотеки OpenCV. 4. С чего начать? (официальная страница, документация, FAQ) 5. Базовые структуры данных и операции над ними: точка, прямоугольник (и другие геометрические фигуры), матрица. 6. Модуль для разработки интерфейса highgui. Примеры приложений. 7. Модуль обработки изображений imgproc. Примеры приложений.	4
Практическая работа 1. Обработка изображений с использованием библиотеки OpenCV.	Задача состоит в том, чтобы разработать библиотеку фильтров с помощью базовых операций над изображениями (матрицами) в OpenCV. Простейшие примеры фильтров: перевод изображения в оттенки серого, изменение разрешения изображения, фотоэффект сепии. *Примечания:* – набор фильтров для реализации может выбираться индивидуально для каждого слушателя или группы слушателей (задания по вариантам).	4
Лекция 3.* Введение в машинное обучение.	1. Что такое машинное обучение? 2. Классификация задач машинного обучения. 3. Задача обучения с учителем. 4. Задача обучения без учителя: кластеризация. 5. Основные ресурсы. 6. Программные пакеты.	4
Лекция 4.* Основы глубокого обучения.	1. Что такое глубокое обучение? 2. Общая схема решения задач компьютерного зрения с помощью глубокого обучения. 3. Классификация нейросетевых моделей по способу обучения. 4. Полносвязные нейронные сети. 5. Сверточные нейронные сети: схема построения, основные операции. Пример модели AlexNet. 6. Перенос обучения (transfer learning) глубоких нейросетевых моделей. 7. Примеры обучения/тестирования глубоких сетей средствами PyTorch. 8. Примеры вывода нейронных сетей средствами модуля DNN библиотеки OpenCV.	4

Продолжение на следующей странице

Таблица 2 – продолжение

Тема	**Содержание**	**Часы**
Лекция 5. Классификация изображений. Классические методы компьютерного зрения.	1. Математическая постановка задачи. 2. Наборы данных. 3. Показатели качества решения задачи классификации. 4. Детекторы и дескрипторы ключевых точек. 5. Методы группы «мешок слов» (bag-of-words).	2
Лекция 6. Склеивание изображений.	1. Введение. 2. Постановка задачи склеивания изображений. 3. Общая схема решения задачи. 4. Классические и нейросетевые методы сопоставления признаковых описаний. 5. Классические и нейросетевые методы регистрации изображений. 6. Классические и нейросетевые методы наложения изображений.	2
Лекция 7. Классификация изображений. Нейросетевые модели.	1. Хронология развития моделей и изменение качества решения задачи классификации на наборе данных ImageNet. 2. Примеры конкретных архитектур моделей: модели VGG, модели ResNet, модели Inception, модели DenseNet, трансформеры.	2
Практическая работа 2. Классификация изображений с использованием библиотеки OpenCV.	Задача состоит в том, чтобы разработать два приложения для классификации изображений в некотором наборе данных. 1. Приложение, которое реализует алгоритм «мешок слов». При реализации можно использовать как классические, так и нейросетевые дескрипторы особых точек. 2. Приложение, которое использует перенос обучения глубоких нейронных сетей. *Примечания:* – набор данных может выбираться индивидуально для слушателя или группы слушателей; – набор данных выбирается в начале прочтения курса на усмотрение преподавателя; – преподаватель может предложить слушателям подготовить собственный набор данных.	8

Продолжение на следующей странице

Таблица 2 – продолжение

Тема	**Содержание**	**Часы**
Лекция 8. Детектирование объектов на изображениях.	1. Математическая постановка задачи. 2. Наборы данных. 3. Показатели качества решения задачи детектирования объектов. 4. Алгоритмы детектирования объектов. Методы поиска по шаблону. Методы, основанные на извлечении признаков. Метод «скользящего» окна. Пирамида масштабов. Нейросетевые модели (двухстадийные и одностадийные модели). Модели R-CNN, SSD и другие.	2
Практическая работа 3. Детектирование объектов на изображениях с использованием библиотеки OpenCV.	Задача состоит в том, чтобы разработать приложение для детектирования объектов на последовательности изображений или кадрах видеопотока с использованием обученных нейронных сетей. Набор изображений/видео выдается преподавателем. Цель работы – максимизировать качество решения задачи детектирования. *Примечания:* – набор данных может выбираться индивидуально для слушателя или группы слушателей; – набор данных выбирается в начале прочтения курса на усмотрение преподавателя; – преподаватель может предложить слушателям подготовить собственный набор данных.	4
Лекция 9. Отслеживание движения и сопровождение объектов на видео.	1. Движение в компьютерном зрении. Основные группы задач. 2. Определение областей движения (относительно неподвижный фон). Постановка задачи. Основные методы решения задачи: вычитание фона, полный перебор допустимых вариантов смещения изображений или их фрагментов, параметрические модели движения, вычисление оптического потока. 3. Сопровождение объектов. Постановка задачи. Классификация методов сопровождения. Методы сопровождения объектов, основанные на вычислении оптического потока.	2
Лекция 10. Семантическая сегментация изображений.	1. Постановка задачи. 2. Открытые наборы данных. 3. Показатели качества семантической сегментации изображений. 4. Глубокие модели для семантической сегментации изображений. 5. Сравнение моделей семантической сегментации изображений.	2

Продолжение на следующей странице

Таблица 2 – продолжение

Тема	**Содержание**	**Часы**
Лекция 11. Перенос стиля и синтез изображений.	1. Постановка задачи. 2. Открытые наборы данных. 3. Классические методы. 4. Нейросетевые модели. Модели на базе традиционных сетей прямого распространения. Модели на базе генеративно-состязательных сетей. Диффузионные модели.	2
Лекция 12.* Обучение представлений без учителя.	1. Проблемы и предпосылки развития обучения без учителя. 2. Автокодировщики. 3. Ограниченная машина Больцмана. 4. Глубокая машина Больцмана. 5. Глубокая сеть доверия.	4

3.2. Теоретическая часть

Рассмотрим более детально теоретическую часть курса. Лекция 1 дает общее понимание о том, какие задачи решаются с использованием алгоритмов компьютерного зрения, и почему эти задачи являются сложными. В материалах рассматриваются различные виды цифровых изображений и базовые алгоритмы их обработки. Наряду с этим, описываются постановки трех классических задач компьютерного зрения: классификация изображений, детектирование объектов на изображениях и сопровождение объектов на видео. Приводятся конкретные прикладные области, где указанные задачи возникают. В лекции 2 дается обзор широко известной библиотеки компьютерного зрения OpenCV. Приводится архитектура и цикл разработки, а также типовая схема использования библиотеки. Рассматриваются основные источники, которые необходимо использовать в процессе разработки приложений средствами OpenCV. Демонстрируются примеры, покрывающие базовый функционал библиотеки. Поскольку значительное количество методов компьютерного зрения используют машинное и глубокое обучение, то лекции 3 и 4 содержат постановки задач обучения с учителем и без учителя, а также вводную информацию о классических моделях и методах для их решения. Лекция 5 рассматривает задачу классификации изображений и классические подходы к ее решению, основанные на извлечении признаков (детекторы и дескрипторы ключевых точек) и применении методов группы «мешок слов». Лекция 6 посвящена классическим и нейросетевым методам решения задачи склеивания изображений, которая включает сопоставление признаковых описаний, регистрацию (поиск преобразования) и наложение изображений. Лекция 7 предполагает рассмотрение конкретных глубоких нейросетевых моделей для классификации изображений и изучение вопросов качества работы этих моделей на одном из самых крупных наборов данных ImageNet [15]. В лекции 8 описываются глубокие сети для детектирования объектов разных классов на изображениях. Приводятся особенности моделей по сравнению с классификационными, типовые наборы данных, на которых выполняется обучение, и метрики качества детектирования. Лекция 9 рассматривает задачи отслеживания движения и сопровождения объектов, классические методы решения этих задач, в частности, вычитание фона, вычисление оптического потока. Лекция 10 посвящена задаче семантической сегментации изображений и глубоким

моделям, обеспечивающим решение указанной задачи на различных наборах данных. Лекция 11 затрагивает вопросы переноса стиля на изображение и синтеза изображений с использованием современных нейросетевых моделей, построенных на базе генеративно-состязательных сетей и диффузионных моделей. В лекции 12 рассматриваются типовые модели, обучаемые в отсутствии размеченных данных (без учителя), которые могут использоваться для построения эффективного признакового представления изображений.

3.3. Практическая часть

Практическая часть курса предполагает выполнение трех заданий, которые покрывают существенную часть пайплайна многих приложений (обработка изображений, классификация, детектирование и распознавание объектов). Первая работа является вводной и направлена на освоение базовых операций обработки изображений и получение навыков их реализации с использованием примитивных операций библиотеки OpenCV. Студентам предлагается разработать библиотеку фильтров изображений, в число которых входят масштабирование, пикселизация, наложение простой и фигурной рамок, фильтры сепии и виньетки. Во второй практической работе предполагается решение задачи классификации изображений с использованием классического метода «мешок слов» и переноса обучения глубоких нейронных сетей, которые в настоящее время демонстрируют лучшие результаты. Конкретная задача и набор данных выбирается на усмотрение преподавателя и может меняться при разных прочтениях курса. Программная реализация выполняется на базе функционала библиотек OpenCV и PyTorch [16]. Цель работы состоит в достижении максимального качества решения задачи. В третьей работе студентам предлагается решить задачу детектирования объектов на изображениях/кадрах видеопотока с использованием существующих глубоких нейросетевых моделей и оценить качество детектирования. Программная реализация выполняется с помощью модуля DNN библиотеки OpenCV и различных библиотек глубокого обучения.

4. Реализация курса

Разрабатываемый курс встроен в качестве обязательного спецкурса и спецкурса по выбору в основную программу обучения студентов бакалавриата (4 курс, 7 семестр) Института информационных технологий, математики и механики ННГУ им. Н.И. Лобачевского, специализирующихся по направлениям «Прикладная математика и информатика», «Фундаментальная информатика и информационные технологии», «Программная инженерия». Следует отметить, что по отдельным профилям обучения перечисленных направлений подготовки в рамках программы предусмотрено предварительное прочтение курсов «Машинное обучение», «Глубокое обучение» и «Обработка изображений». Поэтому предполагается самостоятельное изучение нескольких вводных тем курса, либо эти темы рассматриваются для заинтересованного круга студентов на первых практических занятиях или дополнительных лекциях. Практические занятия включают проведение тестирований по отдельным теоретическим темам курса и сдачу практических работ. Сдача работ предполагает демонстрацию запуска приложения (сценарии выполнения устанавливаются в требованиях) и собеседование по коду (ответы на технические и теоретические вопросы). Итоговая оценка знаний за курс (зачет/не зачет) определяется как сумма баллов, полученных в процессе тестирований и сдачи практических работ. В случаях пограничного балла предусматривается собеседование с преподавателем по теоретической части курса.

Заключение

Компьютерное зрение – востребованная и активно развивающаяся область, которая в настоящее время успешно работает во многих сферах. Поэтому выпускники ИТ-специальностей должны иметь необходимый профессиональный кругозор, позволяющий решать реальные прикладные задачи. Курс «Основы компьютерного зрения» направлен на то, чтобы предоставить необходимые для этого базовые знания и навыки.

К моменту подачи работы подготовлены черновики материалов десяти лекций согласно плану курса (таблица 2). Наряду с этим, выполнена необходимая подготовительная работа для последующей разработки материалов практической части курса. К началу сентября планируется полностью разработать презентации лекционной части курса, описание практических заданий с перечнем возможных вариаций. Все материалы предполагается выложить в открытый доступ на сайте университета.

Развитие курса напрямую связано с увеличением количества часов, которое отводится на прочтение теоретического материала и выполнение практических заданий, и изменением отчетности по курсу. Один из возможных путей развития – углубленное рассмотрение методов и нейросетевых моделей для решения отдельных задач компьютерного зрения, а также усиление практической части за счет проектной работы, которая предусматривает решение прикладных задач, требующих от студентов прохождения полного цикла разработки приложений компьютерного зрения.

Список литературы

[1] Szeliski R. Computer Vision. Algorithms and Applications // Springer Cham, 2023.
[2] Bishop C.M. Pattern Recognition and Machine Learning. 2006. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf (дата обращения 26.03.2026).
[3] Adrian K., Bradski G. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, 2017.
[4] OpenCV. URL: https://opencv.org (дата обращения 26.03.2026).
[5] Pan S.J., Yang Q. A Survey on Transfer Learning // IEEE Transactions on Knowledge and Data Engineering. 2010. 22 (10). P.1345-1359.
[6] Plested J., Phiri M., Gedeon T. Deep transfer learning for image classification: a survey. Artificial Intelligence Review. 59 (100). 2026.
[7] Курс «Компьютерное зрение», МФТИ, ШАД, ВМК МГУ, ИТМО. URL: https://vkvideo.ru/playlist/-227352629_7 (дата обращения 04.06.2026).
[8] Курс «Компьютерное зрение», НИУ ВШЭ. URL: https://dp.hse.ru/#/summary?implementationId=16281515491 (дата обращения 26.03.2026).
[9] Курс «Компьютерное зрение и машинное обучение в анализе изображений», НИУ ВШЭ. URL: https://dp.hse.ru/#/summary?implementationId=16281515082 (дата обращения 26.03.2026).
[10] Курс «Компьютерное зрение – CV», Яндекс Практикум. URL: https://practicum.yandex.ru/computer-vision (дата обращения 26.03.2026).
[11] Курс «Нейронные сети и компьютерное зрение», Stepik. URL: https://stepik.org/course/50352/promo (дата обращения 26.03.2026).
[12] Курс «Современное компьютерное зрение», Stepik. URL: https://stepik.org/course/223852/promo (дата обращения 26.03.2026).
[13] Курс «Компьютерное зрение», Otus. URL: https://otus.ru/lessons/cv (дата обращения 26.03.2026).

[14] Курс «Компьютерное зрение. Advanced», Otus. URL: https://otus.ru/lessons/cv-advanced (дата обращения 26.03.2026).
[15] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A.C., Fei-Fei L. ImageNet Large Scale Visual Recognition Challenge // International Journal of Computer Vision, 2015.
[16] PyTorch. URL: https://pytorch.org (дата обращения 26.03.2026).

Учебный курс «Программирование для FPGA»*

Е.А. Козинов

Нижегородский государственный университет им. Н.И. Лобачевского

Курс посвящен основам проектирования цифровых устройств с использованием программируемых логических интегральных схем (FPGA) и охватывает полный цикл разработки: от понимания элементной базы до практической реализации и тестирования проектов. Для описания аппаратуры используется язык SystemVerilog. В процессе обучения студенты начинают с проектирования простых устройств, таких как сумматор на логических элементах, и завершают разработкой ядра процессора на стековой архитектуре. Для понимания полного стека разработки студенты также реализуют транслятор на основе ANTLR, обеспечивающий компиляцию программного кода для спроектированного процессорного ядра. Особое внимание уделяется тестированию: рассматриваются такие инструменты верификации, как ModelSim и Verilator, позволяющие проводить как функциональную, так и временную симуляцию разрабатываемых цифровых устройств.

Ключевые слова: учебный курс, FPGA, ПЛИС, SystemVerilog, тестирование, трансляторы

1. Введение

FPGA (программируемые логические интегральные схемы) [1, 2] – класс вычислительных устройств, обеспечивающий баланс между производительностью специализированных интегральных схем (ASIC) и гибкостью классических процессоров общего назначения. Сложность освоения FPGA обусловлена необходимостью сочетания знаний цифровой схемотехники, языков описания аппаратуры и низкоуровневых знаний об архитектуре внешних вычислительных устройств, с которыми FPGA взаимодействует, что отличается от традиционного подхода к программированию на центральных процессорах. FPGA могут быть задействованы как при решении практических задач, где необходима специализированная обработка информации, так и для прототипирования будущих цифровых устройств. В настоящее время FPGA находят применение в таких областях, как высокопроизводительные вычисления [3], обработка сигналов [4], а также ускорение алгоритмов глубокого обучения [5]. Курсы по основам проектирования цифровых устройств преподаются в ведущих вузах [6-9], а также крупных отечественных компаниях [10, 11]. Как следствие, знание и понимание основ проектирования на базе FPGA необходимо в процессе подготовки высококвалифицированных ИТ-специалистов, способных разрабатывать эффективные и адаптируемые аппаратные решения.

Цель разрабатываемого учебного курса «Программирование для FPGA» состоит в том, чтобы сформировать общее представление о методах разработки, описания и тестирования цифровых устройств на FPGA с применением SystemVerilog [2, 12] и широко распространенных инструментов таких как Quartus [13], ModelSim [14]

*Работа выполнена при поддержке Аналитического Центра при Правительстве Российской Федерации (договор No 70-2025-000821 от 04.06.2025).

и Verilator [15]. Для более полного понимания стека разработки курс также предполагает создание транслятора (на основе ANTLR [16]) для разработанного устройства, преобразующего код на естественном языке в формат набора машинных инструкций.

Статья построена следующим образом. Вначале дается описание структуры курса. Далее представлена более полная характеристика теоретической и практической частей курса. В заключении приводятся результаты реализации и перспективы развития курса.

2. Структура курса

2.1. Описание курса

В курсе «Программирование для FPGA» излагаются следующие ключевые моменты:

1. Сравнение парадигм – C++, ASIC, FPGA.
2. Основы схемотехники для получения основных логических операций необходимых в построении АЛУ.
3. Принципы описания интегральных схем с использованием SystemVerilog.
4. Подходы к описанию комбинационной и последовательностной логики.
5. Методы тестирования интегральных схем с использованием ModelSim и Verilator.
6. Принципы построения многотактовых вычислительных схем на основе конечных автоматов.
7. Описание интегральной схемы для простейшего процессора на основе стековой архитектуры и реализация транслятора для описанного устройства.
8. Методы взаимодействия с внешними устройствами на основе UART протокола.

Курс включает в себя 12 часов лекций (6 лекций продолжительностью 2 академических часа) и 12 часов практических занятий (таблица 1). Лекции проводятся в классической форме с элементами мастер-класса и сопровождаются примерами решения практических заданий. В процессе выполнения практических работ осуществляется последовательное решение задач, ведущих к разработке собственного процессора и транслятора для него. Итоговый контроль знаний предполагает представление результатов выполнения практических заданий, тестирование интегральных схем на тестовой плате и анализ достигнутых показателей качества решения задач.

Курс ориентирован на студентов технических и естественно-научных специальностей. Предполагается, что слушатели имеют базовые навыки разработки программ на языках C++ и Python. Наряду с этим, изучение курса требует наличия теоретических знаний и практических навыков в следующих областях: теоретические основы информатики, алгоритмы и структуры данных, дискретная математика. Наличие базовых знаний в области схемотехники ускорит освоение материала, но не является обязательным.

Таблица 1. Структура учебного курса «Программирование для FPGA».

Тема	**Содержание**	**Часы**
Лекция 1. Введение в FPGA.	1. Сравнение парадигм – C++, ASIC, FPGA. 2. NMOS и PMOS транзисторы. Принципы построения логических операций. 3. SystemVerilog – основы синтаксиса. 4. Комбинационная и последовательностная логика. 5. Quartus – основы работы в среде: – создание проекта; – реализация простейшего модуля с управлением светодиодами; – компиляция; – RTL; – Pin Planner; – запуск проекта на отладочной плате.	2
Практика 1. Комбинационная и последовательностная логика.	1. Комбинационная логика. Реализация сумматора и вычитателя на основе логических операций. Значения считываются со свитчей. Вывод на светодиодах. 2. Последовательностная логика. Значения считываются со свитчей по нажатию кнопки. Вычисления выполняются по нажатию кнопки. Вывод на светодиодах.	2
Лекция 2. Тестирование интегральных схем.	1. Реализация тестовых стендов на основе SystemVerilog. 2. Инструменты запуска тестов: – ModelSim – основы работы. Запуск тестов; – Verilator – генерация проекта C++, компиляция, запуск. Примеры проектов: – Комбинационная логика. Реализации декодера для 7-сегментного индикатора; – Последовательностная логика. Разработка секундомера. Старт и стоп по нажатию кнопки.	2
Практика 2. Тестирование интегральных схем.	Разработка тестовых стендов на основе SystemVerilog для сумматора и вычитателя. Проверка корректности сумматора и вычитания в ModelSim и Verilator.	2

Продолжение на следующей странице

Таблица 1 – продолжение

Тема	**Содержание**	**Часы**
Лекция 3. Конечные автоматы в интегральных схемах.	1. Конечные автоматы. Определение и примеры. 2. Примеры реализации конечных автоматов Мура и Мили. 3. Реализация тестов проверки конечного автомата в Verilator и ModelSim. 4. Визуальный редактор конечных автоматов в Quartus. Пример проекта: – Реализация конечного автомата определения состояния кнопки (отпущена, нажата, удерживается).	2
Практика 3. Реализация конечного автомата управления светофорами.	Реализация светофора на автомобильной дороге. Предполагается, что необходимо разработать устройство управления светофором на пешеходном переходе. Светофоры переключают цвет по таймеру. При необходимости пешеход может переключить светофор по нажатию кнопки.	2
Лекция 4. Методы работы со статической памятью.	1. Реализация ROM и SRAM на SystemVerilog. 2. Чтение данных по адресу. Определение режима чтения/записи по регистру we. 3. Начальная инициализация памяти с использованием readmemb, readmemh. 4. Пример использования IP-блока однопортовой памяти. 5. Пример использования визуального редактора схем Quartus. Пример проекта: – Копирование данных из ROM в SRAM.	2
Практика 4. Вычисление арифметических выражений.	Реализация вычисления арифметических выражений в обратной польской записи. Предполагается, что выражения изначально хранятся в ROM. Для вычислений используется стек на основе SRAM. Для успешной сдачи проекта необходимо предусмотреть обработку ошибок и сигнализацию о состоянии светодиодами.	2
Лекция 5. Реализация взаимодействия FPGA и PC на основе протокола UART.	1. Основы протокола UART. 2. Реализация передачи данных через пользовательский модуль и IP блок. 3. Реализация Python скрипта чтения и записи данных на FPGA. Пример проекта: – Реализация отладочного модуля чтения/записи данных на FPGA по адресу SRAM.	2

Продолжение на следующей странице

Таблица 1 – продолжение

Тема	**Содержание**	**Часы**
Практика 5. Вычисление арифметических выражений, передаваемых с PC.	Реализация вычисления выражений в обратной польской записи. Практика является прямым продолжением практики 4. В дополнении к практике 4 польская запись арифметического выражения передается на интегральную схему через UART протокол.	2
Лекция 6. Методы построения трансляторов.	1. Основные этапы построения трансляторов. 2. Лексический и синтаксический анализ. 3. Грамматики. 4. Дерево разбора. 5. Алгоритмы построения дерева разбора. Нисходящий и восходящий разбор. 6. Метод построения транслятора с использованием ANTLR.	2
Практика 6. Построение транслятора арифметических выражений для передачи данных на интегральную схему.	Реализация вычисления выражений в обратной польской записи. Практика является прямым продолжением практики 5. В дополнении к практике 5 реализуется транслятор арифметических выражений в обратную польскую запись в машинных кодах на основе ANTLR. Полученный набор машинных инструкций передается на интегральную схему через UART протокол.	2

2.2. Теоретическая часть

Лекционный материал курса построен на практико-ориентированном подходе. Каждая лекция включает теоретическую основу, необходимую для понимания решаемой задачи, после чего следует практическая демонстрация реализации рассматриваемого подхода на конкретном примере. Рассмотрим более детально теоретическую часть курса.

Лекция 1 дает общее понимание того, какие задачи решаются с использованием программируемых логических интегральных схем, и почему этот класс устройств занимает промежуточное положение между специализированными ASIC и программно-аппаратными решениями на базе CPU. В материалах лекции рассматриваются принципы работы NMOS и PMOS транзисторов, на основе которых строятся базовые логические операции. Показывается как осуществляется переход от электрических сигналов к логическим элементам, являющихся основой любых вычислений. Также вводятся основные понятия и синтаксис SystemVerilog для описания комбинационной и последовательностной логики. В заключительной части лекции демонстрируются основы работы в среде Quartus: создание проекта, реализация простейшего модуля управления светодиодами, компиляция, анализ RTL-схемы, назначение выводов в Pin Planner и запуск проекта на отладочной плате Cyclone V GX Starter Kit.

Лекция 2 посвящена тестированию интегральных схем. Рассматриваются методы реализации тестовых стендов на языке SystemVerilog и C++. Для запуска тестовых стендов SystemVerilog используется ModelSim – инструмент, обеспечивающий высо-

кую точность симуляции и широкий набор средств отладки. Также рассматривается инструмент с открытым исходным кодом Verilator, который позволяет генерировать проект на C++, компилировать его и исполнять. Ключевое отличие Verilator заключается в том, что тестовые стенды реализуются на C++, что обеспечивает очень высокую производительность симуляции. Демонстрация двух инструментов в рамках одной лекции позволяет показать основные подходы к тестированию: ModelSim используется там, где требуется детальная отладка и точность, а Verilator – когда на первый план выходит скорость тестирования.

Лекция 3 студентам напоминает понятия, связанные с конечными автоматами. Рассматриваются примеры реализации автоматов Мура и Мили, а также способы их тестирования в Verilator и ModelSim. Демонстрируется пример реализации конечного автомата для определения состояния кнопки (отпущена, нажата, удерживается). Описание конечных автоматов выполняется как вручную с использованием SystemVerilog, так и с использованием визуального редактора встроенного в Quartus.

Лекция 4 посвящена методам работы со статической памятью. Рассматриваются реализация ROM и SRAM на SystemVerilog, принципы чтения данных по адресу и определение режима чтения/записи с помощью регистра we. Описываются способы начальной инициализации памяти, а также пример использования IP-блока однопортовой памяти. Рассмотренный в лекции материал позволяет разрабатывать вычислительные устройства, в которых общий алгоритм работы задается в ROM памяти, а для непосредственного вычисления используется SRAM память.

Лекция 5 рассматривается реализация взаимодействия между FPGA и персональным компьютером на основе протокола UART. Описываются основы протокола, способы реализации передачи данных через пользовательский модуль и готовый IP-блок. Также в лекции демонстрируется метод разработки Python скрипта для чтения и записи данных на FPGA. Лекция позволяет заменить ROM память из лекции 4 на SRAM и сделать тем самым устройство более гибким за счет возможности смены вычислительной программы.

Лекция 6 завершает теоретическую часть курса описанием методов построения трансляторов. Рассматриваются основы построения трансляторов, понятия грамматик и дерева разбора, приводятся описания и примеры реализации алгоритмов нисходящего и восходящего разбора. Для снижения трудоемкости разработки транслятора демонстрируется использование инструмента ANTLR, автоматизирующего генерацию лексических и синтаксических анализаторов на основе формального описания грамматики.

3. Практическая часть

Практическая часть курса предполагает выполнение серии заданий, которые поддерживают лекционный материал. Рассмотрим более детально практическую часть курса.

Практика 1 посвящена реализации сумматора и вычитателя на основе логических операций, а также выполнению вычислений с использованием комбинационной и последовательностной логики. Практика дает основы понимания построения арифметико-логических устройств (АЛУ).

Практика 2 посвящена разработке тестовых стендов для разработанного АЛУ. Корректность проверяется в средах ModelSim и Verilator.

Практика 3 ориентирована на реализацию конечного автомата для управления светофором на автомобильной дороге, где переключение сигналов происходит как

по таймеру, так и по кнопке, нажатой пешеходом. Практика является переходной от простых устройств и формирует понимание того, как разрабатываются многотактовые вычислительные устройства.

Практики 4-6 охватывают несколько последовательных этапов разработки вычислительного устройства на основе стековой архитектуры. На первом этапе (**практика 4**) студенты реализуют базовую логику стековых вычислений и взаимодействие с АЛУ, при этом сами выражения для вычислений заранее хранятся в ROM памяти. Это позволяет сосредоточиться на отработке внутреннего устройства стекового процессора без усложнения внешними интерфейсами. На втором этапе (**практика 5**) статическая память заменяется на SRAM, а также добавляется возможность записи данных в память через протокол UART, что позволяет загружать выражения с персонального компьютера. На финальном этапе (**практика 6**) вычисляемые выражения задаются на естественном языке в привычной инфиксной записи. Для их обработки разрабатывается транслятор на основе ANTLR, преобразующий математическое выражение в последовательность машинных кодов, которые затем передаются на созданное ранее вычислительное устройство через UART.

Таким образом, в ходе выполнения практических работ формируется целостное понимание всех основных этапов разработки собственного вычислительного устройства: от проектирования архитектуры и реализации базовых блоков до создания инструментального программного обеспечения и организации канала связи между персональным компьютером и FPGA.

Заключение

FPGA – востребованная область при проектировании электронных устройств. Поэтому выпускники ИТ-специальностей должны иметь минимальный кругозор в данной области, обеспечивающий быстрый старт в решении реальных прикладных задач. Курс «Программирование для FPGA» направлен на то, чтобы предоставить необходимые для этого базовые знания и навыки.

К моменту подачи работы подготовлены основные материалы для реализации курса в виде набора кодов, реализующих примеры из лекций. Также полностью готовы лекции 1, 2 и 3 согласно плану курса (таблица 1). Наряду с этим, выполнена необходимая подготовительная работа для последующей разработки материалов практической части курса. К началу сентября планируется полностью разработать презентации лекционной части курса и описания практических заданий. Все материалы предполагается выложить в открытый доступ на сайте университета.

Разрабатываемый курс планируется читать в качестве спецкурса в основной программе обучения студентов четвертого курса бакалавриата Института информационных технологий, математики и механики ННГУ им. Н.И. Лобачевского, специализирующихся по направлению «Фундаментальная информатика и информационные технологии». Отдельные материалы можно задействовать в ходе проведения молодежных школ для студентов и аспирантов.

Один из возможных путей развития курса – усиление лекционной и практической составляющих за счет расширения методов взаимодействия с широким спектром периферийных устройств. В частности, возможно рассмотреть методы подключения и обработку данных с датчиков (например, акселерометров, гироскопов, температурных сенсоров) и микрофонов, организацию доступа и обмена данными с динамической памятью, а также реализацию контроллеров, обеспечивающих видеовыход с использованием интерфейса HDMI. Такое дополнение позволит приблизить содержание курса к реальным задачам разработки встраиваемых систем на FPGA.

Список литературы

[1] Бруно Ф. Программирование FPGA для начинающих // Москва: ДМК Пресс, 2022. – 304 с.

[2] Харрис С.Л., Харрис Д. Цифровая схемотехника и архитектура компьютера: RISC-V // Москва: ДМК Пресс, 2022. – 810 с.

[3] Samayoa W.F., Crespo M.L., Cicuttin A., Carrato S. A Survey on FPGA-Based Heterogeneous Clusters Architectures // IEEE Access, 2023. – Vol. 11. P. 67679–67706.

[4] Tessier R., Burleson W. Reconfigurable Computing for Digital Signal Processing: A Survey // The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology, 2001. Vol. – 28. P. 7–27.

[5] Boutros A., Arora A., Betz V. Field-Programmable Gate Array Architecture for Deep Learning: Survey and Future Directions // Proceedings of the IEEE, 2025. Vol. – 113. № – 7. P. 613-639.

[6] Курс «Цифровая электроника на ПЛИС», МГУ. URL: https://distant.msu.ru/enrol/index.php?id=2986 (дата обращения 02.04.2026).

[7] Курс «Введение в FPGA и Verilog», ФРКТ при поддержке Фонда целевого капитала МФТИ. URL: https://miptstream.ru/2019/02/14/drec-courses/ (дата обращения 02.04.2026).

[8] Курс «Углубленное программирование на HDL», НИУ ВШЭ. URL: https://www.hse.ru/ma/edgeai/courses/1048797272.html (дата обращения 02.04.2026).

[9] Курс «Языки описания аппаратуры», НГУ. URL: https://www.nsu.ru/n/physics-department/programmy/Магистратура-Информационные-процессы-и-системы-Спец курсы/АФТИ/1_1_ЯзыкиОписанияАппарара.pdf (дата обращения 02.04.2026).

[10] Курс «Школа синтеза цифровых схем», YADRO. URL: https://engineer.yadro.com/chip-design-school/ (дата обращения 02.04.2026).

[11] Библиотека образовательных материалов, Alliance RISC-V. URL: https://riscv-alliance.ru/library/?library_section_ids=73&discipline=25 (дата обращения 02.04.2026).

[12] Sutherland S., Davidmann S., Flake P. SystemVerilog for Design: A Guide to Using SystemVerilog for Hardware Design and Modeling // New York: Springer, 2006. – 418 p.

[13] Quartus. URL: https://www.altera.com/products/development-tools/quartus (дата обращения 02.04.2026).

[14] ModelSim. URL: https://www.altera.com/downloads/simulation-tools/modelsim-fpga-standard-edition-software-version-20-1-1 (дата обращения 02.04.2026).

[15] Verilator. URL: https://www.veripool.org/verilator/ (дата обращения 02.04.2026).

[16] Parr T. The Definitive ANTLR 4 Reference // Raleigh: The Pragmatic Bookshelf, 2012. – 328 p.

Учебный курс «Программирование для графических процессоров»*

А.В. Горшков

Нижегородский государственный университет им. Н.И. Лобачевского

Современные графические процессоры широко применяются при решении задач научного моделирования и обработки данных, а также при разработке современных систем искусственного интеллекта. В связи с этим освоение архитектуры графических процессоров и методов программирования для GPU является важной частью подготовки специалистов в области информационных технологий. В лекционной части курса «Программирование для графических процессоров» рассматриваются архитектура современных GPU, модели параллельного программирования CUDA и OpenCL, методы оптимизации производительности вычислительных приложений, а также дается обзор специализированных библиотек CUDA. Практическая часть курса включает реализацию вычислительных алгоритмов на CPU и GPU, последовательную оптимизацию программ и использование специализированных библиотек для ускорения вычислений. Особенностью курса является автоматизированная система приема практических работ на базе GitHub репозитория, позволяющая проверять корректность реализации алгоритмов и оценивать производительность решений.

Ключевые слова: учебный курс, графические процессоры, CUDA, OpenCL, высокопроизводительные вычисления

1. Введение

Графические процессоры (GPU) [1, 2] в последние годы стали одним из ключевых инструментов в области высокопроизводительных вычислений (High Performance Computing, HPC). Благодаря массовому параллелизму и высокой пропускной способности памяти графические ускорители позволяют эффективно решать широкий спектр вычислительно трудоемких задач, возникающих в научном моделировании, обработке сигналов и изображений, численных методах, а также в системах искусственного интеллекта (ИИ) и машинного обучения. По данным рейтинга суперкомпьютеров TOP500 [3], значительная часть наиболее производительных вычислительных систем мира использует GPU-ускорители в качестве основных вычислительных устройств, что подтверждает их ключевую роль в развитии современных HPC-технологий.

Активное использование графических ускорителей в суперкомпьютерных центрах, научных исследованиях и индустриальных вычислительных системах (в том числе в области ИИ) приводит к росту спроса на специалистов, владеющих технологиями программирования GPU. В связи с этим подготовка инженеров, обладающих практическими навыками разработки и оптимизации программ для графических процессоров, является актуальной задачей образовательных программ по информационным технологиям.

Целью учебного курса «Программирование для графических процессоров» является формирование у студентов базовых знаний об архитектуре современных GPU

*Работа выполнена при поддержке Аналитического Центра при Правительстве Российской Федерации (договор No 70-2025-000821 от 04.06.2025).

и методов параллельного программирования, а также развитие практических навыков разработки и оптимизации вычислительных приложений с использованием технологий CUDA и OpenCL. Особое внимание в курсе уделяется практической работе студентов, направленной на освоение принципов разработки эффективных параллельных алгоритмов и анализа их производительности.

Статья построена следующим образом. В разделе 2 приводится обзор существующих учебных курсов по программированию для графических процессоров. В разделе 3 описывается структура курса и содержание его теоретической и практической частей. В разделе 4 рассматриваются результаты реализации курса и перспективы его развития.

2. Обзор существующих курсов

В настоящее время в ряде российских университетов реализуются учебные дисциплины, посвященные программированию графических процессоров в задачах высокопроизводительных вычислений. Ниже приведен краткий обзор некоторых из таких курсов.

В Московском государственном университете имени М.В. Ломоносова (МГУ) читается курс «Массивно-параллельное программирование на CUDA» [4], посвященный изучению архитектуры графических процессоров и программной модели CUDA. Данный курс является одним из наиболее содержательных среди рассматриваемых аналогов: он включает большое количество тем и сопровождается значительным числом заданий. Практическая часть курса ориентирована на решение прикладных вычислительных задач в области физического моделирования, включая моделирование системы N-тел, решение СЛАУ, трассировку лучей, метод Монте-Карло и др.

Курс «Программирование на CUDA» [5], предлагаемый Национальным исследовательским университетом «Высшая школа экономики» (ВШЭ), ориентирован на разработку вычислительных приложений для графических процессоров в контексте решения задач искусственного интеллекта. Курс охватывает архитектуру GPU, модель исполнения CUDA-программ и методы оптимизации вычислений. Практическая часть включает разработку и оптимизацию вычислительных ядер, а также использование библиотек CUDA для решения задач ИИ.

В Южном федеральном университете (ЮФУ) реализуется курс «Программирование ускорителей параллельных вычислений» [6]. В рамках курса изучаются архитектура графических ускорителей, модели параллельного программирования и методы оптимизации алгоритмов. В дополнение к CUDA, рассматриваются технологии OpenCL, OpenACC, OpenMP и OpenCV. Практические задания направлены на реализацию базовых параллельных алгоритмов, анализ эффективности GPU-вычислений и работу с современными тензорными ядрами NVIDIA GPU.

Новосибирский государственный университет (НГУ) совместно с Институтом вычислительной математики и математической геофизики СО РАН предлагает курс «Программирование графических ускорителей (CUDA)» [7], который является достаточно сжатым, однако содержит все основные разделы GPU программирования, включая аппаратную и программную архитектуру CUDA, иерархию памяти графического процессора, и ряд лабораторных работ на закрепление практических навыков у студентов. Аналогичный курс представлен в Дальневосточном федеральном университете (ДФУ) [8].

Анализ представленных курсов показывает, что большинство из них сосредоточено на изучении архитектуры графических процессоров, технологии CUDA и методов

оптимизации параллельных алгоритмов. Различные курсы могут иметь уклон на решение практических задач в области высокопроизводительных вычислений [4], искусственного интеллекта [5, 6], либо давать базовые навыки CUDA-программирования [7, 8]. Во всех рассмотренных курсах даются практические задания для закрепления прикладных навыков у студентов, количество и сложность этих заданий могут варьироваться.

В отличие от рассмотренных аналогов, предлагаемый курс имеет ряд принципиальных особенностей. Во-первых, в нём внедрена полностью автоматизированная система приёма и проверки лабораторных работ на базе GitHub. Во-вторых, итоговая оценка формируется не только на основе корректности реализации, но и на основе её производительности (времени работы алгоритма), что стимулирует студентов к поиску эффективных решений. В-третьих, в курсе используется рейтинговая система, учитывающая как порядок сдачи работ, так и достигнутое быстродействие, что создаёт здоровую конкуренцию и дополнительную мотивацию. Наконец, практические задания организованы так, чтобы обеспечить сквозное сравнение различных технологий на одних и тех же алгоритмах (например, реализация GELU или умножения матриц последовательно выполняется с использованием OpenMP на CPU, CUDA на GPU и OpenCL на GPU). Это позволяет студентам на количественных примерах осознать преимущества и ограничения каждого подхода, что отсутствует в большинстве существующих курсов.

3. Структура курса

3.1. Описание курса

Учебный курс «Программирование для графических процессоров» направлен на формирование у студентов базовых знаний и практических навыков разработки высокопроизводительных приложений для GPU. Курс ориентирован на студентов старших курсов бакалавриата и магистратуры ИТ-направлений и посвящен изучению современных технологий программирования графических процессоров, прежде всего CUDA и OpenCL.

В лекционной части курса рассматриваются основные принципы гетерогенных вычислений и архитектурные особенности современных графических процессоров. Значительное внимание уделяется архитектуре NVIDIA GPU, эволюции вычислительных архитектур и особенностям организации параллельных вычислений. Также рассматриваются вопросы иерархии памяти графических процессоров и эффективного взаимодействия между центральным процессором и GPU.

Отдельный раздел курса посвящен технологии CUDA. В рамках данного раздела изучаются методы взаимодействия программы-хоста с графическим процессором, особенности использования CUDA Host API, а также принципы разработки и запуска вычислительных ядер. Рассматриваются вопросы организации потоков и блоков потоков, использование различных типов памяти GPU и методы оптимизации вычислительных приложений. Особое внимание уделяется оптимальным шаблонам доступа к глобальной и разделяемой памяти.

Помимо базовых механизмов программирования GPU, в курсе рассматриваются специализированные библиотеки для высокопроизводительных вычислений, включая CUBLAS, CUFFT, CURAND и CUSPARSE. Также дается обзор технологии OpenCL как альтернативного подхода к программированию графических ускорителей.

Практическая часть курса включает выполнение лабораторных работ, направленных на закрепление теоретического материала и формирование навыков разработки

параллельных алгоритмов для GPU. В рамках обновления курса в 2025–2026 учебном году практическая часть была существенно расширена: количество лабораторных работ увеличено с пяти до девяти, а также внедрена система автоматизированной проверки решений студентов.

Практические задания охватывают широкий круг задач высокопроизводительных вычислений, включая реализацию нейросетевых операций, таких как функция активации GELU, различные варианты умножения матриц, использование оптимизированных библиотек GPU-вычислений и реализацию алгоритмов быстрого преобразования Фурье. Это позволяет студентам на практике изучить как базовые методы параллельного программирования, так и методы оптимизации вычислительных алгоритмов.

Таким образом, курс сочетает теоретическое изучение архитектуры графических процессоров и технологий программирования GPU с практической работой, направленной на разработку и оптимизацию вычислительных приложений. Это позволяет сформировать у студентов комплексное представление о современных методах высокопроизводительных вычислений на базе графических ускорителей.

3.2. Теоретическая часть

Лекционная часть курса направлена на формирование у студентов понимания принципов организации вычислений на GPU, особенностей архитектуры графических процессоров и методов разработки эффективных параллельных алгоритмов.

Лекция 1 посвящена введению в гетерогенные вычисления и основным предпосылкам использования графических ускорителей. В рамках лекции рассматриваются уровни параллелизма в современных вычислительных системах, области применения GPU и причины их высокой эффективности при решении задач высокопроизводительных вычислений. Обсуждаются основные различия архитектур CPU и GPU, а также роль графических процессоров в современных суперкомпьютерах и задачах искусственного интеллекта.

Лекция 2 содержит обзор архитектуры графических процессоров NVIDIA. Рассматриваются базовые принципы организации вычислительных блоков GPU, иерархия памяти и эволюция архитектур NVIDIA. Также вводится понятие вычислительной возможности (Compute Capability), определяющей поддерживаемые аппаратные возможности различных поколений GPU.

Таблица 1. Структура лекционной части курса «Программирование для графических процессоров».

Тема	**Содержание**	**Часы**
Лекция 1. Введение в вычисления общего назначения на GPU.	1. Гетерогенные вычисления, уровни параллелизма. 2. Вычисления общего назначения на GPU – ускорители, применение, мотивация. 3. Почему GPU? Производительность, память, суперкомпьютеры, ИИ. 4. Современные GPU – особенности архитектуры, отличия от CPU. 5. Технологии программирования GPU.	2

Продолжение на следующей странице

Таблица 1 – продолжение

Тема	**Содержание**	**Часы**
Лекция 2. Введение в архитектуру NVIDIA GPU. Обзор современных архитектур.	1. Архитектура GPU – основные принципы. 2. Иерархия памяти. 3. Эволюция современных архитектур NVIDIA GPU. 4. Особенности современных архитектур. 5. CUDA Compute Capability.	2
Лекция 3. Модель выполнения программ на GPU. Язык CUDA C++.	1. Модель выполнения программ на GPU – потоки, блоки, пространство исполнения, SIMT. 2. Язык CUDA C++. Ядра и их запуск. Типы функций.	2
Лекция 4. Модель памяти GPU. Управление GPU со стороны хоста. CUDA Host API.	1. Модель и типы памяти на GPU. 2. CUDA API – управление устройствами, памятью, процессом исполнения, обработка ошибок. 3. Пример: CUDA “Hello World”.	2
Лекция 5. Асинхронное исполнение задач на GPU. Общая память.	1. Очереди и события. 2. Замеры времени исполнения CUDA ядер. 3. Общая память для GPU и CPU (Unified Memory).	2
Лекция 6. Оптимизация приложений на CUDA.	1. Общие рекомендации. 2. Работа с памятью – глобальная и разделяемая память. 3. Параллельная редукция – пошаговая оптимизация.	2
Лекция 7. Использование стандартных библиотек CUDA.	1. CUBLAS – операции линейной алгебры. 2. CUFFT – быстрое преобразование Фурье. 3. CURAND – генераторы псевдослучайных чисел. 4. CUSPARSE – операции разреженной алгебры.	2
Лекция 8. Программирование с использованием технологии OpenCL.	1. Стандарт гетерогенных вычислений OpenCL. 2. Модель платформы. 3. Модель памяти. 4. Модель исполнения. 5. Модель программирования. 6. Пример приложения с использованием OpenCL.	2

Лекция 3 посвящена модели выполнения программ на GPU и основам языка CUDA C++. Рассматриваются принципы организации параллельных вычислений в модели SIMT (Single-Instruction Multiple-Threads), структура вычислительных сеток (grid), блоков потоков (block) и отдельных потоков (thread). Также рассматриваются основные типы функций CUDA и механизмы запуска вычислительных ядер.

В лекции 4 подробно рассматривается модель памяти графического процессора и механизмы взаимодействия между центральным процессором и GPU. Изучаются различные типы памяти GPU и их особенности, а также средства управления памятью и устройствами с использованием CUDA Host API. В качестве иллюстрации приводится пример простого приложения CUDA.

Лекция 5 посвящена вопросам асинхронного выполнения вычислений на GPU. В рамках лекции рассматриваются очереди выполнения и события, методы измерения

времени выполнения CUDA-ядер, а также механизмы совместного использования памяти CPU и GPU с применением технологии Unified Memory.

Лекция 6 рассматривает методы оптимизации вычислительных приложений на CUDA. Обсуждаются общие рекомендации по повышению эффективности GPU-программ, особенности работы с глобальной и разделяемой памятью, а также методы пошаговой оптимизации параллельных алгоритмов на примере задачи параллельной редукции.

Лекция 7 посвящена использованию стандартных библиотек CUDA для высокопроизводительных вычислений. Рассматриваются возможности библиотек CUBLAS для операций линейной алгебры, CUFFT для быстрого преобразования Фурье, CURAND для генерации псевдослучайных чисел и CUSPARSE для работы с разреженными матрицами.

Лекция 8 знакомит студентов с технологией OpenCL как альтернативным стандартом программирования гетерогенных вычислительных систем. В лекции рассматриваются основные компоненты модели OpenCL, включая модель платформы, модель памяти, модель исполнения и модель программирования. Также приводится пример разработки приложения (возведение в квадрат массива чисел) с использованием OpenCL.

Следует отметить, что модель программирования OpenCL во многом повторяет идеи, заложенные в CUDA, отличаясь главным образом терминологией и деталями API. Поэтому, при условии полноценного усвоения предыдущих лекций по CUDA, для формирования у студентов практических навыков работы с OpenCL достаточно одной лекции, которая фокусируется на ключевых различиях, и одной лабораторной работы.

3.3. Практическая часть

Практическая часть курса направлена на формирование у студентов навыков разработки и оптимизации параллельных приложений для графических процессоров. В отличие от лекционной части, где рассматриваются теоретические основы архитектуры GPU и технологий программирования, практические занятия ориентированы на решение конкретных вычислительных задач с использованием технологий CUDA, OpenMP и OpenCL.

Таблица 2. Структура практической части курса «Программирование для графических процессоров».

Тема	**Часы**
Практика 1. Реализация алгоритма GELU на центральном процессоре с использованием технологии OpenMP.	2
Практика 2. Реализация алгоритма GELU на графическом процессоре с использованием технологии CUDA.	2
Практика 3. Реализация наивного алгоритма матричного умножения на центральном процессоре с использованием технологии OpenMP.	2

Продолжение на следующей странице

Таблица 2 – продолжение

Тема	**Часы**
Практика 4. Реализация наивного алгоритма матричного умножения на графическом процессоре с использованием технологии CUDA.	2
Практика 5. Реализация блочного алгоритма матричного умножения на центральном процессоре с использованием технологии OpenMP.	2
Практика 6. Реализация блочного алгоритма матричного умножения на графическом процессоре с использованием технологии CUDA.	2
Практика 7. Реализация алгоритма матричного умножения на графическом процессоре с использованием библиотеки CUBLAS.	2
Практика 8. Реализация алгоритма быстрого преобразования Фурье на графическом процессоре с использованием библиотеки CUFFT.	2
Практика 9. Реализация алгоритма GELU на графическом процессоре с использованием технологии OpenCL.	2

В рамках обновления курса в 2025–2026 учебном году практическая часть была существенно переработана. Основными целями обновления являлись расширение объема практических заданий и автоматизация процесса проверки лабораторных работ. В результате количество лабораторных работ было увеличено с пяти до девяти, а также была разработана автоматизированная система приема и проверки решений студентов на базе платформы GitHub.

Практические задания охватывают ключевые темы курса и направлены на изучение различных аспектов параллельного программирования. В частности, студентам предлагается реализовать вычислительные алгоритмы как на центральном процессоре с использованием технологии OpenMP, так и на графическом процессоре с использованием CUDA. Это позволяет наглядно сравнить эффективность различных подходов к параллельным вычислениям.

Первая группа лабораторных работ (**практики 1, 2 и 9**) посвящена реализации функции активации GELU, широко используемой в современных нейросетевых моделях. Студенты реализуют данный алгоритм сначала на CPU с использованием OpenMP, а затем на GPU с применением CUDA и OpenCL. Такой подход позволяет продемонстрировать преимущества использования графических ускорителей при выполнении вычислительно интенсивных операций, характерных для задач искусственного интеллекта.

Вторая группа лабораторных работ (**практики 3-7**) предлагает реализацию алгоритмов умножения матриц. Студенты последовательно реализуют наивный алгоритм умножения матриц на CPU и GPU, а затем изучают более эффективный блочный алгоритм. Кроме того, рассматривается использование специализированной библиотеки CUBLAS для выполнения операций линейной алгебры на графическом процессоре. Такая последовательность заданий позволяет проследить эволюцию методов оптимизации вычислений и познакомиться с практическими подходами к повышению производительности GPU-приложений.

Отдельная лабораторная работа (**практика 8**) основана на использовании библиотеки CUFFT для реализации алгоритма быстрого преобразования Фурье. Это

позволяет студентам познакомиться с применением специализированных библиотек GPU-вычислений для решения типовых задач высокопроизводительных вычислений.

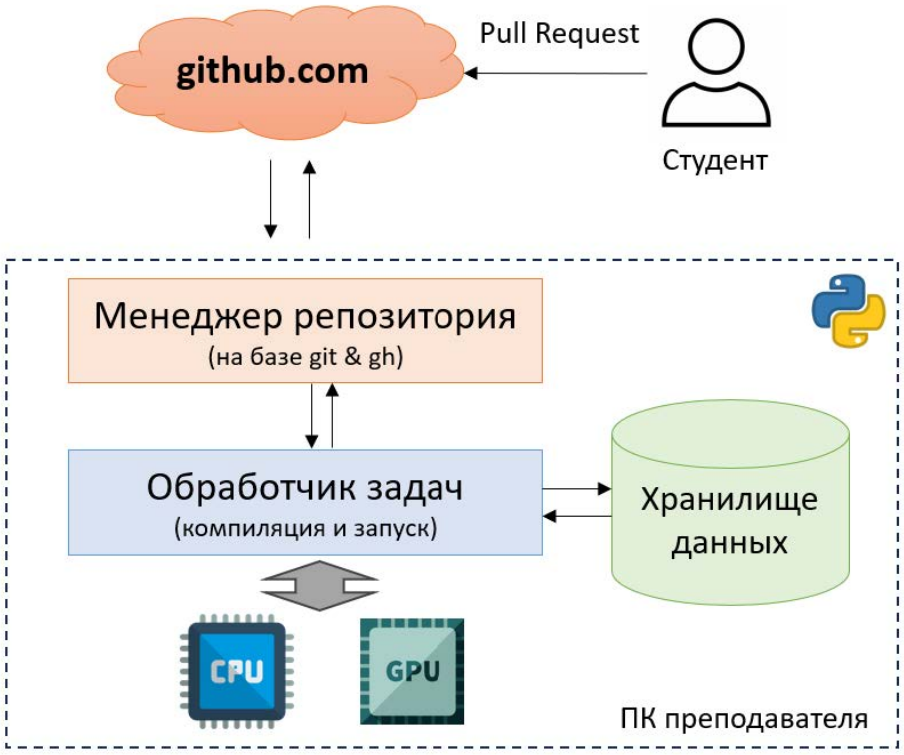


Рис. 1. Архитектура системы автоматизированной проверки лабораторных работ.

Для упрощения процесса приема лабораторных работ и повышения прозрачности оценки была разработана автоматизированная система проверки решений студентов. В рамках данной системы студенты размещают свои решения в виде pull request в общем репозитории курса на платформе GitHub. Система автоматически выполняет компиляцию программ, запуск тестов на корректность работы и измерение времени исполнения. В случае успешного выполнения результаты публикуются в репозитории курса.

Дополнительно в курсе используется рейтинговая система оценки результатов студентов. Итоговый рейтинг формируется на основе двух показателей: порядка сдачи лабораторных работ и производительности реализованных алгоритмов. Оба показателя порядковые: например, в группе из 20 человек, студент, сдавший практическое задание первым, получит 20 баллов, последним – 1 балл. Практическое задание считается сданным, если задача отработала корректно на всех тестах. Аналогично с производительностью – самое быстрое решение получит 20 баллов, самое медленное – 1 балл. Финальный балл рассчитывается как сумма баллов за порядок сдачи и за производительность. Например, если студент сдал работу вторым и был пятым в списке по производительности, он получит 35 баллов. Такой подход стимулирует студентов не только корректно реализовывать алгоритмы, но и уделять внимание их оптимизации и эффективности.

Таким образом, практическая часть курса ориентирована на решение реальных вычислительных задач и позволяет студентам получить опыт разработки и оптимизации высокопроизводительных приложений для графических процессоров. Использование автоматизированной системы проверки и рейтинговой системы оценки повышает мотивацию студентов и делает процесс обучения более прозрачным и конкурентным.

Заключение

Программирование графических процессоров – востребованный в индустрии и научном сообществе навык, позволяющий эффективно решать прикладные задачи в области высокопроизводительных вычислений и искусственного интеллекта. Обладание этим навыком может существенно увеличить привлекательность выпускников ИТ-специальностей в глазах будущих работодателей, а наличие хотя бы минимального кругозора в этой области представляется необходимым в современном мире активно развивающегося ИИ.

Представленный в данной работе курс «Программирование для графических процессоров» разработан для обучения студентов четвертого курса бакалавриата Института информационных технологий, математики и механики ННГУ им. Н.И. Лобачевского, специализирующихся по направлениям «Фундаментальная информатика и информационные технологии» и «Программная инженерия». Курс встроен в качестве спецкурса в основную программу обучения. Отдельные материалы могут быть задействованы в ходе проведения молодежных школ для студентов и аспирантов, а также обучения сотрудников ИТ-компаний в индустрии.

Особенностью курса является автоматизированная система приема и проверки практических работ на базе GitHub, которая позволяет оценивать как корректность реализации, так и производительность решений. Дополнительно применяется рейтинговая система, стимулирующая студентов к поиску более эффективных подходов и оптимизации кода.

В рамках дальнейшего развития курса предполагается его расширение в сторону искусственного интеллекта. Планируется дополнить лекционную часть модулями, посвященными применению тензорных ядер (WMMA API) и библиотек для работы с ними (cuBLASLt, CUTLASS), оптимизации ИИ примитивов, а также обзору библиотек cuDNN и TensorRT для ускорения разработки моделей глубокого обучения. В практической части предполагается внедрение лабораторных работ по реализации базовых операций глубокого обучения (Softmax, LayerNorm) с использованием тензорных ядер, а также по ускорению нейросетевых моделей на GPU. Это позволит студентам применять полученные навыки в актуальных областях ИИ, таких как компьютерное зрение и обработка естественного языка.

Список литературы

[1] Kirk D.B., Hwu W.-m.W. Programming Massively Parallel Processors: A Hands-on Approach. 4th ed. Morgan Kaufmann. 2022.

[2] Sanders J., Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley Professional. 2010.

[3] The TOP500 List of the World's Most Powerful Supercomputers. URL: https://www.top500.org (дата обращения: 30.03.2026).

[4] Московский государственный университет им. М.В. Ломоносова. Массивно-параллельное программирование на CUDA. URL: https://engineering.phys.msu.ru/ru/programmirovanie-dlya-studentov-2-kursa/programmirovanie-na-graficheskikh-protsessorakh (дата обращения: 30.03.2026).

[5] Национальный исследовательский университет «Высшая школа экономики». Программирование на CUDA. URL: https://www.hse.ru/edu/courses/1112506250 (дата обращения: 30.03.2026).

[6] Южный федеральный университет. Программирование ускорителей параллельных вычислений. URL: https://edu.mmcs.sfedu.ru/course/view.php?id=225 (дата обращения: 30.03.2026).
[7] Новосибирский государственный университет / ИВМиМГ СО РАН. Программирование графических ускорителей (CUDA). URL: https://ssd.sscc.ru/ru/content/fit-cuda (дата обращения: 30.03.2026).
[8] Дальневосточный федеральный университет. Введение в программирование на CUDA. URL: https://cc.dvfu.ru/ru/category/library/cuda_introduction/ (дата обращения: 30.03.2026).

Численное исследование турбулентного атмосферного погра-ничного слоя с помощью мезомасштабной модели TSUNM3 высокого разрешения*

А.В. Старченко[1], А.И. Сваровский[1]

[1]Томский государственный университет, Томск

Предлагается оригинальный метод расчета турбулентных параметров в атмосферном пограничном слое в «серой» зоне моделирования турбулентности с использованием нестационарной трехмерной трехпараметрической модели турбулентности и алгебраических соотношений для напряжений Рейнольдса и турбулентных потоков тепла. Мезомасштабная модель с такой параметризацией турбулентности в пограничном слое с горизонтальным разрешением 0,333 км прошла тестирование на измерениях, выполненных с помощью метеоприборов Базового экспериментального комплекса ИОА СО РАН. Получены результаты динамики вертикального распределения температуры в течение суток для условий г. Томск.

Ключевые слова: численный прогноз погоды, мезомасштабная модель TSUNM3 с высоким разрешением, турбулентная структура, атмосферный пограничный слой

1. Введение

Математическое моделирование атмосферных процессов с горизонтальным разрешением от нескольких десятков метров до нескольких километров необходимо для понимания результатов численного прогнозирования и управления погодными, климатическими и экологическими воздействиями на местном и региональном уровнях. Это касается, например, прогнозирования погоды с высоким разрешением, предупреждения локальных заморозков на почве, оценки качества воздуха в городах и рассеивания загрязнения воздуха на уровне улиц, оценки ветровых воздействий на здания и городскую инфраструктуру, развития лесных и городских пожаров и задымлений, использования возобновляемых источников энергии, а также снижения риска техногенных аварий (выбросы химических/биологических веществ) [1].

В настоящее время производительность современных суперкомпьютеров позволяет проводить предсказательные расчеты мезомасштабных атмосферных процессов, включая процессы в турбулентном пограничном слое, с гораздо более высоким разрешением, чем это было возможно несколько десятилетий назад. Уменьшение горизонтального и вертикального размера вычислительной сетки при мезомасштабном прогнозе погоды дает возможность многие атмосферные процессы учитывать явно, без параметризации, что, безусловно, повышает качество численного прогноза погоды. Однако переход к моделированию атмосферных процессов с горизонтальным шагом сетки от 100 м до 1 км (что сопоставимо с доминирующими масштабами длины турбулентности в пограничном слое атмосферы) приводит к проблеме «серой» зоны моделирования турбулентной структуры атмосферного пограничного слоя, когда ни методы вихреразрешающего моделирования атмосферы (разрешение в несколько

*Работа выполнена при поддержке Министерства науки и высшего образования РФ (соглашение № 075-02-2026-1339).

десятков метров), ни методы традиционных метеорологических моделей (разрешение в несколько километров) не подходят, поскольку нарушаются фундаментальные предположения, лежащие в основе их параметризации турбулентности [1, 2].

В [3] разработали и внедрили в известную модель численного прогноза погоды Weather Research & Forecasting (WRF) 3D-параметризацию АПС для численного мезомасштабного моделирования с высоким разрешением. Эта параметризация основана на ансамблевом осреднении Рейнольдса для уравнений движения и тепло-влагообмена и применении алгебраической модели турбулентности, разработанной Меллором и Ямадой [4], которая дополнительно учитывает влияние трехмерных эффектов на турбулентную структуру, явно рассчитывая турбулентные потоки импульса, тепла и влаги в дополнение к кинетической энергии турбулентности. Разработанная 3D параметризация для моделирования АПС [3] показала преимущества ее использования в конвективных условиях в «серой зоне» (0,1 км $< \Delta x <$ 1 км) моделирования турбулентности, поскольку она более точно воспроизводит эволюцию конвективных характеристик, чем традиционная 1D-схема АПС [4]. Тем не менее, авторы [3] указывают на необходимость улучшения в своей модели турбулентности параметризации масштаба турбулентности.

В связи с этим, целью данной работы является разработка и применение оригинального метода расчета турбулентных параметров АПС в мезомасштабных моделях численного прогноза погоды с высоким разрешением (0,1 км $< \Delta x <$ 1 км: $\Delta x = 0,333$ км), который опирается на ансамблевое осреднение по Рейнольдсу и трехпараметрическую модель турбулентности, включающую трехмерные нестационарные уравнения переноса для турбулентной кинетической энергии k, масштаба турбулентных пульсаций l и дисперсии турбулентных пульсаций потенциальной температуры [5]. Для оценки достоверности полученных результатов численного моделирования привлекаются данные наблюдений приборной базы Института оптики атмосферы СО РАН, выполненные в 2022 году.

2. Математическая постановка

2.1. Основные уравнения

В настоящей работе при проведении численного моделирования динамики атмосферного пограничного слоя в течение суток использовалась негидростатическая мезомасштабная модель TSUNM3 (Tomsk State University Nonhydrostatic Mesoscale Meteorological Model) [5, 6]. Основные уравнения этой модели представляются в следующем виде:

$$\frac{\partial(\rho U)}{\partial x} + \frac{\partial(\rho V)}{\partial y} + \frac{\partial(\rho W)}{\partial z} = 0. \tag{1}$$

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} + W\frac{\partial U}{\partial z} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + fV - \frac{\partial\langle\rho u\rangle}{\partial x} - \frac{\partial\langle\rho v\rangle}{\partial y} - \frac{\partial\langle\rho w\rangle}{\partial z}, \tag{2}$$

$$\frac{\partial V}{\partial t} + U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} + W\frac{\partial V}{\partial z} = -\frac{1}{\rho}\frac{\partial P}{\partial y} - fU - \frac{\partial\langle\rho u\rangle}{\partial x} - \frac{\partial\langle\rho v\rangle}{\partial y} - \frac{\partial\langle\rho w\rangle}{\partial z}, \tag{3}$$

$$\frac{\partial W}{\partial t} + U\frac{\partial W}{\partial x} + V\frac{\partial W}{\partial y} + W\frac{\partial W}{\partial z} = -\frac{1}{\rho}\frac{\partial P}{\partial z} + g\frac{\Theta - \Theta_0}{\Theta_0} - \frac{\partial\langle\rho u\rangle}{\partial x} - \frac{\partial\langle\rho v\rangle}{\partial y} - \frac{\partial\langle\rho w\rangle}{\partial z}, \tag{4}$$

$$\frac{\partial \Theta}{\partial t} + U\frac{\partial \Theta}{\partial x} + V\frac{\partial \Theta}{\partial y} + W\frac{\partial \Theta}{\partial z} = -\frac{\partial \langle \rho u \rangle}{\partial x} - \frac{\partial \langle \rho v \rangle}{\partial y} - \frac{\partial \langle \rho w \rangle}{\partial z} + \frac{1}{\rho c_p T}(Q_{rad} - \rho L \Phi), \quad (5)$$

$$P = \rho R T, \quad R = R_0 \left[\frac{1 - q_v}{M_{air}} + \frac{q_v}{M_{H_2O}} \right]. \quad (6)$$

Здесь t – время, U, V, W – продольная, поперечная и вертикальная компоненты вектора осредненной по Рейнольдсу скорости ветра в направлении декартовых координат Ox, Oy, Oz, соответственно, ρ – плотность, f – параметр Кориолиса, u, v, w – пульсации компонент скорости, g – ускорение свободного падения, P – осредненное давление, $\langle\rangle$ – операция осреднения по Рейнольдсу, T – температура, $\Theta = T(P_0/P)^{R_0/c_p}(1 + 0.061 q_V)$ – виртуальная потенциальная температура, c_p – теплоемкость воздуха при постоянном давлении, $P_0 = 101300$ Н/м^2, R_0 – газовая постоянная, Q_{rad} – нагрев (охлаждение) атмосферы за счет радиационных длинноволновых и коротковолновых потоков тепла, распространяющихся во влажной атмосфере, $\rho L \Phi$ – изменение температуры за счет фазовых переходов влаги в атмосфере, θ – пульсации температуры, q_V – удельная плотность водяного пара в атмосфере.

В качестве граничных условий для рассматриваемых дифференциальных уравнений используются следующие:

- на поверхности – соотношения теории подобия Монина-Обухова, учитывающие изменение температуры и влажности в верхнем слое почвы;
- на верхней границе атмосферы – граничные условия второго рода;
- на боковых границах области – условия «радиационного» типа Орлански [6], обеспечивающие выход вычислительных возмущений из области без отражения и учитывающие тенденции изменения фоновых значений метеорологических величин.

Для обеспечения расчетов начальными условиями и фоновыми значениями применялись данные численного прогноза погоды, полученные с помощью оперативной глобальной модели ПЛАВ Гидрометцентра РФ [7] с горизонтальным разрешением около 0,25°.

2.2. Модель турбулентности

Для турбулентного замыкания уравнений (1)–(6) используется трехпараметрическая модель турбулентности [8], в которой учитываются эффекты неоднородной турбулентной структуры атмосферы не только в вертикальном направлении, но и по горизонтали [5]:

$$\frac{\partial k}{\partial t} + U\frac{\partial k}{\partial x} + V\frac{\partial k}{\partial y} + W\frac{\partial k}{\partial z} = P_s + P_b + \frac{\partial}{\partial x}\left(\sigma_x \sqrt{k}\frac{\partial k}{\partial x}\right) + \frac{\partial}{\partial y}\left(\sigma_y \sqrt{k}\frac{\partial k}{\partial y}\right) + \\ + \frac{\partial}{\partial z}\left(\sigma_z \sqrt{k}\frac{\partial k}{\partial z}\right) - \frac{C_D k^2}{l} \quad (7)$$

$$\frac{\partial l}{\partial t} + U\frac{\partial l}{\partial x} + V\frac{\partial l}{\partial y} + W\frac{\partial l}{\partial z} = C_{l1}\left(P_s + P_b\right)\frac{l}{k} + C_{l2}\sqrt{k}\left[1 - \left(\frac{l}{\kappa z}\right)^2\right] + \\ + \frac{\partial}{\partial x}\left(\sigma_x \sqrt{k}\frac{\partial l}{\partial x}\right) + \frac{\partial}{\partial y}\left(\sigma_y \sqrt{k}\frac{\partial l}{\partial y}\right) + \frac{\partial}{\partial z}\left(\sigma_z \sqrt{k}\frac{\partial l}{\partial z}\right) \quad (8)$$

$$\frac{\partial\langle\theta^2\rangle}{\partial t}+U\frac{\partial\langle\theta^2\rangle}{\partial x}+V\frac{\partial\langle\theta^2\rangle}{\partial y}+W\frac{\partial\langle\theta^2\rangle}{\partial z}=\frac{\partial}{\partial x}\left(C_\theta\sqrt{k}\frac{\partial\langle\theta^2\rangle}{\partial x}\right)+\frac{\partial}{\partial y}\left(C_\theta\sqrt{k}\frac{\partial\langle\theta^2\rangle}{\partial y}\right)+$$
$$+\frac{\partial}{\partial z}\left(C_\theta\sqrt{k}\frac{\partial\langle\theta^2\rangle}{\partial z}\right)-2\left(\langle u\theta\rangle\frac{\partial\Theta}{\partial x}+\langle v\theta\rangle\frac{\partial\Theta}{\partial y}+\langle w\theta\rangle\frac{\partial\Theta}{\partial z}\right)-2\frac{\langle\theta^2\rangle}{C_\theta\tau} \quad (9)$$

Здесь $k=0.5(\langle u^2\rangle+\langle v^2\rangle+\langle w^2\rangle)$; P_s – генерация кинетической энергии турбулентности за счет градиента средней скорости; $P_b=g\langle w\theta\rangle/\Theta$ – генерация турбулентности k за счет плавучести; временной масштаб $\tau=l/C_D\sqrt{k}$; $\sigma_z=0.54$, $C_{l1}=-0.12$, $C_{l2}=0.2$, $C_D=0.238$, $C_\theta=3.0$ – числовые коэффициенты [5, 8, 9].

Выражения для напряжений Рейнольдса и турбулентных потоков тепла получаются из соответствующих уравнений в равновесном приближении для условий квазиоднородного АПС [5, 8, 9]:

$$\begin{aligned}
\langle u^2\rangle+\frac{2\tau}{3C_1}\left[E_1\langle uw\rangle\frac{\partial U}{\partial z}-E_2\langle vw\rangle\frac{\partial V}{\partial z}+E_3\frac{g}{\theta}\langle\theta w\rangle\right]&=\frac{2k}{3};\\
\langle v^2\rangle+\frac{2\tau}{3C_1}\left[-E_2\langle uw\rangle\frac{\partial U}{\partial z}+E_1\langle vw\rangle\frac{\partial V}{\partial z}+E_3\frac{g}{\theta}\langle\theta v\rangle\right]&=\frac{2k}{3};\\
\langle w^2\rangle-\frac{2\tau}{3C_1}\left[E_4\langle uw\rangle\frac{\partial U}{\partial z}+E_4\langle vw\rangle\frac{\partial V}{\partial z}+2E_3\frac{g}{\theta}\langle\theta w\rangle\right]&=\frac{2k}{3};\\
\langle uw\rangle+E_5\tau\langle w^2\rangle\frac{\partial U}{\partial z}-E_6\tau\frac{g}{\theta}\langle\theta u\rangle&=0;\\
\langle vw\rangle+E_5\tau\langle w^2\rangle\frac{\partial V}{\partial z}-E_6\tau\frac{g}{\theta}\langle\theta v\rangle&=0;\\
\langle uv\rangle+\frac{(1-C_2)}{C_1}\tau\left[\langle vw\rangle\frac{\partial U}{\partial z}+\langle uw\rangle\frac{\partial V}{\partial z}\right]&=0;\\
\langle\theta u\rangle+\frac{\tau}{C_{1\theta}}\left[\langle uw\rangle\frac{\partial\Theta}{\partial z}+(1-C_{2\theta})\langle\theta w\rangle\frac{\partial U}{\partial z}\right]&=0;\\
\langle\theta v\rangle+\frac{\tau}{C_{1\theta}}\left[\langle vw\rangle\frac{\partial\Theta}{\partial z}+(1-C_{2\theta})\langle\theta w\rangle\frac{\partial V}{\partial z}\right]&=0;\\
C_{1\theta}\langle\theta w\rangle+\langle w^2\rangle\frac{\partial\Theta}{\partial z}&=\tau(1-C_{3\theta})\frac{g}{\theta}\langle\theta^2\rangle.
\end{aligned} \quad (10)$$

Здесь [9] $C_1=1.8$; $C_2=0.6$; $C_3=0.5$; $C_{1\theta}=3.0$; $C_{2\theta}=0.346$; $C_{3\theta}=1/3$; $E_1=2-2C_2$; $E_2=1-C_2$; $E_3=1-C_3$; $E_4=E_1-E_2$; $E_5=(1-C_2)/C_1$; $E_6=(1-C_3)/C_1$.

Система (10) из 9 линейных уравнений для 9 неизвестных турбулентных моментов решается в каждой точке области значимой турбулентности $k>k_{min}$ с помощью метода исключения Гаусса.

2.3. Область моделирования

Расчеты с помощью мезомасштабной метеорологической модели TSUNM3 проводились над областью размером 100 × 100 км. Горизонтальный шаг сетки составлял 1 км и 0,333 км. В вертикальном направлении рассматривался 41 уровень вычислительной сетки, нижние тринадцать горизонтальных уровней сетки находились в приземном километровом слое. Координаты центра области (56,5° с.ш., 85° в.д.) совпадают с центром города Томск ((0,0) на рис. 1).

Высота земной поверхности и распределение категорий землепользования для области исследования были получены с использованием статических геоданных WPS

(https://www2.mmm.ucar.edu/wrf/users/download/get_sources_wps_geog_V3.html), а также с более высоким разрешением данных SRTM-миссии (Shuttle Radar Topography Mission) и оптических снимков со спутников Sentinel. На рис. 1 для примера представлен рельеф земной поверхности для области 32 × 32 км с разрешением 1 км и с разрешением 333 м.

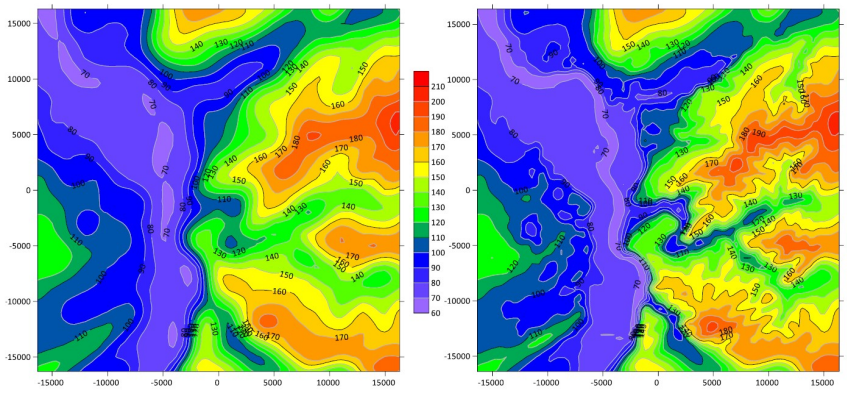


Рис. 1. Рельеф земной поверхности из различных источников с различным горизонтальным разрешением. Слева – GTOPO30 ($\Delta x = 1$ км), справа – SRTM ($\Delta x = 333$ м). Фиолетовым цветом указано русло реки Томь

3. Численный метод и параллельная реализация

Для системы уравнений (1) – (9) перед ее численным решением методом сеток применялось преобразование координат вида:

$$\begin{cases} \xi = x; \\ \eta = y; \\ \sigma = H\dfrac{z - h(x,y)}{H - h(x,y)}. \end{cases} \tag{11}$$

Здесь H – высота области исследования, $h(x, y)$ – высота рельефа подстилающей поверхности над уровнем моря. Преобразование (11) позволяет отобразить трехмерную область с криволинейной нижней границей на параллелепипед. Заметим, однако, что в результате выполнения такого преобразования в уравнениях (1) – (9) в новых переменных появляются смешанные производные.

Для дискретизации дифференциальной задачи (1) – (9) используется равномерная по горизонтальным направлениям сетка, допускающая сгущение горизонтальных сеточных плоскостей при приближении к поверхности Земли. Аппроксимация дифференциальной задачи с выполненным преобразованием (11) осуществляется на основе метода конечного объема [10]. Неизвестные значения компонент скорости соотносятся с центрами соответствующих граней конечных объёмов, а неизвестные скалярные характеристики (температура, абсолютная влажность, энергия турбулентности, ...) – с центрами конечных объемов. Аппроксимация конвективных членов уравнений переноса выполняется с использованием противопотоковой схемы MLU (Monotonized Linear Upwind) Ван Лира [11]. При дискретизации по времени используется явно-неявная разностная схема, в которой только диффузионные слагаемые по вертикальной координате аппроксимируются по схеме Кранка-Николсон, а остальные члены уравнений – по схеме Адамса-Бэшфорда. В результате для каждого горизонтального узла сетки получается система линейных алгебраических уравнений с трехдиагональной матрицей, которая на каждом шаге по времени решается экономичным методом прогонки

с целью получения значений сеточной функции вдоль вертикальной линии сетки на новом слое по времени.

Для расчета поля давления в данной работе используется его разложение на три составляющие: базовую, гидростатическую и негидростатическую. Для согласования поля скорости и давления на каждом шаге по времени применяется методика «предиктор-корректор» [6]. Негидростатическая часть давления находится из решения разностного уравнения Пуассона с использованием итерационного метода Гаусса-Зейделя [10], параллельная версия которого обеспечивает высокие показатели эффективности.

Для разработки параллельной программы, ориентированной на многопроцессорную вычислительную технику с распределенной памятью, в данной работе применялась технология параллельного программирования Message Passing Interface. Для рассматриваемой дифференциальной задачи и выбранного численного метода эта технология использовалась в сочетании с двумерной декомпозицией сеточной области по направлениям осей Ox и Oy. При такой декомпозиции каждому участвующему в вычислениях процессорному элементу (ПЭ – это вычислительное ядро центрального процессора с выделенной для него с помощью MPI локальной оперативной памятью) распределяются значения сеточной функции, принадлежащие локальной подобласти. В каждой подобласти одновременно и независимо решались трехдиагональные системы линейных уравнений.

Из-за используемого шаблона полунеявной разностной схемы для нестационарных конвективно-диффузионных уравнений (рис. 2, выделено в нижнем правом углу) для вычисления очередного приближения в приграничных узлах каждой локальной подобласти необходимо знать значения сеточной функции из соседней граничащей подобласти. Для обеспечения однородности параллельных вычислений по периметру каждой подобласти двумерной декомпозиции использовались два ряда фиктивных ячеек в силу применения монотонизированной противопоточной схемы Ван Лира. Фиктивные ячейки (рис. 2, светлые узлы) используются только для хранения данных с соседнего процессорного элемента, рассчитывающего приграничные значения в соседней подобласти декомпозиции. Пересылка этих значений осуществляется с помощью функций библиотеки MPI. Для передачи сообщений между процессами применялись сдвоенные блокирующие функции MPI_Sendrecv. Замена этих функций на неблокирующие функции передачи сообщений MPI_Isend и MPI_Irecv дала около 5% выигрыша в общем времени выполнения тестовой задачи.

Таблица 1. Значения ускорения и эффективности параллельной программы.

Кол-во ПЭ p	4	16	36	64	144
Ускорение S_p	3,4	15,2	32	64	106,7
Эфф-ть E_p	0,85	0,95	0,89	1,0	0,74

При использовании вычислительной сетки $100 \times 100 \times 41$ были получены следующие значения ускорения и эффективности (таблица 1). Расчеты проводились на кластере Томского университета Cyberia (https://cyberia.scc.tsu.ru/). Характеристики вычислительного узла: 2 6-ти ядерных процессора Intel Xeon 5670 и 48 Гб ОЗУ, интерконнект InfiniBand QDR 40 Гбит/с. Время моделирования одного часа развития процессов в атмосферном пограничном слое в параллельном исполнении на 64 ядрах около 4 минут.

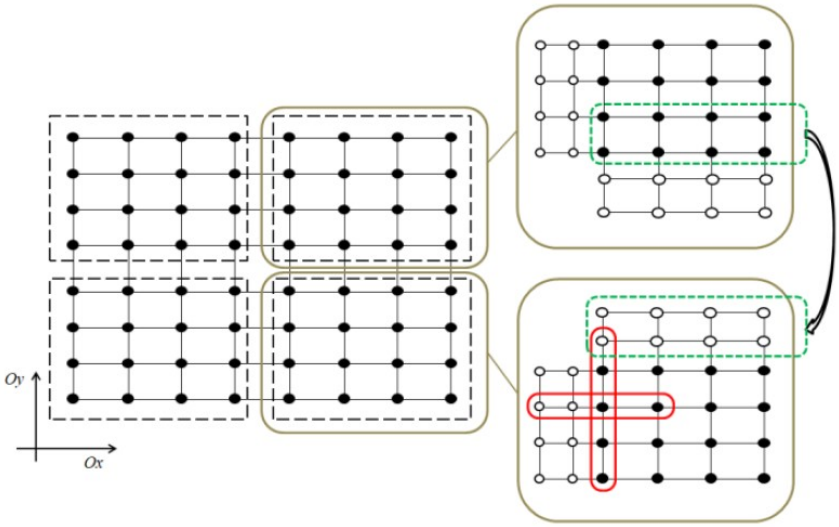


Рис. 2. Двумерная декомпозиция сеточной области вдоль направлений Ox и Oy и организация передачи данных между подобластями.

Причинами высокой масштабируемости программы TSUNM3 являются следующие:

- разработанный параллельный численный метод решения конвективно-диффузионного уравнения, как показывают приведенные выше результаты, обладает хорошей масштабируемостью вплоть до использования относительно небольшого числа узлов в подобластях сетки, полученных после применения двумерной декомпозиции; последнее связано с большим объемом вычислений коэффициентов разностных аналогов уравнений (1)–(9), особенно для блоков микрофизики влажности, турбулентности и теплопередачи;
- по-видимому, более интенсивном использовании быстрой кэш-памяти с увеличением степени декомпозиции конечно-разностной задачи при росте количества используемых параллельных процессов.

4. Результаты расчетов

В данной работе модель (1)–(9) была использована для численного исследования турбулентной структуры над Базовым экспериментальным комплексом (БЭК) ИОА СО РАН (85,10° в.д. и 56,42° с.ш.) с горизонтальным разрешением 1000 м и 333 м для 14 июля 2022 года. Расчетные значения метеорологических параметров сравнивались с приземными значениями температуры и скорости, измеряемыми с помощью ультразвуковой метеостанции «Метео-2», расположенной над поверхностью на высоте 10 м, а также вертикальными профилями абсолютной температуры в АПС, восстанавливаемыми с помощью температурного профилемера МТР-5.

На рис. 3 представлены рассчитанные и измеренные вертикальные профили температуры в нижнем 1000-метровом слое воздуха над пунктом наблюдения с интервалом в 4 часа местного времени 14 июля 2022 года. Вертикальный профиль температуры рассчитан в целом верно с небольшими отклонениями (не более 2°).

Результаты исследования показывают, что в ночные и утренние часы над пунктом наблюдения сформировался инверсионный слой, который был предсказан на сетках с различной плотностью узлов. После 8 часов утра за счет солнечного нагрева начала образовываться неустойчивая стратификация АПС. Для этих условий рас-

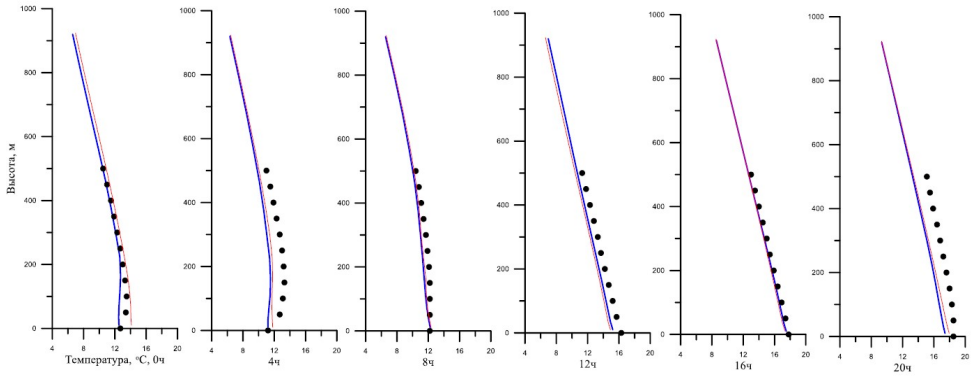


Рис. 3. Вертикальные профили температуры приземного слоя воздуха над Базовым экспериментальным комплексом ИОА СО РАН в различные моменты времени 14 июля 2022 года. Красные линии – расчеты с горизонтальным шагом сетки 1000 м, синие – 333 м, значки – измерения температурного профилемера МТР-5.

сматриваемые горизонтальные сеточные разрешения 0,333 км и 1,0 км дали близкие результаты.

Из рис. 4 видно, что в рассматриваемую дату 14 июля 2022 года модель TSUNM3, в целом, хорошо воспроизводит изменение метеорологических параметров в приземном слое воздуха в течение суток, хотя измерения выполняются в пункте наблюдений, а расчеты представляют осредненные на площади ячейки горизонтальной сетки раз-

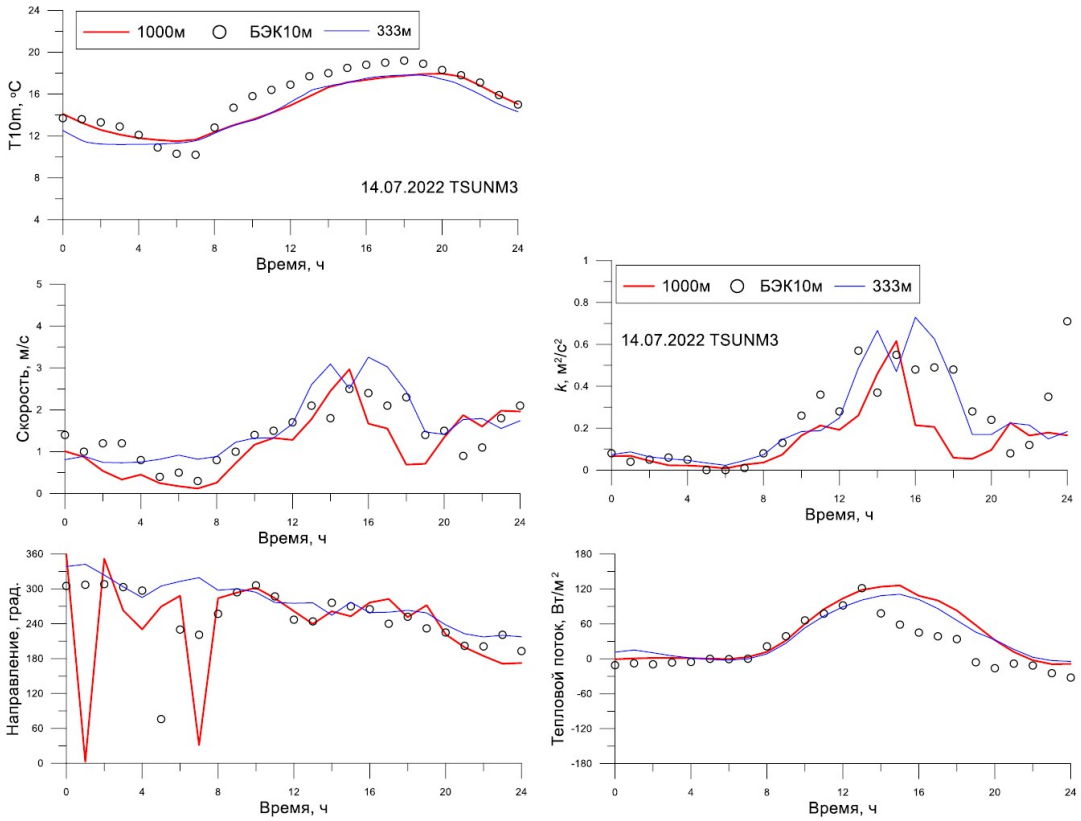


Рис. 4. Рассчитанные на сетке с горизонтальным шагом 1000 м (красные линии) и 333 м (синие линии) и измеренные с помощью метеостанции «Метео-2» БЭК ИОА СО РАН значения температуры, скорости, направления ветра, турбулентной энергии на высоте 10 м и адвективного теплового потока 14 июля 2022 года.

мером 1 × 1 км или 0,333 × 0,333 км. Сравнение наблюденных и рассчитанных температур показало, что суточные изменения отражены верно, различие составило около 2° утром и незначительное – в другое время. Рассчитанная и измеренная скорости ветра и турбулентной кинетической энергии удовлетворительно согласуются. Предсказанное направление ветра хорошо согласуется с расчетами.

Сравнивая результаты расчетов, выполненные на сетках с различной плотностью узлов, можно отметить, что повышение горизонтального разрешения, в целом, позволяет получить более близкие к наблюдениям результаты (таблица 2, 3), причем средние абсолютные ошибки численного прогноза приземной скорости ветра и температуры не превышают двух единиц, соответственно.

Таблица 2. Метрики MAE и RMSE для временных рядов 14 июля 2022 года.

Метеопараметр	Метрика	1000 м	333 м
Температура	MAE	1.008	1.309
	RMSE	1.218	1.399
Скорость ветра	MAE	0.466	0.386
	RMSE	0.572	0.509
Направление ветра	MAE	48.928	32.126
	RMSE	86.845	56.079
Турбулентная энергия	MAE	0.132	0.101
	RMSE	0.19	0.154
Адвективный тепловой поток	MAE	22.381	22.357
	RMSE	31.801	28.119

Заключение

Для исследования возможности расчета характеристик атмосферного пограничного слоя в «серой» зоне моделирования турбулентности с помощью осредненных по Рейнольдсу уравнений гидротермодинамики атмосферы рассмотрена нестационарная трехмерная трехпараметрическая модель турбулентности и алгебраические соотношения для напряжений Рейнольдса и турбулентных потоков тепла в приближении квазиоднородности пограничного слоя.

Рассматриваемая дифференциальная задача решается численно с помощью метода конечного объема, структурированных сеток, аппроксимаций второго порядка по времени и координатам. Для ускорения вычислений применяется двумерная декомпозиция сеточной области и технология интерфейса передачи сообщений. Такой подход обеспечил не менее 75% эффективность параллельной программы вплоть до использования 144 параллельных процессов.

Мезомасштабная модель с предлагаемой параметризацией турбулентности в пограничном слое с горизонтальным разрешением 0,333 км прошла тестирование на

Таблица 3. Метрики MAE и RMSE для вертикальных профилей 14 июля 2022 года (температура на высоте 0–500 м).

Время	Метрика	1000 м	333 м
0 ч	MAE	0.494	0.414
	RMSE	0.617	0.515
4 ч	MAE	1.171	1.36
	RMSE	1.205	1.465
8 ч	MAE	0.415	0.516
	RMSE	0.448	0.565
12 ч	MAE	1.147	0.872
	RMSE	1.164	0.892
16 ч	MAE	0.471	0.35
	RMSE	0.497	0.374
20 ч	MAE	1.325	1.628
	RMSE	1.37	1.646

измерениях, выполненных с помощью метеоприборов Базового экспериментального комплекса ИОА СО РАН. Получены результаты динамики вертикального распределения температуры в течение суток для условий г. Томск. В целом, применение в расчетах более подробной сетки обеспечивает уменьшение расхождения расчетов с данными наблюдений.

Список литературы

[1] Heinz S. The Atmospheric Gray-Zone (a.k.a. Terra Incognita) Problem: A Strategy Analysis from an Engineering Viewpoint // Fluids. 2025. V. 10(11). N. 301. https://doi.org/10.3390/fluids10110301.

[2] Honnert R., Efstathiou G.A., Beare R.J., Ito J., Lock A., Neggers R., Plant R.S., Shin H.H., Tomassini L., Zhou B. The Atmospheric Boundary Layer and the "Gray Zone" of Turbulence: A Critical Review // Journal of Geophysical Research: Atmospheres. 2020. V. 125 (13). DOI: 10.1029/2019JD030317.

[3] Juliano T.W., Kosović B., Jiménez P.A., Eghdami M., Haupt S.E., Martilli A. "Gray Zone" Simulations Using a Three-Dimensional Planetary Boundary Layer Parameterization in the Weather Research and Forecasting Model // Monthly Weather Review. 2022. Vol. 150. P. 1585–1619. DOI: 10.1175/MWR-D-21-0164.1.

[4] Mellor G.L., Yamada T. Development of a turbulence closure model for geophysical fluid problems // Rev. Geophys. 1982, Vol. 20. P. 851–875.

[5] Старченко А.В., Дель И.В., Сваровский А.И. Моделирование турбулентности

в атмосферном пограничном слое с использованием моментной алгебраической модели // Оптика атмосферы и океана. 2025. Т. 38. № 03. С. 222–231. DOI: 10.15372/AOO20250309.

[6] Численное моделирование погоды и качества атмосферного воздуха в городах / А.В. Старченко, Л.И. Кижнер, Е.А. Данилкин, Е.А. Шельмина [и др.]. Томск: Изд-во ТГУ, 2022. 140 с.

[7] Толстых М.А. и др. Система моделирования атмосферы для бесшовного прогноза. Гидрометеорол. науч.-исслед. центр Рос. Федерации. — М., 2017. — 166 с.

[8] Беликов Д.А., Старченко А.В. Численная модель турбулентного переноса примеси в пограничном слое атмосферы // Оптика атмосферы и океана. 2007. Т. 20. № 08. С. 667–673.

[9] Andren A. Evaluation of a turbulence closure scheme for air-pollution applications // J. Appl. Meteorol. 1990. V. 29. N 3. P. 224–239.

[10] Патанкар С. Численные методы решения задач теплообмена и динамики жидкости: Пер. с англ. / С. Патанкар. – М.: Энергоатомиздат, 1984. – 149 с.

[11] van Leer B. Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second order scheme // Journal of Computational Physics. 1974, V. 14. pp. 361–370.

[12] Starchenko A.V., Danilkin E.A., Prokhanov S., Kizhner L., Shelmina E. A Supercomputer-Based Modeling System for Short-Term Prediction of Urban Surface Air Quality // Supercomputing Frontiers and Innovations. 2022, 9(1). pp. 17–31. https://doi.org/10.14529/jsfi220102.

«Эпоха Келдыша» — ответ математиков на вызовы «Фултонской речи»

Т.А. Сушкевич

Институт прикладной математики им. М.В. Келдыша РАН

В современных условиях тектонических перемен в международных отношениях и геополитике вплоть до отмены международного права и новых военных угроз важно обратиться к исторической памяти. В 2026 году много памятных дат из истории науки и достижений в научно-технической революции XX века, а также юбилеев выдающихся ученых и деятелей науки, образования, техники. К ним добавилась новая дата памяти: 29 апреля 2026 года скончался Воеводин Владимир Валентинович (25.05.1962–29.04.2026) — талантливейший организатор и ключевой креативный участник революционных преобразований в сфере суперкомпьютеров, супервычислений, информационных технологий, искусственного интеллекта, квантовых вычислений — фундаментальных основ «цифровой» цивилизации в XXI веке — продолжатель дела развития «Эпохи Келдыша». В 2026 году впервые за 80 лет резонансно вспомнили «Фултонскую речь» Уинстона Черчилля 5 марта 1946 года в США. Речь У. Черчилля произвела колоссальное влияние на ход мировой истории и внешнюю политику США и Западной Европы. Центральное место заняли проблемы с Советским Союзом, который отгородили «железным занавесом». «Фултонская речь» стала триггером непрекращающейся «холодной войны» сначала с социалистическим СССР во главе с коммунистической партией, а после 1991 года с новой Россией. Причина кроется в двух главных вызовах, открыто озвученных в речи: заявлено, во-первых, о превосходстве англосаксов и, во-вторых, о правах на контроль ООН и управление миром «всей мощью США, Великобритании и других англоязычных стран и их союзников». Однако советские ученые и прежде всего математики во главе с Трижды Героем Социалистического Труда русским уникальным гением Мстиславом Всеволодовичем Келдышем силой интеллекта нанесли ошеломляющий удар по амбициям и планам англосаксов, когда впервые в истории человечества открыли космическую эру: 4 октября 1957 года запустили в космос первый искусственный спутник Земли, а 12 апреля 1961 года вся планета ликовала и аплодировала первому космонавту — советскому гражданину и коммунисту Ю.А. Гагарину. Гегемон повержен, СССР и США — всемирно признанные сверхдержавы.

Ключевые слова: М.В. Келдыш, Эпоха Келдыша, В.В. Воеводин-мл, Фултонская речь У. Черчилля, математики, ЭВМ, открытие космической эры, покорение космоса, сверхдержавы, историческое просвещение

1. Введение

В 2026 году на фоне тектонических сдвигов, возрастающей стохастической неуправляемой и непрогнозируемой турбулентности и радикальных в значительной степени негативных разрушительных перемен в сфере международных отношений, геополитики, международного права, а также нарастающих военных угроз, пожалуй, самыми значимыми являются две исторические юбилейные даты: 250 лет Независимости США (4 июля 1776 года принята «Декларация независимости США» от Королевства Великобритании и бывшие колонии получили статус соединенных штатов) и 35-летие исчезновения с мировой карты Союза Советских Социалистических Республик — ПЕРВОГО в истории человечества народного социалистического государства — ВЕЛИКОЙ ЦИВИЛИЗАЦИИ (Союзный Договор 1922 года). Не случайно впервые за

80 лет столь резонансной стала «Фултонская речь» 5 марта 1946 года Уинстона Черчилля (30.11.1874–24.01.1965). Впервые открыто проходят дискуссии на разных площадках, даже создали документальный фильм. А ранее фактически был запрет на эту «речь», чтобы соблюдать корректность и не обострять отношения с зарубежными «друзьями» из-за прошлого возрождающегося фашизма.

2025 и 2026 годы в исторической памяти в значительной степени связаны с Великой Отечественной Войной 1941–1945 гг.: в 2025 году 80-летие Великой Победы 9 мая 1945, а в 2026 году — 85-летие начала Войны в 4 часа утра 22 июня 1941 года. Война явилась и триггером и драйвером феноменального единения-монолита, героизма, мужества, энтузиазма и подъема творчества советского народа, а также небывалого взрыва мощи интеллекта, креативности и таланта советских ученых, профессоров, инженеров, конструкторов и высочайшего уровня профессионализма рабочих. На основе разработок и исследований, совершенных в годы войны, были получены многие достижения науки и техники, которыми пользуются по сей день. Советский народ, потеряв 27 миллионов граждан, вдохновленный ВЕЛИКОЙ ПОБЕДОЙ над фашизмом в самой страшной и кровавой Войне за всю историю человечества, был готов к новым подвигам и в мирное время. Долго ждать не пришлось. Новые угрозы открыто были объявлены претендентами на исключительное превосходство во всех сферах и власть в масштабах всей планеты.

Актуальнейшая тема, поднятая в публикации и докладе на конференции, гораздо глубже и значимее, чем только напоминание о «Фултонской речи» и математиках. Объем и формат статьи не позволяют представить множество фактов, документов, фотографий, которые в последние годы рассекречены и стали доступны для научного анализа. Однако важно отметить факты. Участник и свидетель реализации «Атомного проекта» Лев Дмитриевич Рябев (р. 08.09.1933) — ныне Научный руководитель РФЯЦ-ВНИИЭФ (Саров) возглавил большой коллектив единомышленников из Сарова и Снежинска и вместе совершили научный подвиг — рассекретили архивы Л.П. Берия и другие и в 1998–2010 гг. издали в бумажной версии историю «Атомного проекта» в документах и фактах в 14 книгах, а в 2019 году к 70-летию первого успешного испытания ПЕРВОГО советского «атомного заряда» 29 августа 1949 года создана и открыта для свободного открытого доступа «Электронная библиотека. История Росатома. Атомный проект СССР», которая непрерывно пополняется. Ничего подобного нет по истории «Космического проекта» и даже планы не известны, потому автор, как Пионер покорения космоса и ученица великого М.В. Келдыша, каждый год готовит и делает доклады по космической тематике и советских достижениях в покорении космоса. Это важно для исторического просвещения и консолидации поколений через «Память знаний» и тех эпохальных достижений в покорении и освоении космического пространства, где Родина была впереди планеты всей! Потомкам это надо знать, как бином Ньютона!

В 2026 году отмечается 115-летие со дня рождения академика М.В. Келдыша, и в календаре много значимых юбилейных дат, связанных с именем М.В. Келдыша — Ломоносова XX века, Лидера науки и величайшего русского гения и государственного деятеля за всю историю человечества, единственного математика Трижды Героя Социалистического Труда (1956, 1961, 1976), Главного математика страны, Научного руководителя проекта создания «Ракетно-ядерного щита», лучшего Президента Великой Академии наук СССР и многое другое. Важно вспомнить о конкуренции и сотрудничестве СССР и США в покорении космоса и ключевой роли «незаменимого» М.В. Келдыша — Главного теоретика, идеолога и стратега космонавтики, который отвечал за науку в стране и поднял престиж и приложения математики на такой высочайший уровень, что родилась новейшая технология «математика как производительная сила».

СССР — самая математическая страна в мире. Не случайно именно в СССР более 30 математиков стали Героями Социалистического Труда! При математиках — президентах Академии наук расцветала наука, а при математиках — ректорах университетов (И.Г. Петровский, Н.И. Лобачевский, В.А. Садовничий, О.М. Белоцерковский) сформировалось самое лучшее общее школьное и высшее образование. Кто правит в космосе, тот правит миром. Кто владеет математикой, тот лидер в «космической» и «цифровой» цивилизациях. И об этом обязаны знать и помнить потомки.

В 2026 году появилась новая дата памяти: 29 апреля скоропостижно скончался молодой член-корреспондент РАН Владимир Валентинович Воеводин (25.05.1962–29.04.2026) — талантливейший наследник и преемник дела «Эпохи Келдыша» и своего отца — моего друга академика Валентина Васильевича Воеводина (22.03.1934–27.01.2007). Воеводин-ст. познакомил меня с Воеводиным-мл., когда он был еще студентом и уже посещал научные семинары Г.И. Марчука. Многое можно написать про достижения и прекрасные качества Владимира, а главное — нет Владимиру замены такого же масштаба. И эту статью я посвящаю памяти Владимира Воеводина, с которым успели обсудить и название и содержание доклада и статьи и он меня благословил, потому что его всегда интересовала истина, правда и увлекался «историей знаний»! Не случайно Владимир стал Лидером в суперкомпьютерах и супервычислениях — это самые точные науки!

С целью исторического просвещения особое внимание в этой работе уделяется четырем юбилейным событиям в 2026 году, связанным с космической тематикой и математикой. С одной стороны, 115-летие со дня рождения гения-математика Мстислава Всеволодовича Келдыша (10.02.1911–24.06.1978) и 65-летие первого полета в космос первого космонавта Ю.А. Гагарина (09.03.1934–27.03.1968) — подарок к 50-летию Главного Теоретика космонавтики М.В. Келдыша — единственного ученого в истории цивилизации, именем которого названа «Эпоха Келдыша». С другой стороны, 250-летие со дня принятия Декларации о независимости США и 80-летие «Фултонской речи» Уинстона Черчилля в США.

М.В. Келдыш гениально выдерживал жесточайшую конкуренцию СССР и США в интеллектуальной сфере и находил эффективные ответы на вызовы со стороны США и их союзников, поскольку как сильнейший стратег с системно-логическим математическим мышлением выбрал три главных приоритетных проекта в государственном масштабе и не ошибся! Эти приоритеты позволили совершить научно-техническую революцию в XX веке: во-первых, покорение атома и «Атомный проект»; во-вторых, покорение космоса и «Космический проект»; в-третьих, разработка и создание электронно-вычислительных машин (ЭВМ), а в итоге из этих проектов естественным путем вышел проект создания «Ракетно-ядерного щита», который и в настоящее время сдерживает ядерные войны. Более того, эти три проекта стимулировали развитие многих направлений науки и образования, а также разных отраслей промышленности и техники. В итоге весь мир в XXI веке вышел на новый этап интенсивного инновационного развития и уже не мыслит жизни без «цифровой» и «космической» цивилизаций, которые породили искусственный интеллект — новую реальность.

2. «Фултонская речь» У. Черчилля — триггер «холодной войны»

24 июня 1945 года прошел Парад Победы на Красной площади в Москве и прогремели салюты в городах-героях. 15–31 июня 1945 года в Москве и Ленинграде прошел «Победный Парад Науки» по случаю Победы, 220-летия Академии наук и 20-летия

Великой Академии наук СССР (АН СССР), сыгравшей важнейшую роль в обеспечении Победы. Вторая мировая война закончилась, когда 2 сентября 1945 года в 04:00 утра по московскому времени на борту американского линкора «Миссури» были поставлены подписи под актом о безоговорочной капитуляции Японии, а для ускорения капитуляции 6 и 9 августа 1945 года США нанесли атомные удары по Японии. Чрезвычайно оперативно 20 августа 1945 года выходит историческое Постановление Государственного Комитета Обороны СССР «О Специальном комитете при ГОКО» № 9887сс/оп (Подпись: Председатель Государственного Комитета Обороны И. Сталин). Это был старт «Атомного проекта» и основания атомной отрасли в СССР, 80-летие которой отметили в 2025 году, — «ядерный щит» и мирная «атомная энергия» с тех пор занимают лидирующие позиции. СПАСИБО!

5 марта 1946 года в Вестминстерском колледже, г. Фултон, штат Миссури, США, перед молодежной аудиторией прозвучала знаменитая «Фултонская речь» Уинстона Черчилля под названием **«Мускулы мира»**. Партия У. Черчилля проиграла на выборах в Великобритании, а бывший премьер-министр не случайно приглашается президентом США Гарри Трумэном для выступления с программной речью. Превосходный оратор У. Черчилль *в октябре 1953 года получил Нобелевскую премию по литературе «за высокое мастерство произведений исторического и биографического характера, а также за блестящее ораторское искусство, с помощью которого отстаивались высшие человеческие ценности».* Речь Черчилля произвела грандиозное влияние на ход всей мировой истории до нынешних дней и дальнейшую внешнюю политику США и Западной Европы, в том числе особенно в отношении Советского Союза, а также современной России после 1991 года, когда СССР исчез с мировой карты. Следует обратить внимание на два главных вызова, озвученных в этой речи. Во-первых, открыто заявлено о превосходстве англоязычных стран (англосаксов) во всех сферах. Во-вторых, прозвучали такие выражения как «особые отношения», «железный занавес», «Мускулы мира», т.е. фактически был опущен «железный занавес» и была объявлена «холодная война» с СССР. У. Черчилль:

> «Протянувшись через весь континент от Штеттина на Балтийском море и до Триеста на Адриатическом море, на Европу опустился железный занавес. ... Мы не должны допустить повторения подобной катастрофы, и добиться этого сегодня, в 1946 году, возможно лишь путем налаживания нормальных отношений и всеобъемлющего взаимопонимания с Россией под эгидой Организации Объединенных Наций. Поддержание таких отношений в течение многих и многих мирных лет должно обеспечиваться не только авторитетом ООН, но и **всей мощью США, Великобритании и других англоязычных стран и их союзников**. Такова в основных чертах суть моих предложений, которые я позволил себе представить моей уважаемой аудитории в своем сегодняшнем выступлении, названном мною «Мускулы мира». ... В целом ряде стран по всему миру, хотя они и находятся вдалеке от русских границ, создаются коммунистические пятые колонны, действующие удивительно слаженно и согласованно, в полном соответствии с руководящими указаниями, исходящими из коммунистического центра. **Коммунистические партии и их пятые колонны во всех этих странах представляют собой огромную и, увы, растущую угрозу для христианской цивилизации**, и исключением являются лишь Соединенные Штаты Америки и Британское Содружество наций, где коммунистические идеи пока что не получили широкого распространения.»

И.В. Сталин адекватно оценил «Фултонскую речь» и предупредил всех о грядущих вызовах и угрозах, которые, как теперь стало очевидно, никогда не менялись после окончания Второй мировой войны и до настоящего времени, но в других формах — бомбежки разными средствами, теракты, экономические и культурные санкции, активная информационная и гибридная война (*Ответ корреспонденту «Правды» // Правда. 1946. 14 марта*):

> «Следует отметить, что господин Черчилль и его друзья поразительно напоминают в этом отношении Гитлера и его друзей. Гитлер начал дело развязывания войны с того, что провозгласил расовую теорию, объявив, что только люди, говорящие на немецком языке, представляют полноценную нацию. Господин Черчилль начинает дело развязывания войны тоже с расовой теории, утверждая, что только нации, говорящие на английском языке, являются полноценными нациями, призванными вершить судьбы всего мира. Немецкая расовая теория привела Гитлера и его друзей к тому выводу, что немцы как единственно полноценная нация должны господствовать над другими нациями. Английская расовая теория приводит господина Черчилля и его друзей к тому выводу, что нации, говорящие на английском языке, как единственно полноценные должны господствовать над остальными нациями мира.»

Ответная реакция И.В. Сталина была мгновенная. Уже 13 мая 1946 года вышло историческое Постановление Совета Министров Союза ССР №1017-419сс «Вопросы реактивного вооружения» (Подпись: Председатель Совета Министров Союза ССР И. Сталин), которое закладывало задачи по развитию реактивной техники и определило конкретные меры по реализации ракетно-ядерного оружия — создание ракетостроительной отрасли в промышленности и освоение ракетного оружия в войсках. В 2026 году отмечается 80-летие ракетно-космической отрасли. Это был старт начала «Космического проекта» и создания «Ракетно-ядерного щита». СПАСИБО!

Открытие космической эры 4 октября 1957 года и полет Юрия Гагарина в космос 12 апреля 1961 года — это исторический ошеломляющий сильнейший удар по амбициям и отмена «превосходства» англосаксов и во всем мире были признаны две сверхдержавы — США и СССР, который более 10 лет в покорении космоса был впереди планеты всей. Новый старт конкуренции в космосе, науке, промышленности — это когда 12 сентября 1962 года на весь мир прозвучала знаменитая «Лунная речь» президента США Джона Ф. Кеннеди, Хьюстон, Техас:

> «... сегодня наш долг — вырваться вперед в освоении Вселенной. Мы принимаем этот вызов, мы участвуем в этой гонке — и выйдем из нее победителями. Все взгляды сейчас устремлены в космос, к Луне, к окружающим нас планетам. И мы хотим, чтобы там реял флаг Соединенных Штатов, символ свободы и мира, а не знамя наших врагов. В космосе нет места для оружия массового поражения! Вселенная должна стать источником новых знаний, двигателем прогресса.»

Вскоре Д. Кеннеди убили.

3. «Эпоха Келдыша» — «золотой век науки» в истории СССР и России

После второй мировой войны и до 1991 года — «золотой век науки» в истории человечества, СССР, русского государства и русской цивилизации, где за «науку отвечал» государственный деятель мирового масштаба РУССКИЙ ГЕНИЙ МАТЕМАТИК М.В. Келдыш. «Золотая эпоха», «золотой век» — это период в истории мировой цивилизации, который характеризуется высоким уровнем развития науки, образования, культуры, технологий. В «золотую эпоху» происходит расцвет науки, технологий, техники, образования, культуры, искусства, что приводит к значительному прогрессу человечества. В XX веке покорили атом, космос и сотворили компьютеры, интернет, искусственный интеллект и другие технологии, которые изменили мир до неузнаваемости — это фундамент «цифровой» и «космической» цивилизаций XXI века. Это был век гениев и творцов, для которых смысл жизни выражен в словах М.В. Келдыша: «Наука — это жизнь, жизнь — это наука». Нобелевский лауреат (1956) Дважды Герой Социалистического Труда (1966, 1976) академик (химическая физика, 29.03.1932) Николай Николаевич Семенов (15.04.1896–25.09.1986) кратко и очень метко выразил жизненное кредо ученых-творцов в XX-ом веке, которые развивали МАТЕМАТИКУ, покоряли КОСМОС, создавали компьютеры, лазеры, электронику и т.д.: «Я не мыслю другой жизни, как жизнь вместе с наукой!»

Никто из математиков и философов, в том числе математик, физик, космолог А.А. Фридман (16.06.1888–16.09.1925) — ученик организатора АН СССР, гениального математика В.А. Стеклова (09.01.1864–30.05.1926), академики математики-философы В.И. Вернадский (12.03.1863–06.01.1945) — основоположник учения о ноосфере и «научная мысль как планетное явление», и Н.Н. Моисеев (23.08.1917–29.02.2000) в трудах «Человек и ноосфера», «Судьба цивилизации. Путь Разума», не смог предсказать, что теоретические достижения в физике и математике, а также открытие космической эры человечества в середине XX века станут фундаментом формирования и развития «Цифровой реальности» в начале XXI века на всей планете: «Цифровая экономика», «Киберпространство», «Кибервойны», «Цифровая Вселенная», «Цифровая Земля», «Цифровая физика», «Цифровая философия», «Теория информации», «Квантовая теория информации», «Синергетическая теория информации», «Цифровая передача информации», «Цифровые изображения», «Цифровая анимация», «Цифровое фотографирование», «Цифровое кино», «Электронный учебник», «Электронный ресурс», Big Data, «Основы кодирования», «Теория алгоритмов», «Криптография», «Госуслуги», «Автоматизация производства», «Логистика» в разных сферах приложений и многое другое, включая «Цифровые технологии» на выборах и т.д.

В 2026 году отмечается 115-летний юбилей М.В. Келдыша и много других значимых юбилейных дат, связанных с именем М.В. Келдыша — Ломоносова XX века, Лидера науки и величайшего русского гения МАТЕМАТИКА — уникального организатора и руководителя эпохальных стратегических проектов, которые определяли развитие мировой цивилизации. За всю историю человечества никогда и ни в одной стране мира ни один МАТЕМАТИК не играл такую ключевую роль в судьбе страны и планеты как **всемирно известный, всемерно засекреченный** М.В. Келдыш! Важно вспомнить о конкуренции и сотрудничестве СССР и США в покорении космоса и ведущей роли «незаменимого» М.В. Келдыша — Главного теоретика, идеолога и стратега космонавтики, который отвечал за науку и образование в стране и поднял престиж и роль именно «математики как производительной силы» на такую высоту, что в XXI веке в новом стандарте подготовки универсальных специалистов будущего

для STEM-отраслей (естественные науки — физика, химия, биология, технологии, инженерия и математика) математику выделили как самостоятельный важнейший раздел научно-образовательной политики в ведущих университетах! В некоторых странах добавили искусство и на английском языке в международном общении это выглядит так: STEAM — science, technology, engineering, arts and mathematics.

Однако, стоит напомнить, что такая политика действовала в СССР после войны на физическом факультете МГУ имени М.В. Ломоносова и в знаменитом МФТИ: не узкая специализация, как у большинства, а престижное универсальное образование для талантливой молодежи. Профессия автора публикации из поколения «физики и лирики» — выпускницы с отличием кафедры Героя Социалистического Труда за первые атомную и термоядерную бомбы (Указ от 04.01.1954) А.Н. Тихонова «высшая математика» на физическом факультете МГУ имени М.В. Ломоносова, а с 1961 года в «Институте Келдыша», — наглядная иллюстрация лучшего в мире советского образования — это синтез («гремучая смесь») физика-теоретика и математика по специальности «теоретическая и математическая физика» с новейшей технологией — компьютерным моделированием на всех поколениях ЭВМ, начиная с первых «Стрела» (1960) и «Весна» (1964), информационными технологиями (с Ю.М. Баяковским построили на ЭВМ «Весна» первый в СССР компьютерный график и создали первую компьютерную анимацию), и широкий спектр приложений в космических исследованиях — единственная в мире женщина-ученая Пионер покорения космоса.

Руководство молодой советской страны сделало четыре шага к успеху МАТЕМАТИКОВ.

ПЕРВЫЙ ШАГ к УСПЕХУ МАТЕМАТИКОВ — это создание в 1925 году Академии наук СССР (по решению И.В. Сталина)! В 2025 году особое внимание заслуживает 100-летие Великой Академии Наук СССР, без которой не было бы ВЕЛИКОЙ ПОБЕДЫ!

ВТОРОЙ ШАГ к УСПЕХУ МАТЕМАТИКОВ — это создание в 1934 году Математического института им. В.А. Стеклова АН СССР, в котором М.В. Келдыш защитил три диссертации, и филиалов в Ленинграде (ЛОМИ, 1940), Свердловске (1956), Новосибирске (1957)!

ТРЕТИЙ ШАГ к УСПЕХУ МАТЕМАТИКОВ — это создание в 1953 году (при личной поддержке И.В. Сталина, но Распоряжение СМ СССР № 6111-рс от 18 апреля 1953 г. подписано Л.П. Берия) ПЕРВОГО в мире Института прикладной математики Академии Наук СССР; первое название: Отделение прикладной математики МИАН СССР со статусом «Институт» (ОПМ МИАН СССР), с 1953 по 1966 гг. был «совершенно секретным» и даже имел статус «почтового ящика 2287», в историю вошел как «Институт Келдыша»!

ЧЕТВЕРТЫЙ ШАГ к УСПЕХУ МАТЕМАТИКОВ — это разработка и создание промышленных ЭВМ (по личным распоряжениям И.В. Сталина)!

В СССР с 1947 года по поручению И.В. Сталина работами по разработке и созданию ЭВМ руководил М.В. Келдыш и первая серийная промышленная ЭВМ «Стрела» — советская ЭВМ первого поколения — создана под его руководством и даже с личным участием (М.В. Келдыш и М.Р. Шура-Бура разрабатывали первую систему команд и ПО) и установлена в «Институте Келдыша» в 1953 г., введена в строй в 1954 г. Базовой в «Ракетно-ядерном щите» была ЭВМ БЭСМ-6 — второго поколения на транзисторах. Первый экземпляр установлен в ИПМ в 1966 г., введен в строй в 1967 г. Правда состоит в том, что разработка ЭВМ являлась составной частью стратегического проекта по созданию «Ракетно-ядерного щита» и осуществлялась в секретном режиме. Без ЭВМ космос покорить нельзя! Конкуренция с США

проходила по трем направлениям — трем составным частям проекта: освоение атома, покорение космоса и создание ЭВМ! В 1954 году разработчики были удостоены Сталинской премии. Главному конструктору Юрию Яковлевичу Базилевскому было присвоено звание Героя Социалистического Труда с вручением ордена Ленина и Золотой медали «Серп и Молот» Указом Президиума Верховного Совета СССР от 27 октября 1954 года за разработку и создание автоматической быстродействующей электронной вычислительной математической машины. Среди награжденных будущий академик и конструктор не только ЭВМ «Весна» (август, 1964), но и главный конструктор современных отечественных суперкомпьютеров академик Владимир Константинович Левин (05.03.1929–04.01.2026). На ЭВМ «Стрела» проводились расчеты запуска первого ИСЗ и полета Ю. Гагарина.

Высокий статус математики — это исключительно заслуга М.В. Келдыша в мировом масштабе при поддержке Андрея Николаевича Тихонова (30.10.1906–08.10.1993) и Ивана Георгиевича Петровского (18.01.1901–15.01.1973)! Нигде никогда ни в одной стране, кроме СССР, «чистый» математик не был руководителем и организатором науки и образования в масштабах большой страны, теоретиком, стратегом, идеологом, министром и генералом, под личную ответственность которого власти разрешали рисковые запуски космических кораблей. Об этом в 2007 году говорил Борис Евсеевич Черток (01.03.1912–14.12.2011) на Торжественном заседании в зале «Академический», посвященном 50-летию запуска первого спутника Земли и открытия космической эры. Присутствовали гости из НАСА и открыто признавали советский ПРИОРИТЕТ!

В СССР «незаменимый» математик М.В. Келдыш — величайший ГЕНИЙ за всю историю цивилизации: при жизни осуществил многовековые мечты человечества о выходе в космос и дал ответы на вызовы «Фултонской речи» — вместе с С.П. Королевым открыли космическую эру и более 10 лет СССР был впереди планеты всей — «математические формулы» мощью интеллекта с помощью ЭВМ гениально были реализованы в космических полетах:

- первого спутника (04.10.1957) и первого человека (12.04.1961);
- первых АМС на Луну (02.01.1959 «Луна-1»; 04.10.1959 «Луна-3» достигла поверхности; 12.09.1959 «Луна-2» фотографирование обратной стороны);
- на Венеру (12.02.1961 — 19.05.1961 «Венера-1»; 12.11.1965–27.02.1966 «Венера-2»; 16.11.1965–01.03.1966 «Венера-3» достигла поверхности);
- на Марс (в 1960–1969 гг. девять зондов в направлении Марса; в 1971 г. исследовательский зонд «Марс-2» первым добрался до поверхности планеты);
- первых Долговременных орбитальных станций — ДОС «Салют» (19.04.1971), ДОС «Салют-3» — «Космос-557» (11.05.1973), ДОС «Салют-4» (26.12.1974), ДОС «Салют-5» (22.06.1976).

Советско-американский проект «Союз-Аполлон» — это торжество сотрудничества Академии наук СССР и НАСА, СССР и США и триумф «Эпохи Келдыша» и лично М.В. Келдыша.

Полет Ю.А. Гагарина и советско-американская Программа «Союз-Аполлон» — это прежде всего политика и конкуренция двух сверхдержав — драйверы прогресса. А «Фултонская речь» — это триггер «холодной войны» и «железного занавеса», которые так и продолжаются, то временно замирая, то резко обостряясь.

А.Н. Несмеянов (09.09.1899–17.01.1980), Президент АН СССР (16.02.1951–19.05.1961): «Для нас, ученых страны социализма, запуск спутника — двойной праздник: это праздник рождения новой эры в завоевании человечеством природы — космической эры существования человечества — и это праздник мужественной зрелости советской науки.» М.В. Келдыш поднял престиж и роль математики на такую

высоту, что общепризнанной стала новейшая технология «математика как производительная сила», которая занимает почетное место после электричества, радио и связи, электроники, атомной энергии и это уже навсегда, что доказано в XXI веке. Показателен научный приоритет математики: после химика А.Н. Несмеянова 19 мая 1961 года впервые с основания Президентом Академии наук избран математик М.В. Келдыш (19.01.1961–19.05.1975) и за исключением физиков А.П. Александрова (13.02.1903–03.02.1994, президент 25.11.1975–16.10.1986) и В.Е. Фортова (23.01.1946–29.11.2020, президент 29.05.2013–23.03.2017) во главе Академии наук были математики Г.И. Марчук (08.06.1925–24.03.2013, президент 16.10.1986–17.12.1991) и Ю.С. Осипов (07.07.1936, президент 17.12.1991–29.05.2013). Все математики и физики — участники трех советских стратегических проектов.

Важно знать и понимать какой героический подвиг совершили математики: «Атомный проект» в СССР начинали оперативно и быстро развивать тогда, когда не было ЭВМ, и три первых главных достижения в СССР — это 29 августа 1949 года первое успешное испытание «атомной» бомбы, которое привело к паритету СССР и США в «атомной гонке», а первое успешное испытание первой в мире водородной («ядерной») бомбы 12 августа 1953 года и 26 июня 1954 года успешный запуск первой атомной станции (Математический отдел Е.С. Кузнецова в ФЭИ, Обнинск) вывели СССР в лидеры и были реализованы благодаря уникальному таланту советских ученых, инженеров, конструкторов, мобилизации всего советского народа и особенно высочайшему уровню математиков — расчеты проводились вручную на электрических настольных счетных машинках группами расчетчиц с математическим образованием, но с оригинальными находками в численных методах и первых параллельных алгоритмах.

Чтобы погрузиться в эпоху начала работ по «Ракетно-ядерному щиту», стоит ознакомиться с Трудами Международного симпозиума «Наука и общество. История советского атомного проекта (40-е — 50-е годы)», который проходил в Дубне 14–18 мая 1996 года. Работы имели гриф сов.секретно и только на этой конференции впервые А.А. Самарский рассказал, а потом и статью написал «Прямой расчет мощности взрыва». По оценке Л.Д. Ландау (22.01.1908–01.04.1968), А.Н. Тихонов совершил научный подвиг и получил первую Звезду Героя! Теоретический прогноз оценки мощности взрыва впервые вручную проводился на основе прямого численного расчета первой атомной бомбы (1949 г.) в Лаборатории А.Н. Тихонова при ГЕОФИАН СССР, а первой термоядерной «водородной» бомбы (1953 г.) — в отделе № 3 «Института Келдыша» (ОПМ МИАН СССР) под руководством член-корреспондента АН СССР с 29.01.1939, Отделение математических и естественных наук, специальность «геофизика, математическая физика», заведующего кафедрой «высшая математика» профессора физического факультета МГУ А.Н. Тихонова при участии А.А. Самарского (19.02.1919–11.02.2008), Б.Л. Рождественского (28.09.1928–01.08.2001), Н.Н. Яненко (22.05.1921–16.01.1984), В.Я. Гольдина (16.06.1924–29.05.2014).

В Распоряжении СМ СССР № 6111-рс от 18 апреля 1953 г. «Об образовании Отделения прикладной математики Математического института АН СССР», сов.секретно, написано о работах по «Атомному проекту», который курировало Первое главное управление:

1. Образовать в Математическом институте им. В.А. Стеклова Академии наук СССР Отделение прикладной математики на базе расчетно-математических бюро, руководимых академиками Петровским и Келдышем, и вычислительного бюро Геофизического института, руководимого чл.-кор. Академии наук СССР Тихоновым.
2. Возложить на Отделение прикладной математики Математического института им.

В.А. Стеклова Академии наук СССР выполнение расчетных работ, составление математических таблиц специальных функций и развитие соответствующих областей математики по планам и под контролем Первого главного управления при Совете Министров СССР.

3. Назначить директором Отделения прикладной математики Математического института им. В.А. Стеклова Академии наук СССР, на правах директора института, акад. Келдыша М.В. и заместителем директора — чл.-кор. Академии наук СССР Тихонова А.Н., освободив его от работы в Геофизическом институте Академии наук СССР.

Однако гениальный М.В. Келдыш по личной инициативе (Келдышу позволяли все!) при основании ОПМ МИАН СССР в том же 1953 году собрал своих учеников и единомышленников и создал Отдел № 5 во главе с Дмитрием Евгеньевичем Охоцимским (26.02.1921–18.12.2005) для проведения перспективных поисковых фундаментальных научно-исследовательских работ в интересах будущего выхода в космос. А 14 февраля 1954 года в кабинете директора состоялось историческое первое рабочее совещание, которое организовал М.В. Келдыш. На это совещание были приглашены С.П. Королев, П.Л. Капица, Л.И. Седов, С.Э. Хайкин, И.А. Кибель, Д.Е. Охоцимский, Т.М. Энеев, В.А. Егоров, В.А. Сарычев, М.К. Тихонравов, Г.Ю. Максимов, И.М. Яцунский. На совещании обсуждались примерные сроки и технические вопросы запуска первого искусственного спутника Земли, научные проблемы, которые предполагалось решить с помощью аппаратуры на искусственных спутниках. Это было стратегическое решение — через несколько лет «Институт Келдыша» становится головным по стратегическим космическим проектам, Отдел № 5 стал самым большим, а М.В. Келдыш, Д.Е. Охоцимский, Г.И. Петров получили Звезды Героев Социалистического труда за обеспечение успешного полета Юрия Гагарина!

В Постановлении СМ СССР от 13 мая 1946 г. «Вопросы реактивного вооружения» фактически учреждается ракетная отрасль, как межведомственное образование (на базе семи министерств-участников ракетной программы), и создается как чрезвычайный руководящий и координирующий орган Спецкомитет по реактивной технике при СМ СССР. Для обеспечения внедрения результатов работ по реактивной технике, выполняемых в научно-исследовательских институтах и конструкторских бюро, в производство были созданы органы оперативного и повседневного руководства: в Министерствах — участниках ракетного проекта были созданы Главные Управления по реактивной технике, а в Госплане СССР — отдел по реактивной технике.

Никакими директивными документами новое направление «на пустом месте» в науке и суперсовременной отрасли ракетной промышленности создать невозможно. Необходимо наличие собственной теоретической базы, длительной практики изобретательской и конструкторской деятельности, научной преемственности нескольких поколений. Советский Союз располагал «исторической наследственностью», которая не прерывалась даже в трудные годы войны. «Русский космизм» и идеи покорения космоса — это в ментальности Русской Цивилизации. И. Сталин обладал уникальными способностями привлекать талантливых ученых и организаторов и при его личном участии появился самый эффективный коллектив научных руководителей стратегических проектов в атомной и ракетно-космической отраслях: М.В. Келдыш — математик, И.В. Курчатов — физик, С.П. Королев — инженер-конструктор. Покорять космос могли только при наличии должного уровня математики и больших электронных вычислительных машин (ЭВМ). 1946 год ключевой в судьбе М.В. Келдыша: 80 лет назад 30 ноября в возрасте 35 лет избирают академиком по специальности «математика, механика» в Отделении технических наук и признают Лидером по

«прикладной математике»; 2 декабря впервые математика назначают Начальником подлежащего возрождению Реактивного НИИ (НИИ-1 МАП, ныне это «Исследовательский Центр имени М.В. Келдыша», входит в Роскосмос) — преемника первых Лабораторий и первого НИИ, где проводились пионерские работы по созданию ракет с 1933 года; ЦАГИ выдвигает работы М.В. Келдыша по безопасности авиации на вторую Сталинскую премию, которую получил в 1947 году.

Три ВЕЛИКИХ МАТЕМАТИКА — три друга в МИАН и в МГУ приняли на себя ответственность и с 1951 года руководили наукой и образованием и прикладными работами с участием математиков и ЭВМ, родились в Российской Империи раз в пять лет, а состоялись в СССР:

- Мстислав Всеволодович Келдыш (10.02.1911–24.06.1978),
- Андрей Николаевич Тихонов (30.10.1906–08.10.1993),
- Иван Георгиевич Петровский (18.01.1901–15.01.1973).

Крайне важно напомнить, что в 2025 году впервые за последние 25 лет утвержден Национальный проект «Развитие космической деятельности Российской Федерации на период до 2030 года и на перспективу до 2036 года», который начал действовать 1 января 2026 года. В нацпроекте «Космос» сформулированы «космические цели»: чем займется «Роскосмос» и другие смежные организации в ближайшие 10 лет. Национальный проект «Космос» охватывает спутниковую связь, доступ в интернет, запуск новой орбитальной станции, развитие ракетной техники, подготовку кадров и т.д. С 6 по 12 апреля 2026 года на основании Указа Президента Российской Федерации от 29.12.2025 № 995 «О проведении в Российской Федерации Недели космоса» прошла первая в нашей стране неделя космоса. 9 апреля 2026 года состоялся Российский космический форум «Космическая повестка-2030: глобальные вызовы и национальные стратегии».

«Неделя космоса» вернула нашу память к событиям, свидетелем которых был весь мир. Московский университет — тоже часть этой истории. 13 апреля в Интеллектуальном центре — Фундаментальной библиотеке МГУ под председательством ректора академика В.А. Садовничего состоялось расширенное первое заседание созданного Научно-координационного совета по космосу МГУ, которое посвятили 65-летию исторического полета Ю.А. Гагарина и на котором вспомнили М.В. Келдыша. За разработку программно-математического обеспечения космических исследований выпускники МГУ председатель Национального комитета по теоретической и прикладной механике РАН академик И.Г. Горячева и экс-директор ИПМ имени М.В. Келдыша АН СССР/РАН академик А.И. Аптекарев вручили В.А. Садовничему медаль имени Д.Е. Охоцимского, учрежденную в память профессора Московского университета и одного из самых ярких участников советского космического проекта. Выпускник мех-мата заведовал кафедрой теоретической механики/теоретической механики и мехатроники механико-математического факультета (1962–2005), заведовал кафедрой теоретической механики физико-технического факультета (1947–1951), а также создал и руководил кафедрой в МФТИ и первой кафедрой робототехники в Бауманском училище. Д.Е. Охоцимский (26.02.1921–18.12.2005) — ученик М.В. Келдыша со времен его обучения на мех-мате МГУ и Герой социалистического Труда (Указ от 17.06.1961) за обеспечение полета Ю.А. Гагарина, после МГУ начал работать в отделе механики Математического института имени В.А. Стеклова, а далее всю жизнь проработал в «Институте Келдыша» с первых дней его основания и заведовал самым большим отделом № 5, который принимал ведущее активное участие в космических проектах, а также стал кузницей новых кадров по робототехнике и компьютерному зрению, когда мало кто знал о таких направлениях деятельности.

4. «Незаменимый» М.В. Келдыш — фотодокументы убедительнее слов

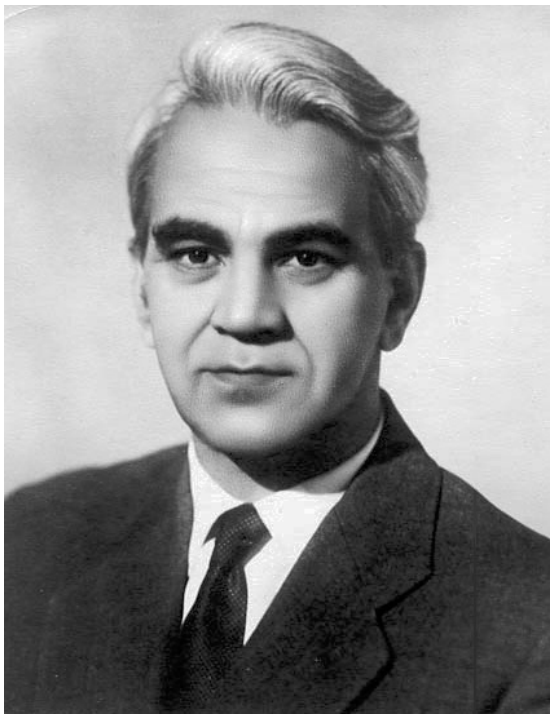


Рис. 5. 30.11.1946 избран академиком и признан Лидером по «прикладной математике».

Рис. 6. Трижды Герой Социалистического Труда (1956, 1961, 1971).

Рис. 7. М.В. Келдыш член Совета Главных (из личного архива).

Члены Государственной комиссии по испытаниям ракеты Р-7 и Первого в Мире искусственного спутника Земли, а также руководители испытаний на 10-й площадке 5-го Научно-исследовательского испытательного полигона № 5 Министерства обороны СССР. (октябрь 1957 года).

Сидят (слева направо):
Григорий Рафаилович Ударов, Анатолий Иванович Семёнов, Александр Григорьевич Мрыкин, Мстислав Всеволодович Келдыш, Сергей Павлович Королёв (технический руководитель испытаний), Василий Михайлович Рябиков (председатель комиссии), Митрофан Иванович Неделин (заместитель председателя комиссии), Георгий Николаевич Пашков, Михаил Сергеевич Рязанский (заместитель технического руководителя испытаний), Константин Николаевич Руднев (заместитель председателя комиссии), Валентин Петрович Глушко (заместитель технического руководителя испытаний), Владимир Павлович Бармин (заместитель технического руководителя испытаний), Виктор Иванович Кузнецов (заместитель технического руководителя испытаний).

Стоят (слева направо):
Павел Ефимович Трубачев, Георгий Александрович Тюлин, Николай Николаевич Смирницкий, Николай Алексеевич Пилюгин, Анатолий Алексеевич Васильев, Василий Иванович Ильюшенко, Александр Иванович Носов, Алексей Федорович Богомолов, Константин Дмитриевич Бушуев, Владимир Иванович Курбатов, Константин Васильевич Герчик (начальник полигона).

Рис. 8. Госкомиссия (октябрь 1957 года) и единственный математик М.В. Келдыш среди них.

Заключение

В условиях новых военных угроз и нарушения стабильности в мировых масштабах главные юбилейные даты — 85 лет начала Великой Отечественной войны 1941–1945 гг. 22 июня 1941 года, 80 лет Великой Победы 9 мая 1945 года, 80 лет «Фултонской речи» 5 марта 1946 года и объявление «холодной войны», 65 лет полета Юрия Гага-

Рис. 9. Кто главный? После вручения второй звезды Героя Социалистического труда.

Рис. 10. С.П. Королев и М.В. Келдыш провожают Ю. Гагарина в первый полет в космос.

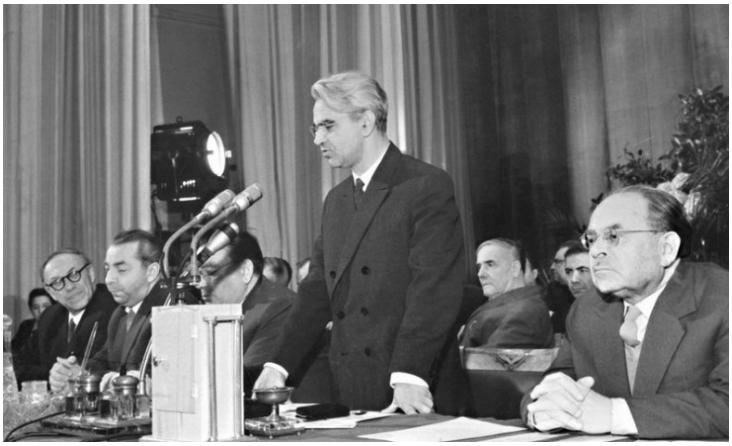


Рис. 11. 19 мая 1961 г. М.В. Келдыш избран Президентом АН СССР. АРАН. Ф.1729. Оп.1. Д.50.

Рис. 12. Любимый Герман Титов в «Институте Келдыша», 1962 г.

рина 12 апреля 1961 года, 35 лет исчезновения ВЕЛИКОЙ РУССКОЙ Цивилизации СССР в 1991 году — вызвали значительный рост интереса разных поколений к истории Родины и мира. Появилось много новых и разных источников информации, документов, фактов. Тематика доклада стала актуальнейшей, поскольку поднимает проблемы зарождения современных реальностей — «цифровой» и «космической» цивилизаций, когда возникла новейшая технология «математика как производительная сила» и математика стала не только «царицей всех наук», но и «царицей войны и мира». В рамках статьи невозможно отразить весь спектр проблем и вспомнить о талантливом поколении ученых, инженеров, конструкторов, государственных деятелей, руководителей науки и организаторов ответных мер на вызовы и угрозы. Предстоит серьезная глубокая научно-исследовательская работа, пока еще сохранились участники научно-технической революции XX века, которая свершилась при жизни одного поколения, к которому относится автор (р. 1940 г.) публикации. Для интересующихся математикой и космической тематикой, а также для студентов, аспирантов, молодых специалистов рекомендую ознакомиться с публикациями из обновленного списка Литературы [1–57].

Из всех вызовов «Фултонской речи» остались пока только два рудимента — английский язык как язык международного общения и доллар, к которому пока еще привязаны экономики практически всех стран, однако начался процесс ослабления зависимости от доллара и усиления роли национальных валют. Что касается пре-

восходства в интеллектуальной сфере, то СССР дал два мощных асимметричных ответа. Во-первых, создали самые сильные академические и вузовские математические научные институты и научные школы мирового уровня и тем самым обеспечили преемственность в математической подготовке новых поколений, так что Россия и сейчас — математическая страна! Во-вторых, СССР первым вышел в космос — это был научно-технологический вызов для США, которые вложили огромные средства в «Лунную программу» и высадили аэронавтов на Луну. А СССР был готов к подобной «Лунной программе» под руководством стратега и идеолога М.В. Келдыша, но отказался «догонять» и ответил своими передовыми достижениями в автоматизации и робототехнике: на Луну запустили первые «Луноходы», а с Луны на Землю прилетели три уникальные ракеты, которые доставили грунт.

«Эпоха Келдыша» — это явление уникальное и исключительное за всю мировую историю в неполной мере отражено в его трудах. После самой разрушительной второй мировой и Великой Отечественной войны произошел невиданный взрыв интеллекта и творческого энтузиазма советского народа. В течение короткой послевоенной жизни М.В. Келдыша в СССР не только совершили три фундаментальных открытия — покорили атом и космос и создали большие математические электронно-вычислительные машины (ЭВМ), но и реализовали «Атомный» и «Космический» проекты и создали «Ракетно-ядерный» щит. При этом характер и научный уровень фундаментальности и технологической глубины этих проектов были таковы, что позволили основать фундамент «цифровой» и «космической» цивилизаций, которые стали реальностью в XXI веке. В истории человечества не было ничего подобного по скорости инноваций и масштабам действия и влияния на всю мировую историю и цивилизацию.

Список литературы

[1] Марчук Г.И., Алдошин С.М., Григорьев А.И., Козлов В.В. Эпоха М.В. Келдыша: выводы и уроки. 17 февраля 2011 г. http://www.ras.ru/news/shownews.aspx?id=6531c71e-d91f-44a2-bd7e-812a1405cffc

[2] Мстислав Всеволодович Келдыш. 100 лет со дня рождения // ИПМ им. М.В. Келдыша РАН. Составители: Езерова Г.Н., Попов Ю.П., Лукичев М.А. Ярославль: ООО Издательство РМП, 2011. 344 с.

[3] Ченцов Н.Н. Всемирно известный, всемерно засекреченный // Наука и жизнь. 1991. № 2. С. 102–107.

[4] Самарский А.А. Прямой расчет мощности взрыва // Труды международного симпозиума ИСАП-96. М.: ИздАТ, 1997. Т.1. С. 214–222. https://keldysh.ru/memory/samarsky/index.htm

[5] Наука и общество: история советского атомного проекта (40-е — 50-е годы) // Труды международного симпозиума ИСАП-96, 14–18 мая 1996 г., Дубна, РНЦ «Курчатовский институт». М.: ИздАТ, 1997. Т.1. 608 с. ISBN 5-86656-073-9 https://elib.biblioatom.ru/text/istoriya-sovetskogo-atomnogo-proekta_t1_1997

[6] Наука и общество: история советского атомного проекта (40-е — 50-е годы) // Труды международного симпозиума ИСАП-96, 14–18 мая 1996 г., Дубна, РНЦ «Курчатовский институт». М.: ИздАТ, 1999–2003. Т.2. 527 с. ISBN 5-86656-090-9 https://elib.biblioatom.ru/text/istoriya-sovetskogo-atomnogo-proekta_t2_1999

[7] Наука и общество: история советского атомного проекта (40-е — 50-е годы) // Труды международного симпозиума ИСАП-96, 14–18 мая 1996 г., Дубна, РНЦ «Курчатовский институт». М.: ИздАТ, 1999–2003. Т.3. 411 с. ISBN 5-86656-146-

8 https://elib.biblioatom.ru/text/istoriya-sovetskogo-atomnogo-proekta_t3_2003

[8] Келдыш М.В. и его институт. Первое двадцатилетие. М.: Мемориальный кабинет-музей академика М.В. Келдыша, 2001. 56 с. https://keldysh.ru/httpd/kiam20.pdf

[9] Казакова Р.К. Точечный взрыв в атмосфере // Прикладная небесная механика и управление движением / Сборник статей, посвященный 90-летию со дня рождения Д.Е. Охоцимского. Сост.: Т.М. Энеев, М.Ю. Овчинников, А.Р. Голиков. М.: ИПМ им. М.В. Келдыша, 2010. С. 107–117. http://keldysh.ru/memory/okhotsimsky/index.htm

[10] Прикладная небесная механика и управление движением / Сборник статей, посвященный 90-летию со дня рождения Д.Е. Охоцимского. Сост.: Т.М. Энеев, М.Ю. Овчинников, А.Р. Голиков. М.: ИПМ им. М.В. Келдыша, 2010. С. 107–117. http://keldysh.ru/memory/okhotsimsky/index.htm

[11] Первое заседание Научно-координационного совета по космосу МГУ. 13.04.2026 («Неделя космоса») https://msu.ru/news/novosti-mgu/pervoe-zasedanie-nauchno-koordinatsionnogo-soveta-po-kosmosu-mgu.html

[12] Келдыш М.В. Речь на XXII съезде КПСС. // Газета «Правда», 1961, 22 нояб. В кн.: М.В. Келдыш. Общие вопросы развития науки. С. 49–55.

[13] Брушлинский К.В. Строитель советской науки. 100 лет со дня рождения Мстислава Келдыша // Наш современник. 2011. № 2. С. 185–190.

[14] Ракетные войска стратегического назначения. Справочник. Библиотека РВСН. Документы. Исторические документы: РВСН и ракетостроение (1945–1967). https://rvsn.info/library_main.html

[15] Советский спутник — первый в мире. Запуску первого искусственного спутника Земли посвящается. Федеральное архивное агентство. https://sputnik.rusarchives.ru/

[16] Лаврентьев М.А. Пути развития советской математики // Изв. АН СССР. Сер. матем. 1948. Том 12, выпуск 4. С. 411–416. https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=im&paperid=3088&option_lang=rus

[17] Международный конгресс математиков // Труды Международного Конгресса математиков, Москва, 1966 г. М.: «Наука» и «Мир», 1968. 726 с. (Келдыш М.В. Речь Президента Академии наук СССР академика М.В. Келдыша на открытии Конгресса. С. 5–6).

[18] Губарев В.С. Мстислав Всеволодович Келдыш. М.: ИД «Комсомольская правда», 2016. 94 с. (Великие умы России 2 кн из 20) ISBN 978-5-4470-0188-9

[19] Губарев В.С. Русский космос (Сверхдержава. Русский прорыв). М.: АЛГОРИТМ, 2006. 464 с.

[20] Губарев В.С. Три звезды Героя: знания и страсти. Несколько страниц из жизни великого ученого нашей Родины М.В. Келдыша // Земля и Вселенная. 2021. № 1. С. 86–100 (начало). https://www.elibrary.ru/item.asp?id=44870432. Земля и Вселенная. 2021. № 2. С. 79–92 (окончание). https://www.elibrary.ru/item.asp?id=44873285

[21] Фултонская речь Уинстона Черчилля в Вестминстерском колледже 5 марта 1946 года / Российское историческое общество. Перевод с английского https://historyrussia.org/index.php?option=com_content&view=article&layout=edit&id=594

[22] История Росатома. Атомный проект СССР. Электронная библиотека http://elib.biblioatom.ru/sections/0201/

[23] Быховский М.Л. Новые американские счетно-аналитические машины // УМН. 1947. № 2. С. 231–234. https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=rm&paperid=6951&option_lang=rus

[24] Келдыш М.В., Ляпунов А.А., Шура-Бура М.Р. Математические вопросы теории счетных машин (Доклад на Сессии АН СССР по научным проблемам автоматизации производства 15–20 октября 1956 г.) // В кн.: М.В. Келдыш. Избранные труды. Математика. М.: Наука, 1985. С. 421–442; http://pyrkov-professor.ru/default.aspx?ArticleId=530&tabid=191

[25] Виртуальный компьютерный музей. https://computer-museum.ru/

[26] Келдыш М.В. Начало космической эры. Речь президента АН СССР академика М.В. Келдыша на Общем собрании АН СССР 19.05.1961 // Вестник АН СССР. 1961. Т. 31. № 6. С. 16–18. https://www.ras.ru/publishing/rasherald/rasherald_articleinfo.aspx?articleid=3ee691e4-8797-427d-91d7-e476ca4a8d32

[27] Келдыш М.В. Страницы памяти (портал Института прикладной математики им. М.В. Келдыша) https://keldysh.ru/memory/keldysh/index.htm

[28] Келдыш М.В. Избранные труды. Математика. М.: Наука, 1985. 448 с. http://pyrkov-professor.ru/default.aspx?ArticleId=530&tabid=191

[29] Келдыш М.В. Избранные труды. Механика. М.: Наука, 1985. 567 с. https://klex.ru/18vf?ysclid=molat3dmpg581972209

[30] Келдыш М.В. Избранные труды. Общие вопросы развития науки. М.: Наука, 1985. 703 с.

[31] Келдыш М.В. Избранные труды. Ракетная техника и космонавтика. М.: Наука, 1988. 493 с.

[32] Келдыш М.В. Творческий портрет по воспоминаниям современников. М.: Наука, 2001. 416 с. https://elib.biblioatom.ru/text/keldysh-tvorcheskiy-portret_2002

[33] Творческое наследие академика Сергея Павловича Королева. Избранные труды и документы / Под ред. М.В. Келдыша. М.: Наука, 1980. 591 с. https://libcats.org/book/482327

[34] Зеленый Л.М., Закутняя О.В. Главный теоретик и стратег отечественной космонавтики // Наука в России. 2011. № 2. С. 42–46. https://elibrary.ru/item.asp?id=16901105

[35] Зеленый Л.М., Закутняя О.В. Научный космос имени Келдыша // Природа. 2011. № 2. С. 43–41. https://iki.cosmos.ru/sites/default/files/popular_article/pdf/priroda2011_02_34-41.pdf

[36] Черток Б.Е. Ракеты и люди (в 4-х книгах). М.: Машиностроение, 1999. https://bookscafe.net/author/chertok_boris-17704.html

[37] Каманин Н.П. Скрытый космос (рукописи «Космические дневники генерала Каманина» в 4-х книгах). https://bookscafe.net/author/kamanin_nikolay-9440.html

[38] Ченцов Н.Н. Всемирно известный, всемерно засекреченный // Наука и жизнь. 1991. № 2. С. 102–107.

[39] Постановление Совета Министров СССР № 149-88с «О создании объекта "Д"». План разработки и изготовления объекта «Д», проведения научно-исследовательских работ и эскизной проработки по объекту «Д» от 30 января 1956 г. Совершенно секретно. Особая папка (рассекречено). https://rvsn.info/library/docs/doc_1_1072.html

[40] Постановление ЦК КПСС и Совета Министров СССР «О развитии исследований по космическому пространству» от 10 декабря 1959 г. № 1388-618. Совершен-

но секретно особой важности (рассекречено). https://www.kosmonavtika.com/bibliographie/documents/1388-618.pdf

[41] Советская космическая инициатива в государственных документах. 1946–1964 гг. / Под ред. Ю.М. Батурина. М.: Изд-во «РТСофт», 2008. 416 с. http://www.coldwar.ru/arms_race/iniciativa/

[42] Постановление Совета Министров СССР «Об утверждении Положения о Междуведомственном научно-техническом совете по космическим исследованиям при Академии наук СССР» от 24 сентября 1960 г. № 1026-421 секретно (рассекречено). http://www.coldwar.ru/arms_race/iniciativa/ob-utverzhdenii-polozheniya.php

[43] Постановление Президиума ЦК «О запуске космического корабля-спутника (с космонавтом на борту)». 3 апреля 1961 г. Выписка из протокола № 322 заседания Президиума ЦК от 3 апреля 1961 г. https://unikdoc.rusarchives.ru/0874-2022-postanovlenie-prezidiuma-ck-kpss-o-zapuske-kosmicheskogo-korablya-sputnika

[44] Задача особой государственной важности. Из истории создания ракетно-ядерного оружия и Ракетных войск стратегического назначения (1945–1959 гг.) / Сборник документов. Сост. В.И. Ивкин, Г.А. Сухина. М.: Российская политическая энциклопедия (РОССПЭН), 2010. 1205 с. https://djvu.online/file/gh7kDzgQGCef5?ysclid=lpns9mw8hp143834036

[45] Попов Ю.П., Боровин Г.К., Сушкевич Т.А. Институту прикладной математики имени М.В. Келдыша Российской академии наук — 50 лет // Вестник РФФИ. 2004, март, № 1 (35). С. 67–72. https://elibrary.ru/item.asp?id=17905955

[46] 4 октября 1957 года начало космической эры. Первая космическая / Сборник статей, посвященных пятидесятилетнему юбилею запуска Первого искусственного спутника Земли. М.: ИКИ РАН, «Регион Инвест», 2007. 335 с. http://www.iki.rssi.ru/books/2007pervaya_r.pdf

[47] Освоение космического пространства в СССР. Официальные сообщения ТАСС и материалы центральной печати 1957–1967 гг. // ИКИ АН СССР. Отв. ред. Скуридин Г.А. М.: Издательство «Наука», 1971. 555 с. https://epizodsspace.airbase.ru/bibl/osvoen-kosm-pr-sssr/1957-1967/01.html

[48] Освоение космического пространства в СССР. Официальные сообщения ТАСС и материалы центральной печати октябрь 1967– 1970 гг. // ИКИ АН СССР. Отв. ред. Петров Г.И. М.: Издательство «Наука», 1971. 360 с. https://epizodsspace.airbase.ru/bibl/osvoen-kosm-pr-sssr/1968-1970/01.html

[49] Освоение космического пространства в СССР. По материалам центральной печати 1971 г. // ИКИ АН СССР. Отв. ред. Нариманов Г.С. М.: Издательство «Наука», 1973. 302 с. https://epizodsspace.airbase.ru/bibl/osvoen-kosm-pr-sssr/1971/01.html

[50] Гагарин Ю.А. Дорога в космос. Записки летчика-космонавта СССР / Лит. запись специальных корреспондентов «Правды» Н. Денисова и С. Борзенко. Под ред. и с пред. генерал-лейтенанта авиации Н. Каманина. М.: Правда, 1961; М.: Воениздат, 1981. 336 с.

[51] Титов Г.С. 700000 километров в космосе. Рассказ летчика-космонавта СССР / Герой Советского Союза Герман Титов (Лит. запись С. Борзенко и др.). М.: Правда, 1961. 142 с.

[52] Первая космическая съемка. Герман Титов. https://www.roscosmos.ru/31701/

[53] Указ Президиума ВС СССР о присвоении звания Героя Социалистического Труда Л.И. Брежневу. № 253/28. 17 июня 1961 г. https://docs.historyrussia.org/ru/nodes/481499

[54] Указ Президиума ВС СССР о присвоении звания Героя Социалистического Труда ученым, руководящим деятелям, рабочим, конструкторам, инженерам и техникам. № 253/29. 17 июня 1961 г. https://docs.historyrussia.org/ru/nodes/481500

[55] Указ Президиума Верховного Совета СССР О награждении Героев Социалистического Труда В.П. Глушко, М.В. Келдыша, С.П. Королева, В.И. Кузнецова, Н.А. Пилюгина, Д.Ф. Устинова и М.К. Янгеля второй золотой медалью «Серп и Молот». № 253/27. 17 июня 1961 г. // ГА РФ. Ф. Р-7523. Оп. 67а. Д. 39. Л. 55, 56. https://docs.historyrussia.org/ru/nodes/481498

[56] Первый пилотируемый полет. Российская космонавтика в архивных документах. В 2-х книгах. Под ред. В.А. Давыдова. Федеральное космическое агентство. М.: Издательство «Родина МЕДИА», 2011. Книга 1 — 560 с. https://djvu.online/file/npXvqXn7zV0PS-CЊPчCЂPёCЂCЖ; Книга 2 — 560 с. https://djvu.online/file/c5OWKbCfqjM34

[57] Полет по околоземной орбите космического корабля «Восток» с нашим соотечественником Ю.А. Гагариным на борту ознаменовал начало эпохи пилотируемых космических полетов. Роскосмос. https://www.roscosmos.ru/28333/

Аннотации стендовых докладов

Parallel computation of density maps in molecular dynamics based on Voronoi tesselation*

R.H. Locon[1], V.V. Pisarev[1]

HSE University, Moscow, Russia[1]

Density maps are useful in such areas of molecular simulation as binding simulations, cavity finding or fluid-surface interaction analysis. The typical ways of obtaining such maps are simple spatial binning and use of gaussian kernels for atomic contributions. As an alternative to gaussian kernels which are model-dependent, smearing a particle contributions over their Voronoi cells can be used. This work implements spatial binning combined with Voronoi cell contributions. Instead of simple binning of point particles, the particle contributes to multiple bins proportional to the fraction of its Voronoi cell volume within each bin. Calculations over Voronoi cells can be performed in parallel, giving the possibility to speed up computations on multi-processor systems. For our purposes, particle's information comes from simulations using LAMMPS package [1] and then they are processed with MDProcessing.jl [2] and VoroPlusPlus.jl tools in Julia language. The former is a generic framework for working with atomic simulation trajectories and the latter is a wrapper for the Voro++ library [3] exposing its functionality in Julia and adding domain-decomposition parallelism. Both software packages are developed by SAMMA laboratory at MIEM/HSE university for processing atomistic simulations results.

1. Introduction

In Molecular Dynamics simulations, formulating a good approximation of density distribution of particles is a challenging task. Density maps is a wide used tool to analyze particle distributions in fluids and distributions around solids. Those density maps are often computed on molecular simulations and one of the challenges is to compute density maps efficiently, using small number of molecular dynamics snapshots and of course to compute them at high spatial resolution. Smooth density map that shows an accurate and real representation of particle density distribution are intended to be an effective way of computation. We analyze the performance of Voronoi tessellation approach for density map calculation in Molecular Dynamics, also we use MDProcessing.jl for trajectories, VoroPlusPlus.jl for Voronoi tessellation and finally we implement parallelization (domain decomposition) through Julia threads.

2. Procedures

We can mention various techniques often used to calculate density maps including: simple spatial binning and density kernels for different types of atoms. Both techniques are not parameter-free. Simple binning, for example, suffers from large dispersion in small subsets. Obtaining a smooth density map requires either using a coarse grid so that each grid cell (representative volume element) contains a sufficiently large number of particles or averaging over a long simulation. On the other hand, with density kernels the map is

*The work is prepared within the framework of the HSE University Basic Research Program

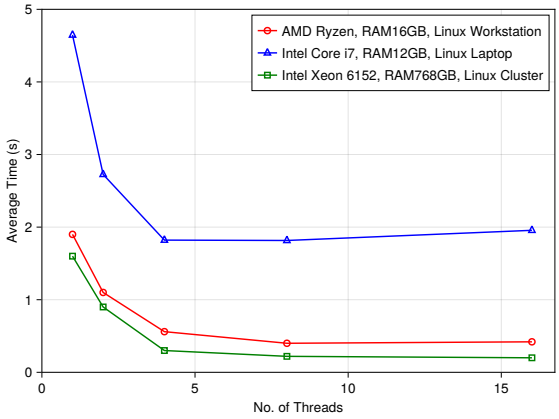


Fig. 1. Timings to compute a 100x100x100 grid map with a system of 31744 particles implementing the parallel Voronoi tessellation approach.

smooth by construction getting a bell-shape kernel, but the method is non-parameter free, meaning that it depends on kernel width for each atom [4]. In this work, we present and test a technique based on Voronoi tessellation, that combines the strengths of density kernel approach with Voronoi cell-besed kernel shape which adapts to the local environment of each particle. Compared to simple spatial binning, the Voronoi tessellation approach provides a smoother density map irrespective of the grid resolution. To speed up the computation, simulation cell is subdivided into subdomains and Voronoi tessellation is calculated using a thread structure from Julia language.

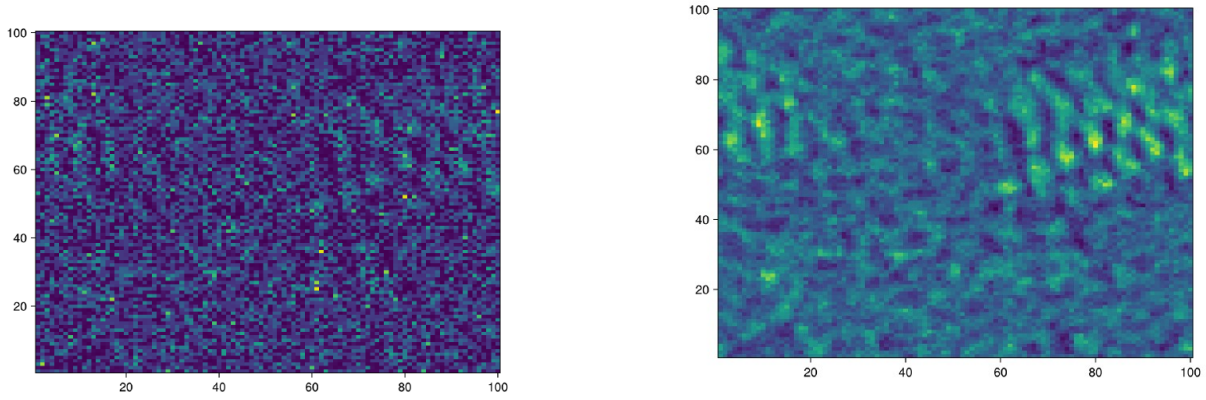


Fig. 2. Slices of density maps for an n-eicosane system. The map is constructed using a $100 \times 100 \times 100$ grid with a system of 31744 particles. (a) Simple binning technique; (b) Voronoi tessellation technique.

Список литературы

[1] Thompson A. P. et al. LAMMPS — a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales // Computer physics communications. 2022. Vol. 271. P. 108171. DOI: hrefhttps://doi.org/10.1016/j.cpc.2021.1081710.1016/j.cpc.2021.10817.

[2] Pisarev V., Panov M. MDProcessing.jl: Julia Programming Language Application for Molecular Dynamics Trajectory Processing // Russian Supercomputing Days. Cham : Springer Nature Switzerland, 2023. P. 209–222. DOI: hrefhttps://doi.org/10.1007/978-3-031-49435-2_1510.1007/978-3-031-49435-2_15.

[3] Rycroft C. H. Voro++: A three-dimensional Voronoi cell library in C++ // Chaos: An Interdisciplinary Journal of Nonlinear Science. 2019. Vol. 19. P. 041111. DOI: hrefhttps://doi.org/10.1063/1.321572210.1063/1.3215722.

[4] Ganthaler E., Mustafa S., Peer A. Comparison of different methodologies for estimating local density in particle-based simulations // Computational Particle Mechanics. 2025. Vol. 12. P. 1217–1231. DOI: hrefhttps://doi.org/10.1007/s40571-024-00870-410.1007/s40571-024-00870-4.

Векторная библиотека математических функций для процессоров архитектуры RISC-V*

В.А. Кабалова, А.А. Воденеева, Е.А. Козинов, Е.В. Коротин, А.В. Линев, Е.А. Панова, Д.И. Суворов, А.В. Сысоев, В.Д. Волокитин, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

Математические функции являются фундаментом для исследований в физике, астрономии и других научных областях, поэтому уже давно были разработаны эталонные скалярные математические библиотеки, позволяющие точно и быстро производить необходимые вычисления. Компиляторы содержат собственные реализации, которые используются в программах в расчете на их точность, скорость и переносимость. Например, такой библиотекой является glibc libm. При этом эталонной по точности библиотекой является CRlibm (Correctly Rounded Mathematical Library), обеспечивающая правильное округление в любой точке и соответствующая стандарту операций с плавающей запятой IEEE-754 [1]. К сожалению, существующие скалярные реализации не всегда позволяют максимально эффективно использовать вычислительные мощности современных процессоров. Достижение наилучшей производительности невозможно без рационального использования инструкций SIMD (Single Instruction, Multiple Data). Оптимизирующие компиляторы в ряде случаев позволяют преобразовать скалярный код в векторный, однако это не всегда возможно. Для эффективной векторизации вычислительного цикла с использованием математических функций компиляторы подставляют их готовые векторные реализации. По этой причине вендоры различных аппаратных решений ведут собственные разработки математических библиотек, направленные на максимальную производительность именно на их чипах. Компания Intel, в свою очередь, разработала специальную библиотеку математических функций SVML (Short Vector Math Library), которая используется для автовекторизации в различных компиляторах. AMD, ARM и NVIDIA также поддерживают аналогичные библиотеки для своих процессоров – AMD AOCL-LibM, ARM Optimized Routines и CUDA Math, соответственно. Подобные библиотеки часто могут жертвовать точностью для достижения максимальной производительности [2], что приемлемо для многих задач. В целом, поиск компромисса между точностью и производительностью является одной из проблем при разработке векторных математических библиотек для разных вычислительных архитектур.

С 2015 развивается открытая и свободная от патентных ограничений архитектура RISC-V, на ее основе разрабатываются процессоры и соответствующее системное ПО. В 2021 году произошла ратификация векторного расширения RVV 1.0, что зафиксировало стандарт для разработчиков и производителей чипов. Основной отличительной особенностью расширения RVV по сравнению с привычным AVX является определение длины регистра в ходе выполнения программы, также можно задать количество элементов в регистре и использовать специальный параметр LMUL (Length Multiplier), позволяющий объединять обработку нескольких векторных регистров в одну инструкцию. За последние годы появилась поддержка автовекторизации в компиляторах GCC и LLVM. Тем не менее, до сих пор отсутствует высокопроизводительная библиотека

*Авторы благодарят ННГУ им. Н. И. Лобачевского за предоставление вычислительных ресурсов.

математических функций, повсеместно интегрированная в процесс автовекторизации в компиляторах для архитектуры RISC-V. Поэтому сохраняется необходимость создания математических библиотек специально на RVV 1.0 и их низкоуровневая оптимизация. Подобные разработки ведутся уже несколько лет, и в 2024 году была добавлена поддержка архитектуры RISC-V в кроссплатформенную библиотеку математических функций SLEEF. Она написана на универсальных интринсиках, и в ней есть реализации только для LMUL равным 1 и 2 для чисел одинарной и двойной точности. Также существует библиотека veclibm, написанная с помощью RVV интринсиков. Она обладает высокой производительностью, но реализована только для чисел двойной точности, и оптимизирована для работы на массивах, а не регистрах.

Ранее был опубликован алгоритм и результаты тестирования векторной функции экспонента [3]. В данной работе мы представляем разработку высокоточной и производительной библиотеки математических функций RVVMF [4] для архитектуры RISC-V с использованием интринсиков RVV 1.0. Реализации должны соответствовать стандарту операций чисел с плавающей запятой и требованию 0.501 ULP на всей области определения функций. В первую очередь, ULP (Unit in the Last Place) означает расстояние между двумя представимыми числами с плавающей запятой и используется для описания ошибок округления. Мы же используем целевой показатель 0.501 ULP – это условное статистическое допущение, которое подразумевает, что подавляющее большинство результатов округляется математически точно (0.5 ULP), но в одном из тысячи случаев допустима ошибка округления. Это является оптимальным компромиссом между точностью и производительностью. Разработка математических функций ведется в форматах, широко распространенных на обычных чипах: двойной, одинарной и половинной точности. Поддержка последней является актуальной тенденцией, позволяющей ускорить вычисления в различных областях, например, в задачах искусственного интеллекта. Кроме того, выполняется низкоуровневая оптимизация всех математических функций для достижения максимальной производительности.

Эксперименты проводились на плате Banana Pi BPI-F3 с процессором SpacemiT K1 (in-order, RVV 1.0, длина векторного регистра 256 бит) с использованием кросс-компилятора GCC 14.2.0. Для тестирования вычислялся массив значений функции по массиву аргументов. Аргументы для тестов выбирались случайным образом из диапазона допустимых значений, а результаты замеров усреднялись по выборке.

RVVMF показывает точность, сравнимую с glibc libm, обеспечивая при этом лучшую производительность за счет векторной реализации (рис. 1). Представленные

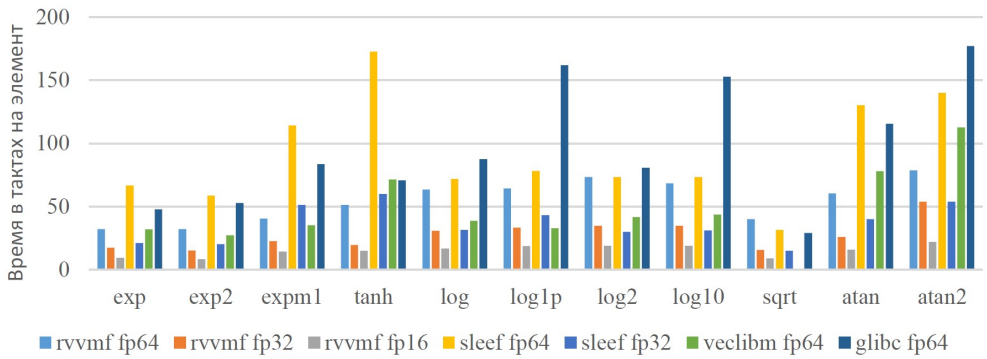


Рис. 1. Результаты замеров производительности математических функций для библиотек RVVMF, SLEEF, veclibm и glibc.

реализации функций также демонстрируют производительность не хуже, чем у библиотеки SLEEF, при значительно лучшей точности. Библиотека veclibm в настоящее время является эталоном по скорости работы и в основном обгоняет RVVMF, однако стоит учитывать ее изначальную архитектурную оптимизацию под работу с массивами. Ситуация с производительностью может измениться при запусках на процессорах с внеочередным (out-of-order) выполнением инструкций.

В дальнейшем планируется завершить разработку остальных математических функций из модуля math.h, а также оптимизировать их производительность под архитектуру RISC-V.

Список литературы

[1] IEEE Standard for Floating-Point Arithmetic: IEEE Std 754-2019 [Электронный ресурс]. 2019. URL: https://ieeexplore.ieee.org/document/8766229 (дата обращения: 13.04.2026).

[2] Innocente V., Zimmermann P. Accuracy of mathematical functions in single, double, double extended, and quadruple precision [Электронный ресурс]. 2026. URL: https://members.loria.fr/PZimmermann/papers/accuracy.pdf (дата обращения: 13.04.2026).

[3] Панова Е.А. и др. Высокопроизводительная реализация экспоненциальных функций для процессоров архитектуры RISC-V // Вычислительные методы и программирование. 2026. Т. 27. С. 27-45. DOI: 10.26089/NumMet.v27r103.

[4] Библиотека RVVMF [Электронный ресурс]. 2026. URL: https://github.com/rvvpl/rvvmf (дата обращения: 13.04.2026).

Динамическое ядро атмосферы на сетке кубическая сфера: переход к вычислениям на GPU*

Д.А. Марханов[1,2], В.В. Шашкин[1,2,3], Г.С. Гойман[1,2,3]

[1]Институт вычислительной математики им. Г.И. Марчука РАН
[2]Гидрометеорологический научно-исследовательский центр РФ
[3]Московский физико-технический институт

Типичное пространственное разрешение горизонтальных сеток глобальных моделей атмосферы в вычислительных экспериментах, связанных с моделированием климата, CMIP6 и HighResMIP — величина около 100 км и 10 км, соответственно [1]. На подобных сетках множество важных атмосферных процессов (например, глубокая конвекция) не разрешаются, их эффект описывается с помощью приближенных эмпирико-статистических моделей (параметризаций). При переходе к горизонтальным сеткам с шагом 3-5 км и мельче влияние параметризаций в значительной степени ослабнет, так как существенная часть процессов будет разрешаться явно [2, 3]. Добиться этого эффекта позволяет не только увеличение разрешения, но и переход от гидростатических уравнений динамики к негидростатическим [3], которые являются более вычислительно сложными. Подобная детализация выдвигает чрезвычайно высокие требования к вычислительным ресурсам и оценивается на уровне $\sim 10^5$ ядер центральных процессоров (CPU) [4]. Эффективная альтернатива суперкомпьютерам на основе классических процессоров — кластера, использующие графические ускорители. Фактически, использование GPU рассматривается как единственная возможность моделирования климата с высоким разрешением, а также, как способ существенно повысить скорость вычислений стандартных экспериментов.

К числу наиболее показательным и передовым примерам, которые представляют особый интерес, можно отнести модели SCREAM [5], CliMA [6] и ICON [7]. Эти модели используют негидростатические уравнения динамики атмосферы и высокое горизонтальное разрешение для моделирования существенной части мезомасштабных и конвективных процессов, за что были охарактеризованы как cloud-resolving, а их вычисления адаптированы под инновационные GPU кластера. Модели демонстрируют качественное улучшение результатов моделирования, а также кардинальный прирост эффективности при расчётах на графических ускорителях. Таким образом, они отражают общий сдвиг в развитии атмосферного моделирования в сторону cloud-resolving и GPU.

В настоящий момент в институте ИВМ РАН разрабатывается модель атмосферы мезомасштабного разрешения (3-5 км) на сетке кубическая сфера [8]. Преимуществом такой сетки являются квази-равномерное разрешение на сфере, позволяющее устранить проблему сингулярности в окрестности полюсов Земли, и естественная логически-прямоугольная структура на гранях куба, которая хорошо соответствует параллельной архитектуре GPU, что делает её перспективной для переноса вычислений на графические ускорители. Однако, использование GPU требует специфическую адаптацию алгоритмов и кодовой базы модели.

*Поддержана Отделением Московского центра фундаментальной и прикладной математики в ИВМ РАН (Соглашение с Минобрнауки России № 075-15-2025-347).

Данная работа является продолжением по адаптации под GPU динамического ядра атмосферы и посвящена алгоритмам численного решения негидростатических уравнений динамики атмосферы на сетке кубическая сфера. Задача заключалась в эффективном переносе на графические ускорители вычислительных компонент, лежащих в основе при расчёте численного решения динамических уравнений. Для этой цели взят за основу стек технологий компании NVIDIA (графические процессоры и программно-аппаратная архитектура CUDA), а в качестве главного инструмента для реализации вычислений выбран программный язык CUDA-C, позволяющий производить расчёты на видеокартах от NVIDIA, а также допускающий взаимодействие с исходным кодом на Fortran.

В ходе адаптации была разработана библиотека CUDALIB на языке CUDA-C, содержащая в себе все компоненты, необходимые для расчёта численного решения негидростатических уравнений динамики атмосферы на сетке кубическая сфера. Библиотека CUDALIB прошла двухстадийную верификацию. Во-первых, все вычислительные GPU-функции проверялись на корректность путём сравнения с эталонными результатами CPU-версии кода с точностью до отклонений, обусловленных погрешностями округления в арифметике с плавающей точкой. Во-вторых, по аналогии, сравнивались бинарные выходные данные негидростатической модели, полученные при вычислениях на CPU и GPU. Перед запуском модели проводилась инициализация начальными данными из де-факто стандартных численных экспериментов, разработанных для валидации динамического ядра модели атмосферы: “Обтекание горы Шара” [9], “Бароклинная неустойчивость” [10], “Тест Хелда-Суареза” [11]. Модель воспроизводит классические решения в любом режиме работы.

Разработанная библиотека интегрирована в программный код модели и позволяет производить вычисления на кластере, обладающим множественным количеством GPU. Библиотека также поддерживает машинную арифметику в одинарной и двойной точностях.

В рамках работы будут представлены основные сложности реализации, а также исследования эффективности и масштабируемости на одном вычислительном GPU-узле: сравнение CPU/GPU, сильная и слабая масштабируемость, сравнение времени вычислений в одинарной и двойной точностях.

Список литературы

[1] Haarsma R. J. et al. High resolution model intercomparison project (HighResMIP v1. 0) for CMIP6 //Geoscientific Model Development. – 2016. – Т. 9. – №. 11. – С. 4185-4208.

[2] Satoh M. et al. Toward reduction of the uncertainties in climate sensitivity due to cloud processes using a global non-hydrostatic atmospheric model //Progress in Earth and Planetary Science. – 2018. – Т. 5. – №. 1. – С. 1-29.

[3] Zeman C. et al. Model intercomparison of COSMO 5.0 and IFS 45r1 at kilometer-scale grid spacing //Geoscientific Model Development. – 2021. – Т. 14. – №. 7. – С. 4617-4639.

[4] Terai C. R. et al. Climate Response to Warming in Cess-Potter Simulations Using the Global 3-km SCREAM //Authorea Preprints. – 2025.

[5] Taylor M. et al. The simple cloud-resolving e3sm atmosphere model running on the frontier exascale system //Proceedings of the international conference for high performance computing, networking, storage and analysis. – 2023. – С. 1-11.

[6] Yatunin D. et al. The climate modeling alliance atmosphere dynamical core: Concepts, numerics, and scaling //Authorea Preprints. – 2025.

[7] Lapillonne X. et al. Operational numerical weather prediction with ICON on GPUs

(version 2024.10) //EGUsphere. – 2025. – Т. 2025. – С. 1-25.

[8] Shashkin V. V., Goyman G. S., Tretyak I. D. Development of the next-generation atmosphere dynamics model in Russia: Current state and prospects //Lobachevskii Journal of Mathematics. – 2024. – Т. 45. – №. 7. – С. 3159-3172.

[9] Jablonowski C. et al. Idealized test cases for the dynamical cores of Atmospheric General Circulation Models: A proposal for the NCAR ASP 2008 summer colloquium //DCMIP-2008_TestCaseDocument_29May2008. pdf. – 2008.

[10] Jablonowski C., Williamson D. L. A baroclinic instability test case for atmospheric model dynamical cores //Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography. – 2006. – Т. 132. – №. 621C. – С. 2943-2975.

[11] Held I. M., Suarez M. J. A proposal for the intercomparison of the dynamical cores of atmospheric general circulation models //Bulletin of the American Meteorological society. – 1994. – Т. 75. – №. 10. – С. 1825-1830.

Квантово-классическая бинарная классификация изображений: эксперименты на квантовом компьютере*

А.Д. Ивлев[1], А.В. Линев[1], М.В. Бастракова[1], М.В. Иванченко[1]

Нижегородский государственный университет им. Н.И. Лобачевского[1]

Введение. Рассматривается задача анализа наличия дефектов в магнитных плитках с помощью гибридных квантово-классических нейронных сетей [1]. Приведены результаты исследования характеристик работы алгоритма бинарной классификации при использовании симуляторов квантового компьютера и квантового компьютера МГТУ им. Н.Э. Баумана [2].

Структура гибридной квантовой нейронной сети и квантового слоя. Для решения задачи бинарной классификации (наличие/отсутствие дефекта) используются два варианта гибридной квантово-классической нейронной сети: первая получена из классической свёрточной нейронной сети добавлением в конец одного квантового слоя и одного линейного; вторая – добавлением к первой линейного классического слоя параллельно квантовому. Подробное описание подходов приведены в статье [1], продолжением которой является данная работа. Параметры двух последних слоев нейронных сетей представлены в Таблице 1.

Таблица 1. Типы и параметры двух последних уровней нейронной сети

Гибридная сеть			Гибридная сеть с параллельными слоями		
Тип слоя	Размерность выхода	Количество параметров	Тип слоя	Размерность выхода	Количество параметров
Квантовый	4	12	Квантовый и параллельный линейный	4/4	12/12
Линейный	2	10	Линейный	2	18

Классические данные загружаются в квантовую часть угловым кодированием с помощью гейта Ry. Квантовый анзац представляет собой применение параметризованных гейтов Ry ко всем кубитам с запутыванием при помощи гейтов CNOT. Итоговая схема изображена на Рисунке 1, её глубина – 12, количество однокубитных операций – 14, а двухкубитных – 5.

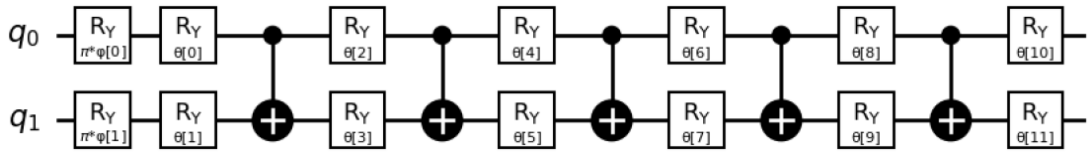


Рис. 1. Схема квантового слоя для бинарной классификации.

*Исследование выполнено при поддержке программы развития региональных научно-образовательных математических центров, соглашение Минобрнауки № 075-02-2026-1346.

Для проведения экспериментов на квантовом компьютере Snowdrop 4Q ver2 и его шумном симуляторе необходимо преобразовать схему с учётом топологии, базисных операций и параметров кубитов. Использовались два подхода, обеспечивающие переносимость схем на квантовые устройства. Первый, классический подход, заключается в однократном преобразовании и оптимизации параметризованной схемы перед проведением экспериментов. Полученная схема представлена на Рисунке 2, её глубина – 22, число однокубитных операций – 24, двухкубитных – 5.

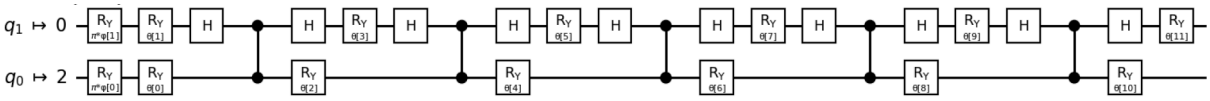


Рис. 2. Параметризованная схема, преобразованная для квантового компьютера Snowdrop 4Q ver2.

Второй подход, предлагаемый нами, заключается в преобразовании и оптимизации квантовых схем перед каждым запуском после подстановки в неё значений всех параметров. Это позволяет уменьшить их размер и глубину при предварительной классической обработке.

Методика проведения эксперимента. Программная реализация выполнена на языке Python и использует библиотеки Pytorch, Qiskit, qiskit_machine_learning и ocillation_client. Квантовые схемы запускались с 2048 измерениями на симуляторе идеального квантового компьютера AerSimulator, квантовом компьютере Snowdrop 4Q ver2 и его шумном симуляторе. Преобразование и оптимизация квантовых схем проводилась с помощью qiskit.transpiler.preset_passmanagers. Тестовая выборка состоит из 168 изображений, из которых 49 содержат дефекты. Функция потерь – взвешенная torch.nn.CrossEntropyLoss.

Результаты экспериментов. Метрики: среднее значение функции потерь и точность.

Таблица 2. Результаты классификации при первом подходе к преобразованию квантовой схемы и эталонные значения, полученные на симуляторе идеального квантового компьютера

Тип исполнительного устройства для квантовой схемы	Гибридная сеть		Гибридная сеть с параллельными слоями	
	Среднее значение функции потерь	Точность	Среднее значение функции потерь	Точность
Идеальный симулятор	0.496270	0.982143	0.369395	0.946429
Шумный симулятор	0.519822	0.982143	0.371692	0.946429
Квантовый компьютер	0.528427	0.982143	0.392241	0.946429

При использовании первого подхода на шумном симуляторе и квантовом компьютере немного увеличилось среднее значение функции потерь, но точность не изменилась, что отображено в Таблице 2 и показывает возможность использования квантово-классических нейронных сетей на практике с использованием современных квантовых компьютеров.

При использовании второго подхода точность также не изменилась, но среднее значение функции потерь снизилось из-за меньших размеров исполняемых схем и ста-

Таблица 3. Результаты классификации при втором подходе к преобразованию квантовой схемы

Тип исполнительного устройства для квантовой схемы	Гибридная сеть		Гибридная сеть с параллельным слоем	
	Среднее значение функции потерь	Точность	Среднее значение функции потерь	Точность
Шумный симулятор	0.508149	0.982143	0.370719	0.946429
Квантовый компьютер	0.521403	0.982143	0.386572	0.946429

ло ближе к эталонным. Результаты представлены в Таблице 3. Средние значения параметров для квантовых схем меньше, чем для первого подхода и даже исходной схемы: глубина – 10.773, количество однокубитных операций – 11.523, двухкубитных – 3. Пример одной из полученных схем представлен на Рисунке 3. При этом среднее время, затраченное на преобразование и оптимизацию, составило 0.216 с, что значительно меньше, чем среднее время исполнения на квантовом компьютере – 16.6 с (с учётом накладных расходов на удалённый доступ).

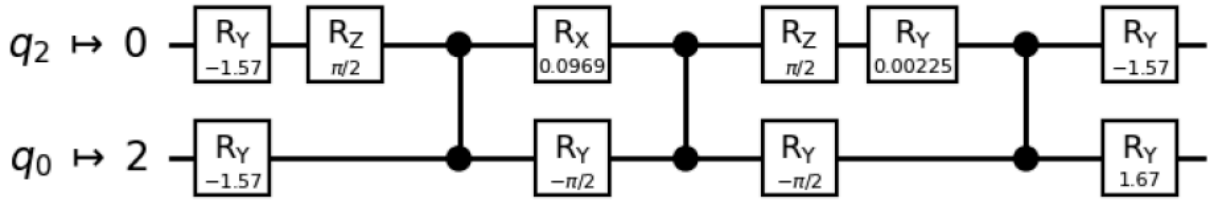


Рис. 3. Пример квантовой схемы, преобразованной и оптимизированной для квантового компьютера Snowdrop 4Q ver2 после подстановки параметров.

Результаты показывают возможность применения гибридных квантово-классических нейронных сетей с использованием современных квантовых компьютеров эпохи NISQ для решения задач классификации и целесообразность проведения дальнейших исследований в данном направлении.

Список литературы

[1] A. D. Ivlev, A. V. Liniov, M. V. Bastrakova, V. D. Kustikova, and I. B. Meyerov. Hybrid Quantum-Classical Neural Networks for Analyzing Magnetic Tile Images for Defects // Lobachevskii Journal of Mathematics, 2026, Vol. 47, No. 1, P. 190–199.

[2] Облачная платформа Bauman Octillion. URL: https://quantum.sever.bmstu.ru (дата обращения: 26.04.2026).

Моделирование динамических воздействий на коленный сустав человека с использованием многоблочных и наложенных сеток и параллельных вычислений*

Н.А. Волков[1], Н.И. Хохлов[1,2], И.Б. Петров[1]

Московский физико-технический институт (национальный исследовательский университет)[1], Федеральное государственное автономное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт»[2]

Современная медицина опирается на изучение биомеханических процессов, лежащих в основе диагностики и лечения травм и патологий. Клинический опыт в настоящее время активно дополняется методами вычислительного моделирования, развитие которых позволяет создавать точные модели сложных биологических систем. В связи с этим растёт интерес к численному анализу функционирования коленного сустава.

Коленный сустав является одним из наиболее нагруженных и уязвимых элементов опорно-двигательной системы, что определяет его высокую подверженность травмам. Разработка эффективных методов профилактики и лечения требует детального понимания его механики и механизмов повреждений как в интактном состоянии, так и после хирургических вмешательств.

Для анализа напряженно-деформированного состояния коленного сустава при статических и квазистатических нагрузках традиционно применяется метод конечных элементов [1]. Это позволяет выявлять и объяснять механизмы возникновения повреждений. Вместе с тем возможности подхода ограничены при исследовании кратковременных высокоинтенсивных воздействий. В связи с этим целью работы является разработка двумерной модели коленного сустава для анализа его отклика на динамические нагрузки.

Модель основана на системе уравнений линейной теории упругости [2]. Для решения используется сеточно-характеристический метод (схема Русанова 3 порядка точности) на структурированных сетках с расщеплением по пространственным направлениям. Была разработана двумерная осесимметричная модель коленного сустава [3], включающая бедренную и большеберцовую кости, суставные хрящи и мениск (рис. 1, а). Все компоненты рассматриваются как изотропные упругие среды с физико-механические свойствами (плотность, модуль Юнга, коэффициент Пуассона), заданными на основе литературных данных [1] и приведенными в табл. 1. Граничные и контактные условия [2] схематично показаны на рис. 1, а.

Для построения расчетной сетки использовались многоблочный подход и метод наложенных сеток [4]. Каждая анатомическая структура описывается отдельным набором сеток, для их согласования применяется процедура сшивки с использованием интерполяционного алгоритма обмена информацией с привлечением фиктивных

*Работа выполнена при финансовой поддержке Российского научного фонда (номер проекта 25-71-10027).

(ghost) узлов. Глобальное решение формируется на основе объединения значений, полученных на отдельных сетках.

Высокие требования к пространственно-временному разрешению приводят к необходимости большого числа шагов по времени. С целью повышения вычислительной эффективности предлагается использование параллельных вычислений на основе технологии MPI. Применяется специальная стратегия декомпозиции сеток по процессам [5], обеспечивающая равномерное распределение вычислительной нагрузки. Результат работы алгоритма показан на рис. 1, б.

Динамическое нагружение моделировалось в виде короткого импульса давления на нижней границе бедренной кости. Расчеты проводились на вычислительной системе с двумя процессорами AMD EPYC 7663. Результаты моделирования и график ускорения приведены на рис. 2.

В дальнейшем планируется учесть вязкоупругие свойства, разработать трехмерную постановку задачи и провести масштабные исследования производительности на детализированных расчетных сетках с использованием большего числа параллельных процессов.

Таблица 1. Механические характеристики материалов

Компонент	**Модуль Юнга, МПа**	**Коэффициент Пуассона**	**Плотность, кг/м^3**
Бедренная кость	20000	0,220	2000
Берцовая кость	18400	0,120	2000
Хрящ	15	0,475	1000
Мениск	20	0,450	1500

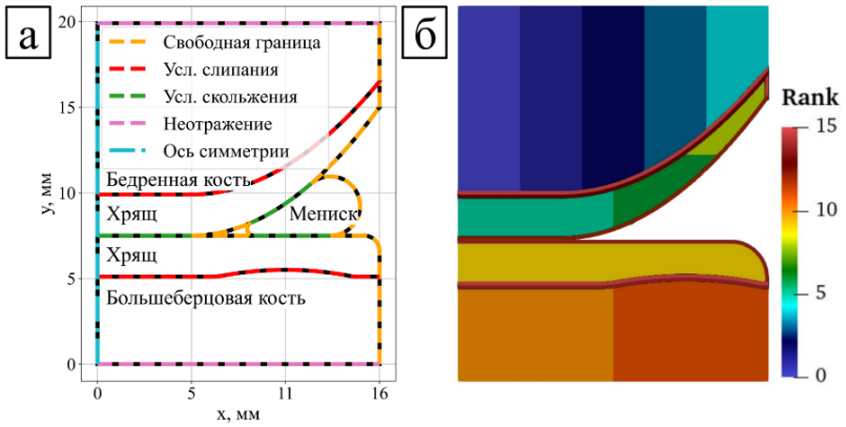


Рис. 1. Двумерная осесимметричная модель коленного сустава с граничными и контактными условиями (а); Декомпозиция расчетной сетки по MPI процессам (б)

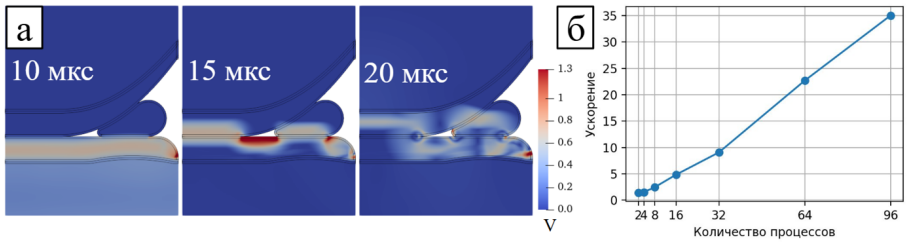


Рис. 2. Поле скоростей в коленном суставе в последовательные моменты времени (а); График ускорения в зависимости от количества MPI процессов (б)

Список литературы

[1] Trad Z., Barkaoui A., Chafra M. et al. FEM Analysis of the Human Knee Joint A Review. Cham: Springer, 2018. 79 p. DOI: 10.1007/978-3-319-74158-1.

[2] Favorskaya A. V., Zhdanov M. S., Khokhlov N. I. et al. Modelling the wave phenomena in acoustic and elastic media with sharp variations of physical properties using the grid-characteristic method // Geophys. Prospect. 2018. Vol. 66, No. 8. P. 1485–1502. DOI: 10.1111/1365-2478.12639.

[3] Vaziri A., Nayeb-Hashemi H., Singh A. et al. Influence of Meniscectomy and Meniscus Re-placement on the Stress Distribution in Human Knee Joint // Ann Biomed Eng. 2008. Vol. 36, No. 8. P. 1335–1344. DOI: 10.1007/s10439-008-9515-y.

[4] Favorskaya A. V., Khokhlov N. I. Using Chimera Grids to Describe Boundaries of Complex Shape // Smart Innov. Syst. Technol. 2022. Vol. 309. P. 249–258. DOI: 10.1007/978-981-19-3444-5_22.

[5] Agrelov I.N., Khokhlov N.I., Stetsyuk V.O. et al. On an Algorithm for Decomposing Multi-Block Structured Meshes for Calculating Dynamic Wave Processes in Complex Structures on Supercomputers with Distributed Memory // Supercomputing Frontiers and Innovations, 2025. Vol. 11, No. 4. P. 54–65. DOI:10.14529/jsfi240405.

Новый формат хранения разреженных матриц*

М.А. Загрядсков, В.Д. Волокитин

Нижегородский государственный университет им. Н.И. Лобачевского

Разреженные матрицы широко используются во многих прикладных вычислительных задачах. В свою очередь, умножение матрицы на вектор (SpMV, $y := \alpha Ax + \beta y$) является одной из ключевых операций с такими матрицами, поэтому от его эффективной реализации зависит производительность многих алгоритмов линейной алгебры, в частности, итерационных решателей разреженных СЛАУ. Известно, что выбор структур данных для представления разреженных матриц существенно влияет на производительность алгоритма SpMV, ограниченную множеством факторов, таких как нерегулярный доступ к памяти, дисбаланс нагрузки, косвенная адресация, малая арифметическая интенсивность. К числу основных, наиболее распространенных форматов хранения разреженных матриц следует отнести CRS, COO, Ellpack, Cell-C-Sigma [1]. Существует множество реализаций алгоритма SpMV для архитектур x86 и ARM, однако все они не решают указанные выше проблемы в общем случае.

В работе изучается вопрос о возможности более рационального представления разреженных матриц, учитывающего особенности алгоритма SpMV и ориентированного на архитектуру и систему команд RISC-V. Работа является частью коллективного проекта в институте ИТММ ННГУ, в рамках которого был выполнен детальный анализ производительности алгоритма SpMV для восьми форматов представления разреженных матриц. Эксперименты показали, что традиционный и поддерживаемый в программных библиотеках формат CRS наряду с другими форматами может приводить к низкой эффективности по следующим причинам. Во-первых, обход вектора x в пределах одной строки выполняется по произвольным индексам, что снижает эффективность кэширования данных вектора. Во-вторых, в данный момент возможности автоматической векторизации кода для компиляторов языка C++ под архитектуру RISC-V ограничены, поэтому ручная векторизация может быть эффективнее.

В докладе предлагается новый формат хранения разреженных матриц HCSR, направленный на частичное решение указанных проблем эффективности алгоритма SpMV для матриц в формате CRS. В предложенном формате матрица разбивается на блоки фиксированного размера $m_{block} \times n_{block}$, количество ненулевых элементов в блоках различно и равно nz_{block}. Каждый ненулевой блок хранится как матрица CRS или COO, в зависимости от заполненности ненулевыми элементами. В сравнении с хранением всех блоков в формате CRS, такой подход позволяет более эффективно хранить сильно разреженные блоки, а именно при $nz_{block} < m_{block}$. При этом, параметром переключения p между форматами будет являться среднее количество ненулевых элементов на строку $p_{block} = nz_{block}/m_{block}$. Для заданного порогового значения p_{fix} при $p_{block} > p_{fix}$ используется формат CRS, иначе используется COO. Блоки матрицы представляют собой разреженную матрицу, поэтому ненулевые блоки хранятся как элементы формата CRS, представляющего исходную матрицу по таким блокам. Если размерности матрицы не кратны заданным m_{block}, n_{block}, то последний столбец матрицы блоков будет состоять из блоков размерности $m_{block} \times (n_{block}\ mod\ n)$, а последняя

*Работа выполнена при поддержке программы академического лидерства ННГУ «Приоритет-2030».

строка блоков – из блоков размерности $(m_{block}\ mod\ m) \times n_{block}$. Похожие блочные форматы представления разреженных матриц описаны в работах [2], [3], однако в формате, описанном в работе [2], блоки хранятся как матрицы в координатном формате, что позволяет выполнять операции Ax и $A^T x$ за одинаковое время, а в формате, описанном в работе [3], блоки хранятся как плотные матрицы, что позволяет упростить векторизацию алгоритма SpMV.

Реализация алгоритма SpMV в данном формате отличается от SpMV в формате CRS только тем, что в качестве произведения элемента матрицы на элемент вектора применяется произведение блока на некоторый сегмент вектора. При этом умножение различных строк блоков на сегменты вектора можно выполнять параллельно. Для небольших размеров блоков при умножении его на сегмент вектора x он может оказаться достаточно малого размера, чтобы полностью сохраниться в кэш-памяти, за счёт чего эффективность кэширования вектора x будет повышена. Также в алгоритме умножения блока в формате CRS на вектор была использована ручная векторизация, где элементы строки блока и элементы вектора x загружались в векторные регистры, после чего вычислялось скалярное произведение данных, сохраненных в этих регистрах, результат которого сохранялся в соответствующую компоненту итогового вектора y. Тестирование показало, что векторная реализация произведения матрицы в формате COO на вектор менее эффективна, чем скалярная реализация. В реализациях алгоритмов для параллелизма на общей памяти использовалась библиотека OpenMP.

Вычислительные эксперименты производились на плате Banana Pi BPI-F3, процессор SpacemiT Keystone K1 с поддержкой 256-bit RVV 1.0, 16 GB, ОС Bianbu 1.0.15, кросс-компилятор GCC RISC-V 14.2.0. Для тестов было использовано 90 матриц из коллекции Suite Sparse [4], время работы SpMV в формате CRS на этих матрицах находится в диапазоне от 0,002 до 2 секунд. В качестве меры производительности предложенного алгоритма по сравнению с алгоритмом в формате CRS использовано число, названное относительным ускорением, которое равно отношению времени работы SpMV в формате CRS к времени работы SpMV в HCSR. Наилучшие значения параметров m_{block}, n_{block}, p_{fix} были подобраны экспериментально и использовались *одни и те же* для всех матриц в эксперименте. Все вычислительные эксперименты производились в числах двойной точности.

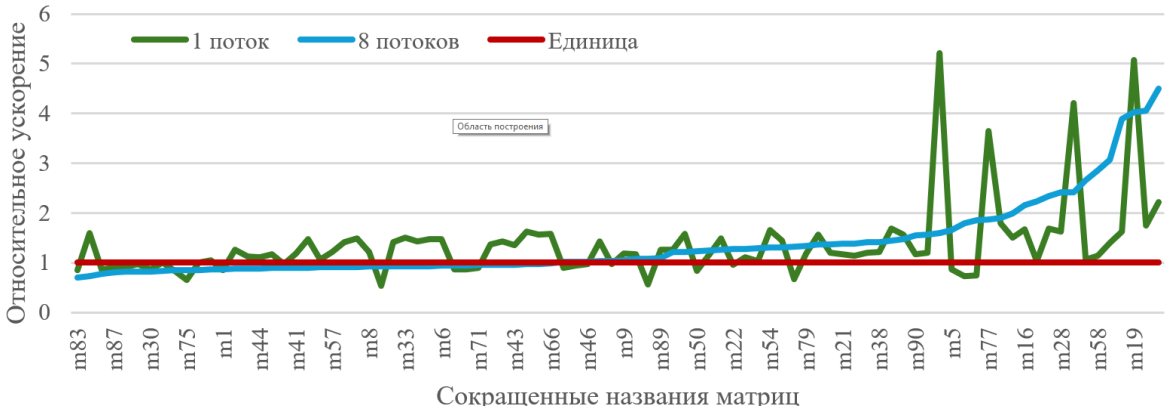


Рис. 1. Относительная производительность алгоритма SpMV в HCSR в сравнении с CRS для 1 и 8 потоков. Данные отсортированы по возрастанию относительного ускорения для 8 потоков. Значения больше 1 означают ускорение относительно CRS, значения меньше 1 соответствуют замедлению.

Как показали эксперименты (рис. 1), и CRS, и предложенный формат HCSR показывают наилучшее время на наибольшем количестве потоков (равном 8), поэтому основным показателем эффективности алгоритма будет именно прирост на 8 потоках. Максимальное ускорение на 8 потоках достигает 4,5, при этом на 50 матрицах алгоритм сработал быстрее, чем CRS, а на 40 – медленнее. Минимальное ускорение составило 0,7 (то есть, наблюдалось замедление на 30% относительно CRS). В среднем, предложенный формат опережает CRS в 1,39 раз.

Результаты для количества потоков, равного 1 (последовательная реализация) и равного 8 существенно отличаются. Максимальное ускорение при последовательной реализации алгоритмов SpMV равно 5,22 и достигается не на матрице, на которой достигалось максимальное ускорение для 8 потоков. Максимальное замедление составило 0,53 (т.е. замедление на 47%). В среднем, предложенный формат опережает CRS в 1,36 раз при последовательной реализации обоих алгоритмов. Результаты будут интегрированы в библиотеку разреженных алгоритмов линейной алгебры, разрабатываемую в ННГУ.

Список литературы

[1] Gao J. et al. A systematic literature survey of sparse matrix-vector multiplication // arXiv preprint arXiv:2404.06047. – 2024.

[2] Yang, B., Gu, S., Gu, T., Zheng, C. and Liu, X. (2014) Parallel Multicore CSB Format and Its Sparse Matrix Vector Multiplication. *Advances in Linear Algebra & Matrix Theory*, **4**, 1-8. doi.

[3] Ma, M., Huang, X., Xu, J. *et al.* Implementation and optimization of SpMV algorithm based on SW26010P many-core processor and stored in BCSR format. *Sci Rep* **14**, 16574 (2024).

[4] SuiteSparse Matrix Collection, `https://suitesparse-collection-website.herokuapp.com/`, дата последнего доступа 01.11.2025.

Оценка точности симулятора системы управления заданиями*

А.В. Баранов[1], Д.С. Ляховец[1], А.А. Тулинов[2]
НИЦ «Курчатовский институт»[1], РТУ МИРЭА[2]

Системы управления заданиями (СУЗ) являются одним из ключевых компонентов су-перкомпьютерных комплексов. СУЗ отвечает за планирование заданий, то есть определе-ние порядка их запуска на вычислительных ресурсах. При исследовании алгоритмов и настроек планирования широко применяются имитационные модели и специализирован-ные симуляторы [1]. Практическая ценность моделирования определяется тем, насколько точно симулятор воспроизводит поведение реальной системы. Даже при использовании симулятора, повторяющего логику СУЗ, результат моделирования может заметно отли-чаться от реального плана запуска заданий [2].

В исследовании рассматривается задача оценки точности симулятора системы управ-ления заданиями. Точность моделирования будем оценивать путём сравнения плана за-пуска симулятора с планом запуска реальной суперкомпьютерной системы. Это позволяет анализировать не только усреднённые интервальные показатели качества, такие как за-грузка ресурсов, среднее время ожидания и среднее время замедления, но и порядок за-пуска заданий. Исследование ориентировано на практический вопрос: какие метрики мо-гут быть использованы для оценки точности симулятора. В исследовании рассматривают-ся следующие метрики: средняя загрузка ресурсов, среднее время ожидания задания в очереди, среднее ограниченное замедление, показатель близости [2], метрика Вассер-штейна, мера Жаккара (или Intersection over Union, IoU), оценка по алгоритму динамиче-ской трансформации временной шкалы (Dynamic Time Warping, DTW).

На постере авторами будут представлены результаты двух серий экспериментов. Для первой серии экспериментов использовался журнал работы раздела Broadwell суперком-пьютера МВС-10П ОП в Отделении суперкомпьютерных систем и параллельных вычисле-ний НИЦ «Курчатовский институт» за недельный период. Раздел находится под управле-нием СУППЗ. Сравнение производилось для результатов работы реальной системы, мо-делировании в симуляторе СУППЗ с приоритетами пользователей, в симуляторе СУППЗ без приоритетов пользователей, и в симуляторе Slurm. Для второй серии экспериментов использовался входной поток, созданный с помощью генератора [3] и содержит 5000 за-даний, рассчитанных на выполнение в течение 5 суток на 128 процессорных ядрах. Вход-ной поток обрабатывался в симуляторе Slurm с различным коэффициентом ускорения – параметром, отвечающем за скорость моделирования.

Полученные результаты позволяют сделать вывод, что для оценки точности симулятора недостаточно одной усреднённой метрики, и даны практические рекомендации по использованию набора метрик для оценки точности.

Список литературы

[1] Simakov N. A., Deleon R. L., Lin Y., Hoffmann Ph. S., Mathias W. R.. Developing accurate Slurm simulator // PEARC '22: Practice and Experience in Advanced Research

*В рамках государственного задания НИЦ «Курчатовский институт».

Computing, 2022, article 59, P. 1–4. DOI: 10.1145/3491418.3535178.

[2] Baranov A., Telegin P., Shabanov B., Lyakhovets D. Measure of adequacy for the super-computer job management system model // Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019, ACSIS, Vol. 18, P. 423-426. DOI: 10.15439/2019F186.

[3] Lublin U., Feitelson G. The Workload on Parallel Supercomputers: Modeling the Characteris-tics of Rigid Job // Journal of Parallel and Distributed Computing, 2003, v. 63, № 11, P. 1105–1122. DOI: 10.1016/S0743-7315(03)00108-4.

Параллельный алгоритм генерации оптимальных циркулянтных графов

П.В. Бусько[1], Е.А. Козинов[1], В.Д. Волокитин [1], Э.Р. Рзаев[2], И.Б. Мееров[1]

[1]Нижегородский государственный университет им. Н. И. Лобачевского, Нижний Новгород

[2]Национальный исследовательский университет «Высшая школа экономики», Москва

Увеличение количества ядер в многопроцессорных системах на кристалле является одним из ключевых способов повышения вычислительной мощности современных систем. Ограничением таких систем является шинная организация обмена данными между ядрами. При увеличении их количества нагрузка на шину возрастает, и ее пропускная способность становится недостаточной, что снижает эффективность параллельных вычислений. Возможным подходом к решению данной проблемы является использование сетей на кристалле (СтнК) – коммуникационных подсистем, в которых обмен данными осуществляется через маршрутизаторы по нескольким соединениям одновременно. Ключевой задачей при проектировании СтнК является выбор топологии соединения вычислительных ядер. Циркулянтные топологии [1] представляют собой эффективный подход к организации межсоединений, обладающий рядом преимуществ с точки зрения масштабируемости и коммуникационной эффективности. В работе [2] подтверждена актуальность их применения в проектировании СтнК, а также представлен набор данных оптимальных циркулянтных топологий по критерию минимизации диаметра и средней длины кратчайших путей. Целью настоящего исследования является оптимизация работы ПО для автоматизированного синтеза циркулянтных топологий [3], что позволит повысить эффективность проектирования СтнК.

Циркулянтным графом [1, 4] называют неориентированный связный граф с множеством вершин $V = \{0, 1, \ldots, N-1\}$ и ребер $E = \{(v_i, v_j) : |i-j| \equiv s_m (\mathrm{mod}\ N), m = 1, \ldots, k\}$, где $s_1, s_2, \ldots, s_k, N$ – целые числа, такие, что $1 \leq s_1 < s_2 < \ldots < s_k < N/2$. Сигнатура такого графа (его параметрическое описание) имеет следующий вид: $C(N; s_1, s_2, \ldots, s_k)$, где N – количество вершин, k – размерность графа, а множество $S = \{s_1, s_2, \ldots, s_k\}$ – множество образующих. Образующие циркулянтного графа определяют переходы между соседними вершинами. Подробное описание математической модели циркулянта представлено в работе [4].

Основными характеристиками циркулянтных графов являются диаметр и средняя длина кратчайших путей (average shortest path length, ASPL). Задача поиска оптимальных циркулянтных графов заключается в нахождении одного или нескольких графов с фиксированными параметрами N и k, имеющих минимальные показатели диаметра и ASPL.

В данной работе представлен гибридный параллельный алгоритм, объединяющий технологии MPI и OpenMP. Ниже приведены основные этапы предлагаемого алгоритма:

1. Инициализация MPI.
2. Распределение комбинаций диапазонов параметров графов:

- Вычисление общего количества комбинаций.
- Равномерное распределение задач между MPI-процессами.

3. Поиск глобальных оптимальных характеристик графов:
   - Распределение комбинаций между OpenMP-потоками (внутри каждого MPI-процесса).
   - Генерация набора генераторов по заданному индексу для каждой комбинации.
   - Вычисление диаметра и средней длины кратчайшего пути (стандартный BFS).
   - Определение локально-оптимальных характеристик внутри потока.
   - Определение глобально-оптимальных характеристик по всем процессам.
4. Определение всех оптимальных комбинаций.
5. Завершение MPI-процессов.

Для сокращения перебора возможных комбинаций учитывается свойство симметрии циркулянтов: генераторы s и $N - s$ дают один и тот же граф. Общее количество комбинаций генераторов определяется с помощью биномиального коэффициента:

$$C^{k-s_0}_{\left\lfloor \frac{N}{2} \right\rfloor - s_0}.$$

Комбинации равномерно распределяются между MPI-процессами, после чего внутри каждого процесса обрабатываются параллельно OpenMP-потоками. Для каждой комбинации генераторов вычисляются характеристики графа с использованием алгоритма поиска в ширину. Благодаря регулярной структуре и симметрии циркулянтных графов можно не хранить топологии графа в памяти в явном виде. Соседи вершины вычисляются динамически по модульной арифметике с использованием образующих, что снижает требования к памяти с $O(N^2)$ (для матрицы смежности) до $O(k)$. Данный подход гарантирует сохранение эффективного использования памяти при масштабировании до больших размеров графа. После обработки всех комбинаций определяются глобально-оптимальные характеристики, по которым далее находятся оптимальные сигнатуры графов.

Вычислительные эксперименты проводились на суперкомпьютере ННГУ «Лобачевский» с использованием процессоров Intel Sandy Bridge E5-2660 (8 ядер, поддержка HT, 64 ГБ ОЗУ). Были рассмотрены различные конфигурации количества процессов и потоков. Установлено, что на одном узле наилучшие результаты достигаются при использовании одного MPI-процесса и максимального количества OpenMP-потоков.

Полученные результаты показали, что при запуске на одном узле в 32 потока (с HT) достигается ускорение в 17–19 раз. Исследование масштабируемости проводилось при определенной ранее оптимальной конфигурации запуска на одном узле (1 MPI-процесс, 32 OpenMP-потока) при увеличении количества узлов с 1 до 32. На рисунке 1 (а) приведен график зависимости времени работы программы от количества узлов для графа $N = 1000,\ k = 5$. Из рисунка следует, что при увеличении количества вычислительных узлов с 1 до 32 время поиска сигнатур оптимальных циркулянтов сократилось в 25 раз.

На рисунке 1 (б) приведен график зависимости эффективности алгоритма поиска при увеличении количества узлов. При использовании 32 узлов эффективность сильной масштабируемости составляет 79,75%, что приемлемо для практического использования.

Список литературы

[1] Boesch F., Tindell R. Circulants and their connectivities // Journal of Graph Theory. 1984. Vol. 8, № 4. P. 487–499.

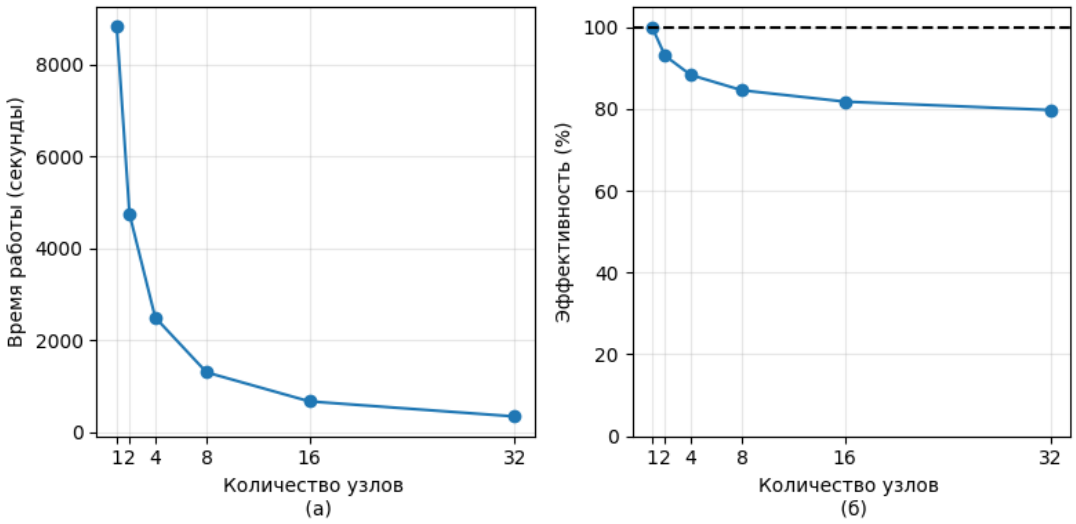


Рис. 1. (а) – график зависимости времени работы программы от количества узлов для параметров $N = 1000,\ k = 5$; (б) – график эффективности параллельных вычислений при увеличении количества узлов для тех же параметров.

[2] Romanov A. The dataset for optimal circulant topologies // Big Data and Cognitive Computing. 2023. Vol. 7, № 2. P. 80.
[3] nyoxue. pcgp: ПО для синтеза циркулянтных топологий [Электронный ресурс] / nyoxue. – Режим доступа: git.hab.com/nyoxue/pcgp (дата обращения: 09.04.2026).
[4] Монахова Э.А. Структурные и коммуникативные свойства циркулянтных сетей // Прикладная дискретная математика. 2011. № 3 (13). С. 92–115.

Сравнение производительности операции SpMV для различных форматов хранения разреженной матрицы на процессорах архитектуры RISC-V*

А.А. Воденеева, А.Ю. Пирова, И.Б. Мееров, Е.А. Козинов, В.Д. Волокитин, А.В. Устинов, К.И. Ковалев, М.А. Загрядсков, А.И. Кулик, Д.Д. Литвяков

Нижегородский государственный университет им. Н. И. Лобачевского

В работе исследована зависимость производительности умножения разреженной мат-рицы на вектор (SpMV) от формата хранения разреженной матрицы на процессорах архи-тектуры RISC-V. Алгоритм SpMV является одним из ключевых алгоритмов линейной ал-гебры. В частности, он является основным вычислительно-трудоемким ядром итерацион-ных решателей СЛАУ с разреженными матрицами, которые применяются для решения широкого спектра задач математической физики с использованием суперкомпьютеров. На сегодняшний день при кажущейся простоте данного алгоритма до сих пор не решены проблемы с эффективным представлением разреженных матриц, позволяющим раскрыть потенциал современных вычислитель-ных систем. Наиболее распространенные программ-ные библиотеки разреженной линейной алгебры (Intel oneAPI MKL, AOCL-Sparse, SuiteSparse и др.) используют классические форматы хранения матриц, которые не позво-ляют в полной мере задей-ствовать механизм SIMD инструкций, поддерживаемых совре-менными процессорами, и не адаптированы для процессоров RISC-V. За последние годы было предложено немало оригинальных форматов хранения разреженных матриц для процессоров x86 и GPGPU, однако большинство из них распространяется в виде отдель-ных библиотек и не интегрировано с решателями СЛАУ. Более того, эффективность пред-ложенных решений для процессоров архитектуры RISC-V не изучена.

В работе была выполнена программная реализация операции SpMV для чисел с оди-нарной и двойной точностью для форматов хранения CRS, Sell-C-Sigma [1], CSR2 [2], CSR5 [3], CVR [4], VHCC [5], VNEC [6] и LAV [7], а также для предложенного блочного формата HCSR. Программная реализиция была выполнена с использованием интринсиков RVV 1.0 и технологии OpenMP.

Эксперименты проводились на плате Banana Pi BPI-F3 с процессором SpacemiT Keystone K1. Для экспериментов было выбрано 89 матриц из коллекции SuiteSparse [5] размером от 200 тысяч до 5 млн. строк. Запуски проводились в 1 и 8 потоков, для чисел одинарной и двойной точности. При проведении экспериментов сравнивалось время выполнения операции SpMV для всех реализованных форматов относительно скалярной ре-ализации для формата CRS. Рисунки 1 и 2 показывают относительное ускорение операции SpMV для двойной точности при работе в 1 и 8 потоков соответственно. На графиках по горизонтали отмечены названия тестовых матриц, по вертикали – достигнутое ускорение относительно формата CSR на конкретной матрице. Матрицы упорядочены в порядке воз-растания количества ненулевых элементов.

*Работа выполнена при поддержке программы академического лидерства ННГУ «Приоритет-2030».

Как видно из графиков, для двойной точности при работе в 1 поток скалярный CRS проигрывает в большинстве тестовых случаев. При этом лидером по производительности является формат SELL-C-Sigma, который показал лучшее среди всех форматов время ра-боты на 65% тестовых матриц, в среднем опережая CRS в 1.5 раза. Также на половине те-стового набора прирост производительности демонстрируют форматы CVR и VHCC, опе-режая CRS на 24% и 41% соответственно. Векторная версия CRS опережает скалярную чуть более, чем на половине тестовых матриц, в среднем на 27%. Предложенный формат HCSR опережает CRS в 1.4 раза в среднем и в 5.2 раза в лучшем случае. При работе в 8 потоков векторный или скалярный CRS показал лучшее время работы на трети тестовых матриц. Среди других форматов выделяются Sell-C-Sigma и VHCC, показавшие лучшую производительность среди всех форматов на 20% тестовых матриц каждый, опережая CRS в среднем на 12% и 79% соответственно. Умножение в формате CSR5, хотя и не было са-мым быстрым, в среднем работает на 77% быстрее скалярного CRS. Предложенный фор-мат HCSR опережает CRS на 39% в среднем.

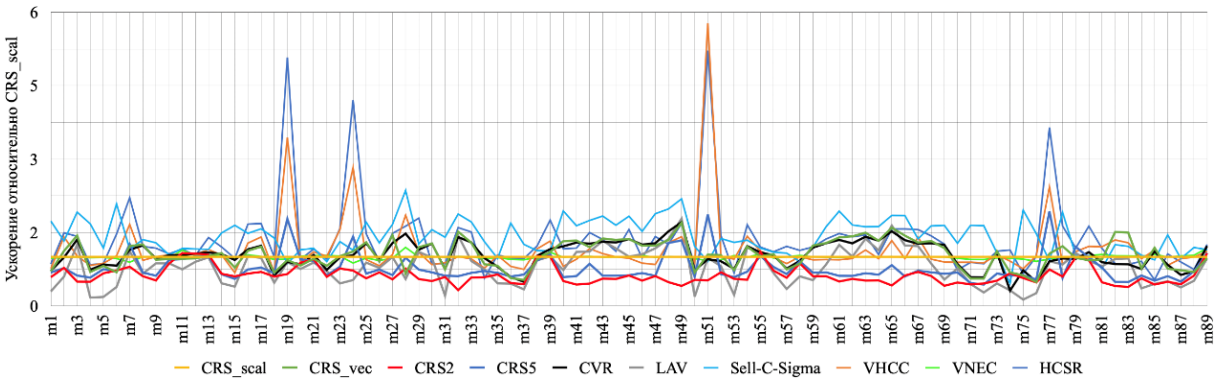


Рис. 1. Результаты работы библиотеки на тестовом наборе матриц для чисел с плавающей запятой двойной точности (double) при работе в 1 поток

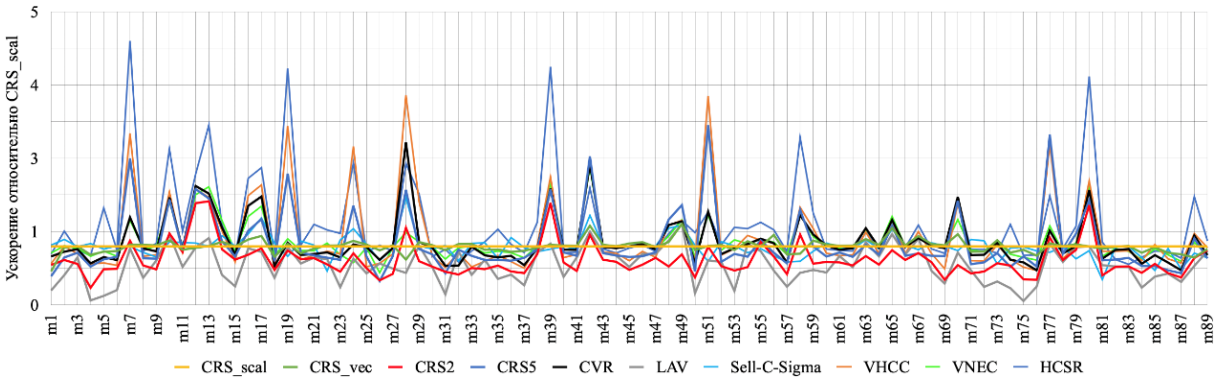


Рис. 2. Результаты работы библиотеки на тестовом наборе матриц для чисел с плавающей запятой двойной точности (double) при работе в 8 потоков

Результаты экспериментов показывают, что использование нестандартных форматов хранения разреженных матриц позволяет ускорить вычисления до 5.8 раз, в среднем – на 34%. В дальнейшем планируется интеграция с итерационными решателями разреженных СЛАУ.

Список литературы

[1] Kreutzer M. et al. A unified sparse matrix data format for efficient general sparse matrix-vector multiplication on modern processors with wide SIMD units //SIAM Journal on Scien-tific Computing. – 2014. – Т. 36. – №. 5. – С. C401-C423.

[2] Bian H. et al. A simple and efficient storage format for SIMD-accelerated SpMV //Cluster Computing. – 2021. – Т. 24. – №. 4. – С. 3431-3448.

[3] Liu W., Vinter B. CSR5: An efficient storage format for cross-platform sparse matrix-vector multiplication //Proceedings of the 29th ACM on International Conference on Supercompu-ting. – 2015. – С. 339-350.

[4] Xie B. et al. Cvr: Efficient vectorization of spmv on x86 processors //Proceedings of the 2018 International Symposium on Code Generation and Optimization. – 2018. – С. 149-162.

[5] Tang W. T. et al. Optimizing and auto-tuning scale-free sparse matrix-vector multiplication on Intel Xeon Phi //2015 IEEE/ACM International Symposium on Code Generation and Op-timization (CGO). – IEEE, 2015 – С. 136-145.

[6] Wang L. et al. VNEC: A Vectorized Non-Empty Column Format for SpMV on CPUs //2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS). – IEEE, 2024 – С. 14-25.

[7] Yesil S. et al. Speeding up SpMV for power-law graph analytics by enhancing locality & vectorization //SC20: International Conference for High Performance Computing, Network-ing, Storage and Analysis. – IEEE, 2020 – С. 1-15.

[8] SuiteSparse Matrix Collection, https://suitesparse-collection-website.herokuapp.com/, дата последнего доступа 01.11.2025.

Численное моделирование распространения фронта пожара на динамически-адаптивных сетках

Е.М. Гащук[1,2], В.И. Суязова[2,3], Е.В. Мортиков[3,4]

Московский государственный университет им. М.В.Ломоносова, Механико-математический факультет, Москва, 119234, Российская Федерация[1]
Московский государственный университет им. М.В. Ломоносова, Научно-исследовательский вычислительный центр, Москва, 119234, Российская Федерация[2]
Институт физики атмосферы им. А.М. Обухова РАН, Москва, 119017, Российская Федерация[3]
Институт вычислительной математики им. Г.И. Марчука РАН, Москва, 119333, Российская Федерация[4]

В работе рассматривается численная модель распространения фронта природного пожара на основе метода функции уровня. Скорость фронта задается полуэмпирической моделью Ротермела, а для уточнения разрешения вблизи границы фронта используется динамически-адаптивная сетка типа восьмидерево. Программная реализация поддерживает выполнение расчетов как на центральных процессорах, так и на графических ускорителях. Для валидации алгоритма функции уровня проведены идеализированные тесты. В дальнейшем планируется сравнение численных результатов с применением и без применения AMR (Adaptive Mesh Refinement) для идеализированных тестов и моделирования реальных пожаров.

Ключевые слова: природные пожары, функция уровня, модель Ротермела, динамически-адаптивные сетки, графические ускорители

Природные пожары представляют серьезную угрозу для экосистем и населенных пунктов. Они приводят к деградации растительного покрова, повреждению объектов транспортной и энергетической инфраструктуры. Задача моделирования распространения фронта пожара играет важную роль при планировании тушения, а также при поддержке эвакуационных и иных спасательных операций. Вместе с тем моделирование пожаров важно и с точки зрения моделирования состояния атмосферы, так как пожары приводят к значительному изменению структуры приземного слоя.

Численные модели пожаров различаются по уровню физической детализации. Полностью физические модели основаны на фундаментальных законах термодинамики, механики жидкости и химии горения (например, WFDS [1]). Полуфизические модели основаны на эмпирических зависимостях. Например, модель Ротермела [4], описывает скорость фронта природного пожара. Данный класс моделей получил более широкое распространение в задачах прогноза природных пожаров, поскольку сочетает вычислительную эффективность с возможностью описания основных механизмов распространения огня. Среди полуэмпирических моделей наиболее распространен подход, в котором фронт пожара описывается методом функции уровня. Такой подход

позволяет учитывать сложную форму фронта, его слияние, расщепление и образование несгоревших участков без явного отслеживания отдельных точек границы [3]. Точное воспроизведение геометрии фронта требует достаточно мелкого пространственного разрешения, особенно в областях с сильной кривизной границы и резким градиентом смещения фронта. С другой стороны, применение такого же высокого разрешения во всей расчетной области избыточно. Это делает естественной идею локального сгущения сетки в окрестности фронта пожара с сохранением более грубого разрешения вдали от него. Данный подход может быть реализован с использованием динамически-адаптивных сеток [2].

В работе рассматривается модель распространения фронта пожара на основе метода функции уровня. Скорость распространения фронта пожара вычисляется с использованием полуэмпирической модели Ротермела [4], учитывающей влияние характеристик топлива, поверхности и скорости ветра на интенсивность горения. В процессе численного интегрирования накопленная погрешность может сильно сказаться на полученных результатах. По этой причине после каждого шага интегрирования решается уравнение реинициализации [6] функции уровня. Численная модель поддерживает проведение расчетов с использованием динамически-адаптивных сеток типа восьмидерево с коэффициентом сгущения 2. Сгущение сетки выполняется локально вблизи фронта пожара, а на границах ячеек различных уровней проводится корректировка потоков через границу грубой ячейки. Программная реализация написана на C++/CUDA и поддерживает проведение расчетов на NVIDIA GPU (Graphics Processing Unit).

Для валидации алгоритма функции уровня без использования AMR сеток проведены тесты на идеализированных постановках распространения фронта пожара с ненулевой поперечной компонентной. Оценки выгоревшей площади в зависимости от времени для шагов 10 и 20 м оказались близкими, тогда как при шаге 50 м отличие стало существенным. На следующем этапе будет проведено сравнение численных результатов на идеализированных тестах при использовании AMR сеток и без. Будут сопоставляться форма фронта, выгоревшая площадь и чувствительность результата к разрешению. Отдельно будет выполнено сравнение времени расчета для фиксированных и адаптивных сеток, включая расчеты на CPU (Central Processing Unit) и GPU. Далее планируется перейти к расчетам реальных пожаров. Для таких тестов планируется сравнение численных результатов со спутниковыми данными о выгоревшей области. Аналогичный подход к анализу реального события использован при моделировании пожара Кемп в Калифорнии 2018 года при использовании модели WRF-Fire [5]. Будет проведено сравнение результатов с AMR сеткой и без, а также времени выполнения расчетов.

Список литературы

[1] Mell William, Charney Joseph, Jenkins Mary Ann, Cheney Phil, Gould Jim. Numerical Simulations of Grassland Fire Behavior from the LANL-FIRETEC and NIST-WFDS Models // Remote Sensing and Modeling Applications to Wildland Fires. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. 209–225.

[2] Min Chohong, Gibou Fr´ed´eric. A second order accurate level set method on non-graded adaptive cartesian grids // Journal of Computational Physics. Vol. 225, No. 1. P. 300–321. 2007. DOI: https://doi.org/10.1016/j.jcp.2006.11.034.

[3] Rehm Ronald G., McDermott Randall J. Fire-Front Propagation Using the Level Set Method. Gaithersburg, MD, 2009.

[4] Rothermel Richard C. A mathematical model for predicting fire spread in wildland fuels. 115. 1972.

[5] Shamsaei Kasra, Juliano Timothy W., Roberts Matthew, Ebrahimian Hamed, Kosovic Branko, Lareau Neil P., Taciroglu Ertugrul. Coupled fire-atmosphere simulation of the 2018 Camp Fire using WRF-Fire // International Journal of Wildland Fire. Vol. 32, No. 2. P. 195–221. 2023. DOI: 10.1071/WF22013.

[6] Sussman Mark, Smereka Peter, Osher Stanley. A level set approach for computing solutions to incompressible two-phase flow // Journal of Computational physics. Vol. 1994. 114, No. 1. P. 146–159.

Содержание

Полные и короткие статьи

Аннотации

Russian Supercomputing Days : Proceedings of the International Conference. September 28–29, 2026, Moscow, Russia / Ed. by A.S. Antonov, Vad. V. Voevodin, D.A. Nikitenko. – Moscow : MAKS Press, 2026. – 282 p.

ISBN: 978-5-317-07665-8, e-ISBN: 978-5-317-07672-6
https://doi.org/10.29003/m5440.978-5-317-07665-8

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

Научное издание

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ-2026
Труды международной конференции
28–29 сентября 2026 г. Москва

Издательство «МАКС Пресс»
Главный редактор: *Е. М. Бугачева*
Компьютерная верстка: *С. И. Кауль*
Обложка: *А. В. Кононова*

Подписано в печать 15.09.2026 г. Формат 60x90 1/8.
Усл.печ.л. 35,25. Тираж 24 экз. Изд. № 153.

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»
109316, г. Москва, Волгоградский проспект, д. 42,
корп. 5, эт. 1, пом. I, ком. 6.3-23Н

9 785317 076658

СУПЕРКОМПЬЮТЕРНЫЕ ДНИ В РОССИИ

Труды международной конференции

28–29 сентября 2026 г., Москва

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

Proceedings of the International Conference

September 28–29, 2026, Moscow

RUSSIAN SUPERCOMPUTING DAYS

Proceedings of the International Conference

September 28–29, 2026, Moscow, Russia

МАКС ПРЕСС