Supercomputing Frontiers and Innovations

2025, Vol. 12, No. 4

Scope

- Future generation supercomputer architectures
- Exascale computing
- Parallel programming models, interfaces, languages, libraries, and tools
- Supercomputer applications and algorithms
- Novel approaches to computing targeted to solve intractable problems
- Convergence of high performance computing, machine learning and big data technologies
- Distributed operating systems and virtualization for highly scalable computing
- Management, administration, and monitoring of supercomputer systems
- Mass storage systems, protocols, and allocation
- Power consumption minimization for supercomputing systems
- Resilience, reliability, and fault tolerance for future generation highly parallel computing systems
- Scientific visualization in supercomputing environments
- Education in high performance computing and computational science

Editor-in-Chief's Introduction for Special Issue

dedicated to the 70th anniversary of the Moscow State University Research Computing Center, reflecting the center's current research in the field of high-performance computing

The Moscow State University Computing Center was established in 1955. It was the first computing center in the higher education system and one of the first in the country. The creation of the computing center at Moscow State University was driven by the need to train a large number of highly qualified specialists in the field of computer science, as well as specialists capable of solving complex scientific and economic problems using state-of-the-art computing technology.

The computing center has always played a significant role in the dissemination of advanced computer technologies. This dissemination took a variety of forms, including providing scientific and technical consultations, allocating computer time, sharing expertise, and assisting in solving specific problems. This last type of activity led to the creation of the largest library of numerical analysis programs in the country at the computing center.

Ensuring the effective use of computing technology requires highly qualified specialists, not only in engineering, but also in programming, numerical methods, mathematical modeling, and other fields. That is why the primary computing equipment was concentrated in the Computing Center, where the essential personnel with the required qualifications were available.

The MSU Supercomputing Center is one of the leading supercomputing centers in Russia, and one of the most important areas of the Research Computing Center's work is supporting this HPC facility, ensuring the effective use of high-performance computing systems. At present, the core of the Center is the Lomonosov-2 supercomputer. This issue contains selected papers reflecting the current research of Research Computing Center scientists in the field of computing technology.

Editor-in-Chief
Vladimir Voevodin, Corr. RAS,
Director of Research Computing Center,
Lomonosov Moscow State University,
Moscow, Russia

Contents

DOI: 10.14529/jsfi250401

Distillation for Adaptation Language Models to Russian Language

***Grigory P. Kovalev*[1], *Mikhail M. Tikhomirov*[1]**

Adapting large language models (LLMs) to morphologically rich languages like Russian presents a major challenge, as multilingual models often exhibit limited transfer due to predominantly English-centric pre-training. This study investigates knowledge distillation (KD) as a more effective alternative to supervised fine-tuning (SFT) for the final calibration stage of language adaptation. We introduce an efficient offline top-K distillation approach that transfers knowledge from a 32B Russian-adapted teacher model to a 4B student model through tokenizer alignment and direct logit transfer. Experimental results demonstrate that KD consistently surpasses SFT, achieving up to a 4.22% performance improvement, with top-100 distillation yielding the highest gains (3.27% on average) albeit with increased memory consumption (62 GB vs. 7 GB for top-10). Moreover, the advantages of KD are most pronounced for student models with lower adaptive capacity (i.e., smaller LoRA α values). These findings underscore the efficacy of KD as a practical and scalable approach for language adaptation, while emphasizing the necessity of balancing performance improvements against computational efficiency.

Keywords: large language model, distillation, fine-tuning, model adaptation, Russian language.

Introduction

Large Language Models (LLMs) such as Qwen [20], DeepSeek [3], Llama [4], and GPT [1] have rapidly advanced the state of the art in NLP. Despite nominal multilinguality, their pre-training data is heavily dominated by English, which limits performance in underrepresented languages. The gap is particularly pronounced for morphologically rich languages like Russian, where inflectional complexity and orthographic variation amplify subword fragmentation under default tokenizers. As a result, language adaptation – porting an existing LLM to a target language and tokenizer while preserving its instruction alignment and skills – remains a practical necessity.

A common adaptation pipeline involves extending the model's tokenizer, performing continued pre-training, and then conducting supervised fine-tuning (SFT) to align the model with language-specific instructions. While this approach improves fluency, SFT relies on hard targets and may not be the most effective way to transfer knowledge, especially when adapting a smaller model under the guidance of a much larger, more capable one. A more potent alternative is knowledge distillation, where a "student" model learns from the full output distribution of a "teacher" model. However, distillation is often complicated by tokenizer mismatches, making direct logit transfer between different models problematic.

This paper investigates logit distillation as a superior alternative to SFT within a language adaptation pipeline. Our core methodology resolves the tokenizer mismatch problem by first aligning the vocabularies of the teacher and student models through a shared extension of Russian-specific tokens. This enables a direct and clean transfer of knowledge. Concretely, we leverage a pre-existing 32B Qwen3[2] model, which was fully adapted for Russian. We then use it as a teacher to guide the adaptation of a 4B Qwen3[3] student model. Our primary contribution

[1]Lomonosov Moscow State University, Moscow, Russian Federation

[2]`https://huggingface.co/RefalMachine/RuadaptQwen3-32B-Instruct`

[3]`https://huggingface.co/Qwen/Qwen3-4B`

is a comprehensive comparison showing that replacing the final SFT stage with tokenization-aligned logit distillation results in a more powerful and efficient Russian language model.

To ensure reproducibility and to facilitate the development of language-specific models, we publicly release our code on GitHub[4].

The remainder of this paper is organized as follows. Section 1 reviews prior work on multilingual large language model adaptation and knowledge distillation. Section 2 describes the model adaptation pipeline based on tokenization alignment and the Learned Embedding Propagation (LEP) technique. Section 3 presents the proposed logit distillation approach designed to improve training efficiency and stability. Section 4 details the datasets used for adaptation and fine-tuning, while Section 5 outlines the evaluation framework and benchmarks employed to assess model performance across diverse linguistic and reasoning tasks. Section 6 reports experimental results and analysis. Section 7 discusses the main limitations of our study, and the paper concludes with a summary of key findings and outlines directions for future research.

1. Background

Several adaptation methods have been proposed to overcome the limitations of multilingual LLMs for languages such as Russian. The most straightforward method for such adaptation is supervised fine-tuning (SFT) on target-language instructions [16]. A more complex but efficient variant is to perform tokenization vocabulary adaptation [21] before the SFT step, which can better align the model's internal representations with the target language's morphology. The fine-tuning process itself is an active area of research, with methods including classical SFT, reinforcement learning from human feedback (RLHF) to align outputs with human preferences [17], and knowledge distillation, where a smaller "student" model learns from a larger "teacher" models outputs to transfer knowledge efficiently [10]. Among these, knowledge distillation is particularly promising for language adaptation, as it enables the creation of compact, language-specific models without full retraining. However, fine-tuning is not without challenges, as modern LLMs have dense knowledge distributions, and improper fine-tuning can lead to forgetting, where the model loses previously learned capabilities.

2. Model Adaptation

Our work builds upon the methodology proposed by Tikhomirov et al. [21] for adapting large language models (LLMs) to target languages, with a focus on addressing the challenges posed by morphologically rich languages like Russian. This approach systematically modifies the model's tokenization and internal representations to better capture language-specific nuances, followed by a calibration phase to optimize performance on target-language tasks. The adaptation process consists of the following steps:

1. **Construction of a new tokenization vocabulary:** A language-specific vocabulary is created to account for the morphological and linguistic characteristics of the target language, augmenting the original tokenizer's vocabulary.
2. **Training embeddings for new vocabulary elements:** New token embeddings are trained to represent the added vocabulary items, ensuring compatibility with the model's architecture and preserving semantic richness.

[4]`https://github.com/RefalMachine/ruadapt`

3. **Alignment of the base language model with the new vocabulary:** The model's parameters are adjusted to align with the updated tokenizer, ensuring seamless integration of the new embeddings into the model's existing knowledge [21].
4. **Transfer of core linguistic knowledge:** Core linguistic knowledge is transferred to the target version of the model using the Learned Embedding Propagation (LEP) method [21].
5. **Calibration of the adapted model:** The adapted model is fine-tuned on task-specific examples in the target language to optimize performance.

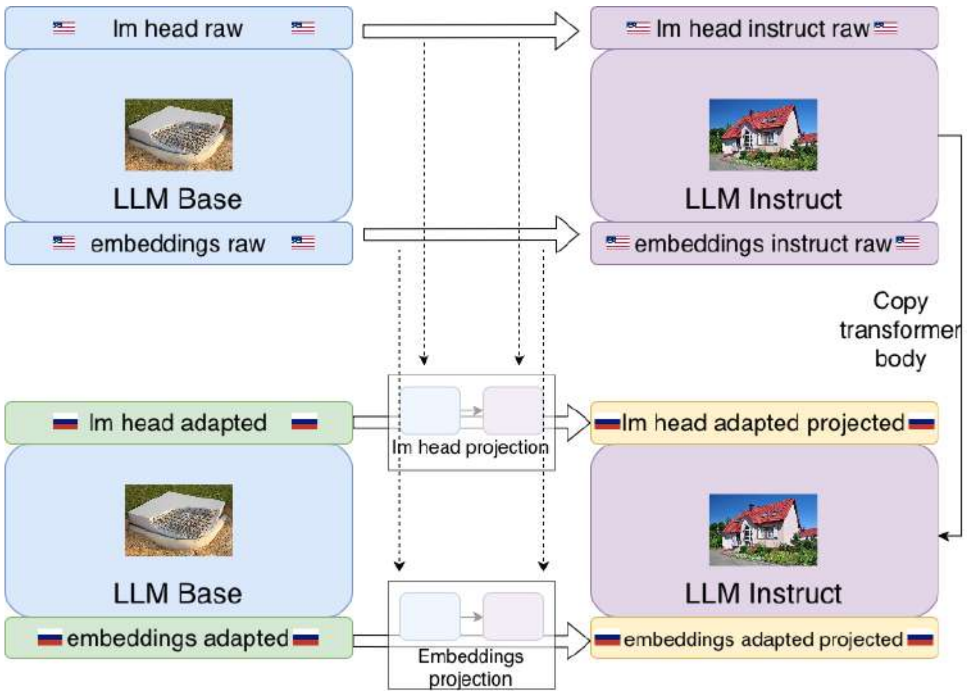


Figure 1. Language adaptation scheme

The fifth step – calibration of the adapted model – is the primary focus of our research. Calibration is critical, as it ensures that the model not only understands the target language's linguistic structures but also performs effectively on downstream tasks. To address this, we explore efficient yet computationally effective calibration methods, comparing classical supervised fine-tuning (SFT) with knowledge distillation. Classical SFT directly optimizes the model on labeled target-language data, while knowledge distillation transfers task-specific knowledge from a larger teacher model to a smaller student model.

In our experiments, we selected a relatively small student model to enable extensive experimentation, thereby providing deeper insights into the benefits of our proposed methodology. Specifically, we used an adapted version of the 4B Qwen3 model, evaluating its performance on both English and Russian-specific benchmarks to assess the trade-offs between efficiency, computational cost, and task performance. Our findings aim to provide practical guidelines for adapting LLMs to low-resource languages with complex morphologies.

3. Distillation Methodology

Knowledge distillation is a well-established technique for transferring knowledge from a larger "teacher" model to a smaller "student" model, enabling efficient model compression while preserving performance [8, 10, 11, 15]. Its flexibility lies not only in the variety of distillation methods but also in the ability to experiment with the teacher model's configuration and outputs.

As previously mentioned, the student model is an adapted version of the 4B Qwen3 following the Learned Embedding Propagation (LEP) step [21]. In this step, we applied the LEP procedure described in [21] to propagate the newly learned token embeddings and linguistic knowledge from the base 4B Qwen3 model to the instruction-tuned 4B variant. The method approximates full re-training with a newly adapted tokenizer through a series of learned linear operators that align embedding spaces between the source (multilingual) and target (Russian-adapted) vocabularies. Before LEP, the base model underwent re-tokenization with an extended Unigram-based vocabulary optimized for Russian morphology and subsequent fine-tuning of embeddings and internal layers on approximately 150 GB of Russian text data constructed as a combination of the rulm [9] and Fineweb-2 [18] corpora. LEP then transferred these updated embeddings and LoRA adapter weights onto the instruction-tuned 4B checkpoint, allowing the resulting model to preserve instruction-following capabilities while achieving native-level Russian tokenization and representation quality. This approach substantially reduces the computational cost of full re-training.

For the "teacher" model, we chose RuadaptQwen3-32B[5], which has been pre-adapted for Russian-language tasks and is currently the largest and most capable publicly available model using the same tokenization.

Simultaneously storing both models in memory for distillation is challenging due to their sizes, and directly training the student model on the teacher's full output distributions is computationally expensive, especially when aiming for an efficient adaptation methodology that balances quality and resource demands.

To address these challenges, we adopt a top-k offline distillation approach [2, 19], which separates the teacher logit generation and student training phases to reduce computational overhead. In the first step, we generate and store the top-k logits produced by the teacher model for each token in the training dataset, where k is a tunable hyperparameter that controls the trade-off between information retention and memory efficiency. This precomputation eliminates the need to load the teacher model during student model training, significantly reducing memory requirements. In the second step, we train the student model using these precomputed logits. To further enhance efficiency, we employ parameter-efficient fine-tuning via Low-Rank Adaptation (LoRA) adapters, which minimize the number of trainable parameters while maintaining performance [12]. The student model is trained on the same dataset used to generate the teacher's logits, ensuring consistency in the knowledge transfer process.

Our training objective combines two loss functions: (1) a classical supervised fine-tuning (SFT) loss, specifically Cross-Entropy between the true tokens and the student's predicted tokens, to calibrate the model for Russian-specific tasks; and (2) a Kullback–Leibler Divergence (KLDivLoss) term that aligns the student's output distribution with the teacher's precomputed top-k logits [10]. This combination enables the student model to learn both from ground-truth data and the teacher's "dark knowledge" – patterns in the teacher's output probabilities that enhance generalization [10]. Our combined loss function adopts the formulation proposed by Raman et al. [19], integrating Cross-Entropy and Kullback–Leibler Divergence to balance task-

[5]`https://huggingface.co/RefalMachine/RuadaptQwen3-32B-Instruct`

specific calibration and knowledge transfer from the teacher model.

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{CE}} + \lambda \left(1 - \exp\left(-\frac{s_i}{t_i + \varepsilon}\right)\right) \mathcal{L}_{\text{KD}}, \tag{1}$$

where:

- $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss between the student predictions and the ground-truth labels;
- $\mathcal{L}_{\text{KD}}$ is the knowledge distillation loss, computed as the KL divergence between the students and teachers probability distributions over the top-K tokens;
- s_i is the student logit corresponding to the ground-truth token at position i;
- t_i is the teacher logit for the ground-truth token at position i (if the token is not in the teachers top-K, t_i is set to 0);
- ε is a small constant added for numerical stability;
- λ is a scaling coefficient (denoted as `loss_multi` in our implementation).

4. Data

The quality of training data is paramount in any training process, whether pre-training or fine-tuning, as it directly influences model performance. Fine-tuning, in particular, is a delicate process, as inconsistencies or contradictions between new training data and the data used for prior training can result in neutral or even negative outcomes. To ensure effective fine-tuning for Russian language adaptation, we selected the `RefalMachine/ruadapt_hybrid_instruct` dataset[6], which comprises approximately 80,000 instruction samples tailored for Russian-language tasks.

This dataset was originally introduced as part of the RuAdapt framework[7] and was specifically designed to serve as calibration data for Russian instruction-tuned models. It consists of high-quality synthetic examples generated by the `Qwen3-235B-A22B` model using prompts drawn from the GrandMaster-PRO-MAX [16] collection. For each prompt, three candidate responses were produced, and the shortest valid response written in Russian and free of non-cyrillic symbols was retained. The resulting dataset captures a broad spectrum of instruction-following behaviors while maintaining linguistic consistency with the target language.

5. Evaluation

For evaluation in our experiments, we utilized the `llmtf` framework[8], an open-source toolkit designed to assess the performance of instruction-tuned language models in both few-shot and zero-shot scenarios. This framework supports flexible evaluation across diverse tasks, enabling comprehensive analysis of model capabilities on Russian-specific benchmarks. The `llmtf` framework also standardizes prompt templates and scoring procedures, ensuring reproducibility of the reported results.

To obtain a balanced picture of model performance, we evaluated the models in the zero-shot setting on a diverse suite of datasets that cover multiple linguistic and reasoning abilities. Specifically, the following benchmarks were used:

[6]https://huggingface.co/datasets/RefalMachine/ruadapt_hybrid_instruct
[7]https://github.com/RefalMachine/ruadapt
[8]https://github.com/RefalMachine/llmtf_open

- **NEREL** [13], a Russian named entity recognition benchmark derived from news and Wikipedia texts, testing the models ability to identify and classify named entities in context.
- **Summ**[9], a summarization dataset based on Russian news articles, assessing the models capacity for text compression and important information extraction.
- **MultiQ** and **USE** (from MERA [5]), multi-domain question-answering and semantic understanding benchmarks that evaluate reasoning and general comprehension abilities across diverse Russian topics.
- **Copy**, a diagnostic test of generation robustness that measures the models tendency to produce repetitive or degenerate outputs under constrained input prompts.
- **FLORES** (ru–en and en–ru) [6], a multilingual machine translation benchmark used to measure cross-lingual generalization and translation consistency between Russian and English.
- **enMMLU** and **ruMMLU**, multilingual general knowledge and reasoning benchmarks adapted from the Massive Multitask Language Understanding dataset, providing a standardized measure of factual recall and reasoning accuracy across academic domains.
- **IFEval** (en and ru versions) [22], a meta-evaluation suite for instruction-following behavior that quantifies how well a model adheres to explicit task instructions and constraints.
- **ruOpinionNE** [14], a sentiment analysis dataset focusing on Russian social media and news, testing contextual polarity and stance detection.
- **ruParam** [7], a benchmark for paraphrase and semantic similarity detection in Russian, designed to measure semantic coherence and lexical flexibility.

Together, these benchmarks cover a wide range of linguistic competencies, including named entity recognition, summarization, question answering, translation, reasoning, and adherence to instructions. This diversity allows us to evaluate both general language understanding and the effectiveness of Russian-specific adaptation. All evaluations were conducted in the zero-shot setting, using the default templates provided by `llmtf`, to ensure fair comparison across models and reproducibility of results.

6. Experiments

The distillation process is undoubtedly more computationally demanding. To assess its effectiveness, we conducted a series of experiments comparing classical supervised fine-tuning (SFT) with knowledge distillation during the calibration stage of model adaptation. Our central question is whether knowledge distillation can improve model performance over classical SFT, and whether the potential gains justify its additional cost.

6.1. Supervised Fine-Tuning

For the supervised fine-tuning (SFT) stage, we employed LoRA adapters [12]. Building on prior experiments, we fixed the LoRA rank at 128 and treated LoRA α together with the learning rate as tunable hyperparameters. Since our distillation approach introduces two additional hyperparameters – top-K and λ – we first identified the optimal values of LoRA α and the learning rate using classical SFT, and only then extended the setup to knowledge distillation.

[9]`https://huggingface.co/datasets/IlyaGusev/gazeta`

Table 1. Comparison of different hyperparameter configurations
(a – LoRA alpha, lr – learning rate) across multiple evaluation benchmarks

Config a	lr	**mean**	**NEREL**	**Summ**	**MultiQ**	**USE**	**flores**	**copy**	**ru babllong**	**en IFEval**	**ru IFEval**	**en MMLU**	**ru MMLU**	**ru opinionqa**	**ru param**
64	1e-4	**0.484**	**0.489**	**0.225**	**0.200**	0.034	**0.507**	0.940	0.504	**0.712**	**0.636**	0.705	0.621	0.088	**0.632**
	2e-5	0.468	0.478	0.154	0.186	0.029	0.504	**0.985**	0.490	0.688	0.560	0.706	0.620	0.088	0.600
	5e-5	0.483	0.475	0.223	0.195	**0.041**	**0.507**	0.950	**0.509**	0.704	0.621	**0.708**	**0.624**	**0.097**	0.631
128	1e-4	**0.483**	**0.495**	**0.224**	**0.194**	**0.037**	**0.508**	0.945	0.504	**0.717**	0.601	**0.706**	0.620	**0.092**	0.634
	2e-5	0.472	0.480	0.181	0.183	0.028	0.506	**0.980**	0.494	0.684	0.582	**0.706**	**0.622**	0.084	0.603
	5e-5	0.482	0.478	0.223	0.193	0.031	**0.508**	0.955	**0.507**	0.693	**0.608**	0.705	0.620	0.091	**0.653**
256	1e-4	**0.490**	**0.497**	**0.223**	**0.200**	**0.043**	**0.508**	0.940	0.510	**0.726**	**0.636**	**0.707**	**0.623**	**0.106**	**0.647**
	2e-5	0.479	0.477	0.205	0.185	0.025	0.507	**0.980**	0.498	0.702	0.584	0.706	0.621	0.100	0.634
	5e-5	0.477	0.479	**0.223**	0.195	0.031	0.507	0.955	**0.514**	0.682	0.590	0.705	0.618	0.095	0.609
512	1e-4	**0.494**	**0.503**	0.222	**0.195**	**0.039**	**0.507**	0.950	**0.516**	**0.738**	**0.617**	0.704	**0.623**	**0.112**	0.692
	2e-5	0.475	0.475	0.218	0.189	0.028	0.506	**0.970**	0.509	0.710	0.584	**0.707**	0.620	0.088	0.568
	5e-5	0.488	0.484	**0.224**	0.191	0.036	0.506	**0.970**	0.500	0.713	0.608	0.705	0.621	0.096	**0.694**

From a performance perspective, we selected the four most promising configurations for further investigation of the distillation approach. These configurations were chosen based on their average performance across a diverse set of benchmarks, ensuring robust generalization. The selected configurations are: (1) $a = 128$, lr$=1e-4$, (2) $a = 256$, lr$=1e-4$, (3) $a = 512$, lr$=1e-4$, and (4) $a = 512$, lr$=5e-5$.

6.2. Knowledge Distillation

We adopted the LoRA rank, α, and learning rate values from the selected SFT configurations, while treating top-K and λ as the main hyperparameters of interest in this series of experiments. Each chosen SFT setup was compared against eight distillation variants, with $\lambda \in \{0.1, 0.2, 0.5, 1\}$ and top-$K \in \{10, 100\}$. The top-K parameter controls the number of top-ranked teacher model predictions used in the distillation process, while λ balances the contribution of the distillation loss against the supervised loss. To ensure a fair comparison, we maintained identical training conditions (e.g., batch size, training epochs, and dataset) across SFT and distillation experiments.

The results, presented in Tables 2 to 5 (Table 6 shows more detailed results), demonstrate that knowledge distillation consistently enhances model performance over the SFT baselines. The most significant improvement was observed for the SFT baseline with $\alpha = 128$ and lr=1e-4. When distilled with top-K=100 and $\lambda = 0.5$, this model achieved an aggregate score of 0.503, a 4.22% increase over its SFT counterpart. This highlights the potential of distillation to further refine already fine-tuned models.

The Influence of top-K. A clear pattern emerges when comparing top-K values: using a larger context from the teacher model (top-100) generally yields superior results compared to a smaller one (top-10). For instance, with the $\alpha = 128$ baseline, the average improvement for top-100 variants was 3.27%, compared to 2.82% for top-10. However, this performance gain comes at a significant computational cost. Storing precomputed logits for top-100 required 62 GB of memory, whereas top-10 required only 7 GB – an 8.9-fold reduction. This trade-off makes top-10 a resource-efficient option for achieving modest gains, while top-100 is preferable when maximizing performance is the priority and resources permit.

Tuning the Distillation Weight λ. Our experiments show that $\lambda = 0.2$ emerges as a robust and generally effective choice for the distillation weight. It delivered the highest performance gains in the majority of our tested configurations. However, the single best result (4.22% growth) was achieved with $\lambda = 0.5$ in the $\alpha = 128$ setup. This suggests that while $\lambda = 0.2$ is a strong starting point for tuning, the optimal value can vary depending on other hyperparameters.

Table 2. SFT a=128, lr=1e-4

Base Config	Variant	Aggregate Score	Growth, %
SFT a=128, lr=1e-4	—	0.483	—
a=128, lr=1e-4, top-10	λ=0.1	0.498	3.17
	λ=0.2	0.497	2.95
	λ=0.5	0.495	2.56
	λ=1	0.495	2.58
Avg. Growth Top-10			**2.82**
Max. Growth Top-10			**3.17**
a=128, lr=1e-4, top-100	λ=0.1	0.498	3.20
	λ=0.2	0.498	3.15
	λ=0.5	***0.503***	***4.22***
	λ=1	0.495	2.50
Avg. Growth Top-100			**3.27**
Max. Growth Top-100			**4.22**

Table 3. SFT a=256, lr=1e-4

Base Config	Variant	Aggregate Score	Growth, %
SFT a=256, lr=1e-4	—	0.490	—
a=256, lr=1e-4, top-10	λ=0.1	0.500	2.16
	λ=0.2	***0.503***	***2.67***
	λ=0.5	0.500	2.13
	λ=1	0.500	2.08
Avg. Growth Top-10			**2.26**
Max. Growth Top-10			**2.67**
a=256, lr=1e-4, top-100	λ=0.1	0.497	1.57
	λ=0.2	0.500	2.04
	λ=0.5	0.499	1.84
	λ=1	0.496	1.33
Avg. Growth Top-100			**1.70**
Max. Growth Top-100			**2.04**

Table 4. SFT a=512, lr=1e-4

Base Config	Variant	Aggregate Score	Growth, %
SFT a=512, lr=1e-4	—	0.494	—
a=512, lr=1e-4, top-10	λ=0.1	0.496	0.38
	λ=0.2	0.502	1.63
	λ=0.5	0.496	0.41
	λ=1	0.499	1.00
Avg. Growth Top-10			**0.86**
Max. Growth Top-10			**1.63**
a=512, lr=1e-4, top-100	λ=0.1	0.502	1.66
	λ=0.2	***0.503***	***1.83***
	λ=0.5	0.501	1.46
	λ=1	0.501	1.40
Avg. Growth Top-100			**1.59**
Max. Growth Top-100			**1.83**

Table 5. SFT a=512, lr=5e-5

Base Config	Variant	Aggregate Score	Growth, %
SFT a=512, lr=5e-5	—	0.488	—
a=512, lr=5e-5, top-10	λ=0.1	0.491	0.61
	λ=0.2	0.494	1.06
	λ=0.5	0.491	0.56
	λ=1	0.489	0.18
Avg. Growth Top-10			**0.60**
Max. Growth Top-10			**1.06**
a=512, lr=5e-5, top-100	λ=0.1	0.496	1.56
	λ=0.2	0.497	1.80
	λ=0.5	0.494	1.16
	λ=1	0.488	−0.14
Avg. Growth Top-100			**1.10**
Max. Growth Top-100			**1.80**

7. Limitations

While the proposed approach demonstrates consistent improvements in performance and efficiency over standard knowledge distillation baselines, several limitations remain that should be addressed in future work.

First, the observed performance gains are relatively modest (approximately 2–4% across benchmarks). Although the experiments show improvements, they suggest that further optimization of the distillation procedure is necessary to fully exploit the potential of cross-model knowledge transfer.

Second, the current study investigates only a single teacher–student configuration (`Qwen3-32B` $\rightarrow$ `Qwen3-4B`), as the 32B teacher model is currently the largest available model sharing the same tokenizer. Nevertheless, we believe that the proposed methodology is generalizable and can be applied to other model combinations, potentially leading to greater improvements.

Third, although the method is designed to improve training efficiency, we did not include quantitative measurements such as training throughput, total duration, or GPU-hour cost per epoch. Our analysis focused primarily on algorithmic efficiency and qualitative reductions in computational overhead (e.g., precomputed logits, use of LoRA). A more detailed profiling of hardware utilization and memory footprint will be presented in future work.

Finally, all experiments were conducted within a single computational environment and evaluated on a fixed set of Russian-language benchmarks described in Section 5.

Conclusion

This study evaluated the effectiveness of knowledge distillation (KD) relative to classical supervised fine-tuning (SFT) during the calibration phase of large language model adaptation for Russian. Our experiments, which distilled knowledge from a Russian-adapted RuadaptQwen3-32B teacher model into a 4B Qwen3 student, conclusively demonstrate that KD is a superior approach for enhancing model performance.

A key finding from our research is the fundamental trade-off between performance and computational efficiency, governed by the top-K hyperparameter. Incorporating a broader knowledge context from the teacher (top-100) consistently yielded superior results, achieving average performance gains of up to 3.27%. This performance, however, comes at a significant cost, requiring 62 GB of memory for the precomputed logits. Conversely, a top-10 configuration emerges as a viable, resource-efficient alternative, providing modest gains with a substantially smaller memory footprint of just 7 GB.

Our analysis also provides practical tuning guidelines. We found that a distillation weight of $\lambda = 0.2$ offers a robust and effective starting point across most configurations. Furthermore, we observed that the benefits of distillation are most pronounced for models with a lower LoRA α, indicating that simpler student models with less adaptive capacity gain the most from the teacher's guidance.

Our findings provide a practical roadmap for adapting large language models to morphologically complex languages like Russian, demonstrating that knowledge distillation effectively enhances the adaptation pipeline while emphasizing the importance of strategically balancing hyperparameters such as top-K, λ, and α with the available computational budget to achieve optimal performance.

Acknowledgements

The research was supported by the Russian Science Foundation, project No. 25-11-00191, `https://rscf.ru/project/25-11-00191/`.

The research was carried out using the MSU-270 supercomputer of Lomonosov Moscow State University.

References

1. Achiam, J., Adler, S., Agarwal, S., *et al.*: GPT-4 Technical Report. arXiv e-prints pp. arXiv–2303 (2023). `https://doi.org/10.48550/arXiv.2303.08774`

2. Anshumann, A., Zaidi, M.A., Kedia, A., *et al.*: Sparse logit sampling: Accelerating knowledge distillation in LLMs. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025. pp. 18085–18108. Association for Computational Linguistics (2025). `https://doi.org/10.18653/v1/2025.`

acl-long.885

3. DeepSeek-AI: Deepseek-v3 technical report (2024), https://arxiv.org/abs/2412.19437

4. Dubey, A., Jauhri, A., Pandey, A., *et al.*: The Llama 3 Herd of Models. arXiv e-prints pp. arXiv–2407 (2024). https://doi.org/10.48550/arXiv.2407.21783

5. Fenogenova, A., Chervyakov, A., Martynov, N., *et al.*: MERA: A Comprehensive LLM Evaluation in Russian. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 9920–9948. Association for Computational Linguistics (2024). https://doi.org/10.18653/v1/2024.acl-long.534

6. Goyal, N., Gao, C., Chaudhary, V., *et al.*: The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. Transactions of the Association for Computational Linguistics 10, 522–538 (2022). https://doi.org/10.1162/tacl_a_00474

7. Grashchenkov, P.V., Pasko, L.I., Studenikina, K.A., *et al.*: Russian parametric corpus RuParam. Journal Scientific and Technical of Information Technologies, Mechanics and Optics 158(6), 991 (2024). https://doi.org/10.17586/2226-1494-2024-24-6-991-998

8. Gu, Y., Dong, L., Wei, F., *et al.*: MiniLLM: Knowledge distillation of large language models. In: The Twelfth International Conference on Learning Representations, ICLR 2024. OpenReview.net (2024), https://openreview.net/forum?id=5h0qf7IBZZ

9. Gusev, I.: rulm: A toolkit for training neural language models (2023)

10. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. Stat 1050, 9 (2015)

11. Hsieh, C.Y., Li, C.L., Yeh, C.K., *et al.*: Distilling step-by-step! Outperforming larger language models with less training data and smaller model sizes. In: Findings of the Association for Computational Linguistics, ACL 2023. pp. 8003–8017. Association for Computational Linguistics (2023), https://doi.org/10.18653/v1/2023.findings-acl.507

12. Hu, E.J., Shen, Y., Wallis, P., *et al.*: LoRA: Low-Rank Adaptation of Large Language Models. In: The Tenth International Conference on Learning Representations, ICLR 2022. OpenReview.net (2022), https://openreview.net/forum?id=nZeVKeeFYf9

13. Loukachevitch, N., Artemova, E., Batura, T., *et al.*: NEREL: A Russian Dataset with Nested Named Entities, Relations and Events. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP 2021. pp. 876–885. INCOMA Ltd. (2021), https://aclanthology.org/2021.ranlp-1.100

14. Loukachevitch, N., Tkachenko, N., Lapanitsyna, A., *et al.*: RuOpinionNE-2024: Extraction of Opinion Tuples from Russian News Texts. In: Proceedings of the International Conference "Dialogue". vol. 2025 (2025)

15. Men, X., Xu, M., Zhang, Q., *et al.*: ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. In: Findings of the Association for Computational Linguistics, ACL 2025. pp. 20192–20204. Association for Computational Linguistics (2025), https://aclanthology.org/2025.findings-acl.1035/

16. Nikolich, A., Korolev, K., Bratchikov, S., *et al.*: Vikhr: Constructing a state-of-the-art bilingual open-source instruction-following large language model for Russian. In: Proceedings of the Fourth Workshop on Multilingual Representation Learning, MRL 2024. pp. 189–199 (2024). `https://doi.org/10.18653/v1/2024.mrl-1.15`

17. Ouyang, L., Wu, J., Jiang, X., *et al.*: Training language models to follow instructions with human feedback. In: Proceedings of the 36th Int. Conf. on Neural Information Processing Systems. vol. 35, pp. 27730–27744 (2022). `https://doi.org/10.5555/3600270.3602281`

18. Penedo, G., Kydlíček, H., Sabolčec, V., *et al.*: FineWeb2: One Pipeline to Scale Them All–Adapting Pre-Training Data Processing to Every Language. arXiv e-prints pp. arXiv–2506 (2025). `https://doi.org/10.48550/arXiv.2506.20920`

19. Raman, M., Mani, P., Liang, D., *et al.*: For distillation, tokens are not all you need. In: NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following (2023)

20. Team, Q.: Qwen3 technical report (2025), `https://arxiv.org/abs/2505.09388`

21. Tikhomirov, M., Chernyshev, D.: Facilitating large language model Russian adaptation with learned embedding propagation. Journal of Language and Education 10(4), 130–145 (2024). `https://doi.org/10.17323/jle.2024.22224`

22. Zhou, J., Lu, T., Mishra, S., *et al.*: Instruction-following evaluation for large language models. CoRR (2023). `https://doi.org/10.48550/arXiv.2311.07911`

Appendix

Table 6. Comparison of different hyperparameter configurations (a – LoRA alpha, lr – learning rate) across multiple evaluation benchmarks

a	lr	top	λ	mean	NEREL	Summ	MultiQ	USE	flores	copy	ru babilong	en IFEval	ru IFEval	en MMLU	ru MMLU	ru opinionne	ru param
128	1e-4	10	0.1	0.498	0.485	0.224	0.219	0.089	0.508	0.950	0.508	0.721	0.628	0.708	0.624	0.107	0.704
			0.2	0.497	0.494	0.226	0.223	0.113	0.509	0.940	0.512	0.719	0.627	0.709	0.621	0.104	0.665
			0.5	0.495	0.489	0.225	0.226	0.134	0.506	0.940	0.503	0.717	0.638	0.707	0.620	0.097	0.635
			1	0.495	0.479	0.224	0.226	0.162	0.505	0.945	0.498	0.708	0.612	0.708	0.622	0.105	0.644
		100	0.1	0.498	0.482	0.226	0.222	0.116	0.510	0.950	0.517	0.713	0.623	0.708	0.623	0.102	0.685
			0.2	0.498	0.488	0.225	0.225	0.126	0.508	0.945	0.515	0.713	0.634	0.706	0.623	0.100	0.666
			0.5	0.503	0.495	0.224	0.234	0.165	0.506	0.955	0.518	0.726	0.603	0.708	0.621	0.107	0.679
			1	0.495	0.494	0.225	0.230	0.161	0.503	0.960	0.500	0.702	0.603	0.708	0.619	0.093	0.636
256	1e-4	10	0.1	0.502	0.487	0.224	0.225	0.087	0.508	0.945	0.512	0.750	0.632	0.706	0.619	0.107	0.725
			0.2	0.503	0.496	0.225	0.221	0.117	0.509	0.955	0.515	0.708	0.645	0.707	0.621	0.108	0.709
			0.5	0.500	0.489	0.224	0.225	0.147	0.507	0.950	0.500	0.728	0.632	0.706	0.622	0.100	0.671
			1	0.500	0.492	0.226	0.219	0.170	0.505	0.960	0.493	0.726	0.619	0.706	0.623	0.101	0.657
		100	0.1	0.497	0.483	0.225	0.221	0.102	0.509	0.960	0.515	0.719	0.643	0.705	0.621	0.093	0.670
			0.2	0.500	0.492	0.224	0.221	0.155	0.508	0.960	0.507	0.721	0.623	0.706	0.622	0.107	0.649
			0.5	0.499	0.498	0.225	0.218	0.164	0.506	0.960	0.490	0.710	0.630	0.708	0.621	0.111	0.642
			1	0.496	0.490	0.225	0.230	0.159	0.504	0.965	0.498	0.702	0.621	0.705	0.616	0.099	0.636
512	1e-4	10	0.1	0.496	0.491	0.221	0.218	0.112	0.507	0.965	0.506	0.721	0.628	0.706	0.622	0.103	0.642
			0.2	0.502	0.510	0.227	0.222	0.109	0.508	0.960	0.522	0.730	0.619	0.707	0.622	0.093	0.693
			0.5	0.496	0.498	0.225	0.219	0.121	0.506	0.960	0.511	0.726	0.614	0.707	0.625	0.113	0.619
			1	0.499	0.495	0.225	0.225	0.151	0.507	0.960	0.506	0.726	0.617	0.706	0.622	0.112	0.630
		100	0.1	0.502	0.500	0.223	0.225	0.112	0.508	0.960	0.521	0.710	0.638	0.705	0.621	0.101	0.700
			0.2	0.503	0.502	0.224	0.223	0.125	0.509	0.960	0.507	0.728	0.638	0.705	0.620	0.118	0.676
			0.5	0.501	0.498	0.226	0.225	0.151	0.507	0.965	0.491	0.721	0.628	0.706	0.621	0.113	0.659
			1	0.501	0.490	0.228	0.234	0.167	0.503	0.960	0.507	0.726	0.610	0.707	0.619	0.106	0.650
	5e-5	10	0.1	0.491	0.482	0.225	0.216	0.056	0.506	0.955	0.508	0.719	0.623	0.707	0.622	0.115	0.653
			0.2	0.492	0.480	0.226	0.221	0.076	0.507	0.955	0.501	0.701	0.614	0.707	0.623	0.120	0.660
			0.5	0.491	0.468	0.228	0.220	0.104	0.507	0.960	0.506	0.715	0.628	0.706	0.620	0.104	0.618
			1	0.489	0.467	0.230	0.225	0.105	0.504	0.955	0.497	0.723	0.623	0.705	0.618	0.102	0.605
		100	0.1	0.496	0.487	0.224	0.220	0.073	0.507	0.960	0.513	0.713	0.628	0.705	0.619	0.113	0.685
			0.2	0.497	0.486	0.226	0.225	0.093	0.507	0.955	0.502	0.721	0.634	0.706	0.618	0.118	0.671
			0.5	0.494	0.474	0.229	0.222	0.102	0.506	0.960	0.503	0.717	0.617	0.705	0.617	0.120	0.646
			1	0.488	0.484	0.229	0.231	0.112	0.503	0.965	0.486	0.717	0.595	0.706	0.617	0.093	0.601

DOI: 10.14529/jsfi250402

Supercomputing Co-Design for Solving Ill-Posed Linear Inverse Problems Using Iterative Algorithms

Alexander S. Antonov[1], *Vladimir V. Voevodin*[1],
Dmitry V. Lukyanenko[2]

The paper considers an approach to applying the ideas of supercomputing co-design for the effective use of arbitrary multiprocessor computing systems with distributed memory when using iterative regularization algorithms to solve ill-posed linear inverse problems, which are reduced to solving large overdetermined systems of linear algebraic equations with a dense matrix. The proposed methodology allows for a large number of algorithms to select the best virtual topology of processes (in terms of parallelization efficiency) for solving problems of the class under consideration within the allocated resources of the supercomputer system being used.

Keywords: supercomputing co-design, parallelization efficiency, parallelism, autotuning, AlgoWiki, inverse problem, iterative regularization, conjugate gradient method.

Introduction

In many cases, applied inverse problems are linear and come down to the need to solve large overdetermined systems of linear algebraic equations of the form

$$A\,x = b_\delta \tag{1}$$

with a dense matrix. Here $A \in \mathbb{R}^{M\times N}$, $x \in \mathbb{R}^N$, $b_\delta \in \mathbb{R}^M$. Usually $M \geqslant N$ and instead of the exact right-hand side b its approximation b_δ is known, measured in an experiment with an error δ, i.e., $\|b - b_\delta\| \leqslant \delta$. The matrix can also be specified with an error, i.e., instead of the exact matrix A, its approximation A_h is known, where $\|A - A_h\| \leqslant h$ (hereafter, all vector norms are assumed to be Euclidean, and matrix norms – Frobenius, unless otherwise stated).

Often, such problems are ill-posed, and to solve them, it is necessary to use regularizing algorithms (see, for example, the classical work [10]). One of the most common classes of regularizing algorithms are iterative regularization algorithms. Most iterative regularization algorithms for solving problems of the form (1) contain only the following computational operations that allow for parallel implementation:

1. multiplication of a matrix of size $M \times N$ by a vector of size N (which requires performing $M \cdot N$ multiplications and $M \cdot (N-1)$ additions – in the sum of $M \cdot (2N-1)$ arithmetic operations);
2. multiplication of a transposed matrix of size $N \times M$ by a vector of size M (requires performing $N \cdot (2M-1)$ arithmetic operations);
3. scalar product of vectors of size N (N multiplications and $N-1$ additions – in the sum of $2N-1$ arithmetic operations) or vectors of size M (requires performing $2M-1$ arithmetic operations);
4. adding vectors of size N or M or multiplying them by a number (requires performing N and M arithmetic operations, respectively).

[1]Research Computing Center, Lomonosov Moscow State University, Moscow, Russia
[2]Department of Mathematics, Faculty of Physics, Lomonosov Moscow State University, Moscow, Russia

Two classical iterative regularization algorithms are given below as examples of such algorithms: the conjugate gradient method and the Nesterov accelerated method (see the description of the features of these algorithms in [3, 4] and [5, 7], respectively):

Algorithm 1: Conjugate gradient method.

```
Data: $A$, $b_\delta$, $\delta$
Result: $x$
$s \leftarrow 1$
$x \leftarrow 0$
$p \leftarrow 0$
while $\|Ax - b_\delta\|^2 > \delta^2$ do
    if $s = 1$ then
        $r \leftarrow A^T(Ax - b_\delta)$
    else
        $r \leftarrow r - \frac{q}{(p,q)}$
    end
    $p \leftarrow p + \frac{r}{(r,r)}$
    $q \leftarrow A^T(Ap)$
    $x \leftarrow x - \frac{p}{(p,q)}$
    $s \leftarrow s + 1$
end
```

Algorithm 2: Nesterov accelerated method.

```
Data: $A$, $b_\delta$, $\delta$, $\beta > -1$
Result: $x$
$s \leftarrow 1$
$x \leftarrow 0$
$p \leftarrow 0$
while $\|Ax - b_\delta\|^2 > \delta^2$ do
    if $s = 1$ then
        $x \leftarrow A^T b_\delta$
    else
        $x_{previous} \leftarrow x$
        $p \leftarrow x + \frac{s-2}{s-1+\beta}(x - x_{previous})$
        $x \leftarrow p - A^T(Ap - b_\delta)$
    end
    $s \leftarrow s + 1$
end
```

Note. Sometimes in the literature, these algorithms are presented in a form that assumes that the matrix A of the system (1) is symmetric and positive definite. In this case, the corresponding algorithm implementations do not include the operation of multiplying the transposed matrix by a vector. This type can also be used in the specified implementations of the algorithms if someone replaces $A := A^T A$ and $b_\delta := A^T b_\delta$. But this replacement is not constructive when using algorithms in practice to solve large problems. This is due to the following fact. The computational complexity of the written algorithms at each iteration is $O(MN)$ arithmetic operations. At the same time, the number of iterations s_{iter} that need to be performed is often significantly less than the number of unknowns N – when developing new algorithms for solving problems of the class under consideration, the main motivation is to increase the convergence rate, i.e., to reduce the number of iterations needed to achieve convergence. Thus, the computational complexity of the presented algorithms can often be estimated as $O(MN)$ provided $s_{iter} \ll N$. This is one of the main advantages of using iterative algorithms for solving problems like (1). These substitutions reduce the computational complexity of each iteration by about 2 times, but they require the preliminary execution of $O(MN^2)$ arithmetic operations, which negates the practical advantages of using iterative solution algorithms.

The approaches to the construction of parallel algorithms and their software implementation used in linear algebra operations in such iterative algorithms are well known. They are usually based on a two-dimensional partition of the matrix A into blocks. In this case, a Cartesian virtual topology based on a two-dimensional rectangular process grid is usually used, – in the address space of each computing process, its own block A_{part} of the original matrix A is stored (see Fig. 1). The size of such a process grid is usually chosen intuitively – most often, the number of rows and columns of this process grid is made as uniform as possible. However, such a choice may not be optimal (in terms of parallelization efficiency) in many special cases, as it depends on various factors, the key ones of which are the following.

First, the optimal choice of the ratio of the sizes of the process grid depends on the problem being solved, namely, on the sizes M and N corresponding to the matrix. If, in the case of a square matrix, it is optimal to choose the same sizes of a two-dimensional process grid, then in the case of significantly different values of M and N, which is quite common in solving applied problems, such a choice will be far from optimal.

Secondly, the optimal size ratio of the process grid depends on the architecture of the multiprocessor computing system used. In particular, there may be a significant dependence on the resources allocated for running the application, which may change with each subsequent launch. At the same time, even if the application is running on the same number of computing nodes, each new launch may result in a set of computing nodes with a significantly different communication profile from the previous set (dedicated processes are located differently in the physical topology of a multiprocessor system).

Therefore, the purpose of this work is to create a methodology that will allow, within the framework of solving the problem of supercomputing co-design, to automatically match the size of the problem being solved and the topology of the computing resources allocated at application launch with the optimal size of a two-dimensional process grid defining the virtual topology of processes in a parallel software implementation of the algorithm.

Solving ill-posed linear inverse problems using iterative algorithms is extremely important for many applied problems in science and technology, while at the same time allowing us to demonstrate the basic ideas of supercomputing co-design [2]. Therefore, it has been chosen as a significant reference problem requiring high-performance computing facilities that test the approaches, methods, tools, techniques and recommendations implemented by the authors of the project to develop the fundamentals of supercomputing co-design based on the descriptions of algorithms in the AlgoWiki Open Encyclopedia of Parallel Algorithmic Features [11].

In connection with the above, the structure of this work will be as follows. Section 1 describes an approach to evaluating the parallelization efficiency of an algorithm for solving a problem of type (1) with fixed dimensions using the allocated resources of a supercomputer system. Section 2 presents a technique for matching the size of the Cartesian virtual topology of processes with the size of the problem being solved, the selected algorithm, and the allocated computing resources in the form of a practice-oriented algorithm. Section 3 demonstrates some of the results of numerical experiments. Section 4 discusses some of the features of the proposed methodology.

1. An Approach to Evaluating the Parallelization Efficiency of an Algorithm within the Framework of the Used Supercomputer System

First, subsection 1.1 will describe the proposed structure for storing problem data in the distributed address space of computing processes. Taking into account the specifics of the problem being solved, the data distribution will be used in accordance with the Cartesian virtual topology of two-dimensional grid type processes. Then, in subsections 1.2–1.5, estimates of the total running time of computational operations that allow for parallel implementation will be presented. Finally, subsection 1.6 will provide a formula for calculating the parallelization effi-

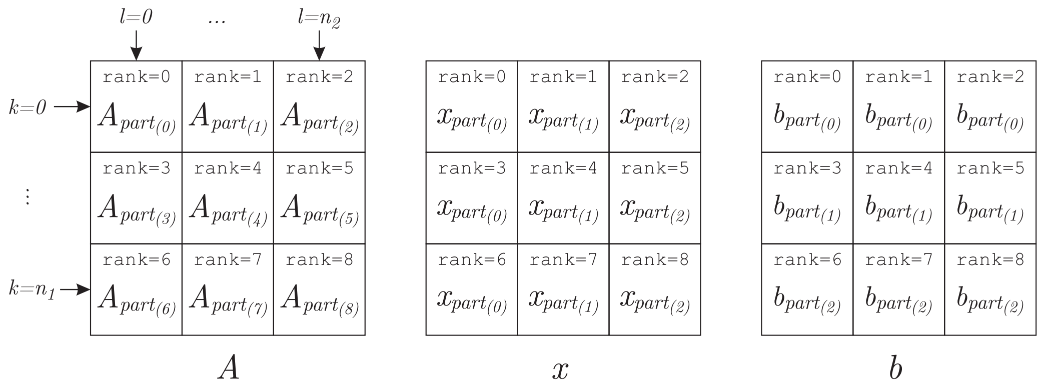


Figure 1. An example of the distribution of data across nine computational processes that form a two-dimensional grid 3×3

ciency of the proposed software implementation of the algorithm under consideration for solving the problem (1), depending on the size of the selected virtual process topology. MPI [1] technology is assumed to be a parallel programming technology for software implementations of the studied algorithms on a computer with distributed memory.

1.1. Distribution of Data by Processes Involved in Calculations

It is assumed that all task data is distributed over the address space of n computing processes involved in the calculations as follows.

Each computing process has its own identifier/number $\texttt{rank} \in \overline{0,\ldots,n-1}$. The matrix A of dimensions $M \times N$ is divided among these processes in two dimensions (see Fig. 1) for blocks $A_{part(\texttt{rank})}$ of sizes $M_{part(k)} \times N_{part(l)}$.

It is assumed that the processes form a two-dimensional process grid of size $n_1 \times n_2$ (with $n_1 \cdot n_2 = n$). Therefore, the process number `rank` is associated with the indexes k and l, which determine the coordinates of the process in the two-dimensional process grid as follows:

$$k = \left\lfloor \frac{\texttt{rank}}{n_2} \right\rfloor, \quad l = \texttt{rank} - \left\lfloor \frac{\texttt{rank}}{n_2} \right\rfloor \cdot n_2.$$

Or, if it is necessary to recalculate `rank` by k and l:

$$\texttt{rank} = k \cdot n_2 + l.$$

We also note that

$$\sum_{k=0}^{n_1-1} M_{part(k)} = M, \quad \sum_{l=0}^{n_2-1} N_{part(l)} = N.$$

The vector x is divided into n_2 parts $x_{part(l)}$, $l = \overline{0, n_2-1}$, of sizes $N_{part(l)}$. In this case, the part $x_{part(l)}$ for a fixed index l is stored on all processes in the column of the process grid with index l (see Fig. 1), that is, on processes with the coordinate $(\cdot, l)$ in a two-dimensional process grid.

The vector b is divided into n_1 parts $b_{part(k)}$, $k = \overline{0, n_1-1}$, of sizes $M_{part(k)}$. In this case, the part $b_{part(k)}$ for a fixed index k is stored on all processes in the row of the process grid with index k (see Fig. 1), that is, on processes with the coordinate $(k, \cdot)$ in a two-dimensional process grid.

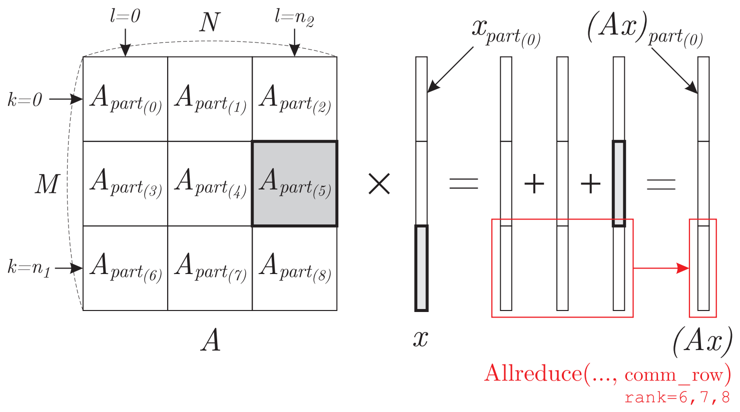


Figure 2. A parallel algorithm for matrix-vector multiplication in the case of two-dimensional matrix partition into blocks. The figure shows the case of data distribution over nine computing processes that form a two-dimensional grid 3×3

Thus, it is assumed that each computing process contains one of the blocks $A_{part_{(\cdot)}}$ of the matrix A, as well as one of the parts $x_{part_{(\cdot)}}$ and $b_{part_{(\cdot)}}$ of the vectors x and b, respectively.

1.2. Estimation of the Implementation Time of a Parallel Matrix-Vector Multiplication Algorithm in the Case of Two-Dimensional Matrix Partition Into Blocks

With the formulated method of storing data on computational processes, the result of multiplying the matrix A of size $M \times N$ by the vector x of size N will be a vector (Ax), which will be distributed over various computational processes in parts $(Ax)_{part_{(\cdot)}}$. At the same time, the distribution structure of this vector for various processes should correspond to the distribution structure of the vector b (see Fig. 1). Part of the vector (Ax) can be calculated using the following formula:

$$(Ax)_{part_{(k)}} = \sum_{l=0}^{n_2-1} A_{part_{(k \cdot n_2 + l)}} x_{part_{(l)}}, \quad k = \overline{0, \dots, n_1 - 1}. \tag{2}$$

An example explaining this formula is provided in Fig. 2.

Each process involved in the calculations can compute the term $A_{part_{(k \cdot n_2 + l)}} x_{part_{(l)}}$ for its pair of values (k, l), independently of similar calculations performed by other processes. That is, the terms in the formula (2) can be calculated in parallel. Then the terms located on the processes of the row of the process grid with index k should be summed up, and the result – vector $(Ax)_{part_{(k)}}$ – should be placed on all processes of this row of the process grid. The organization of the interaction of processes entails overhead costs for receiving/transmitting messages containing the results of intermediate calculations through the physical communication environment of a multiprocessor computing system.

Let us estimate the total running time of this parallel algorithm, taking into account the overhead costs of organizing the interaction of computing processes.

First, we note that the running time of the sequential implementation of the operation of multiplying the matrix A of size $M \times N$ by the vector x of size N (totaling $M \cdot (2N - 1)$ arithmetic operations) can be evaluated using the formula

$$T_1^{Ax} = C_1 \cdot M(2N - 1). \tag{3}$$

Hereafter, the superscript T denotes the operation being evaluated (in this case, the operation of multiplying a matrix by a vector – Ax), and the subscript denotes the number of processes used for calculation (in this sequential case – 1); C_1 is the average speed of performing one arithmetic operation (determined by the architecture of the processor and the computing node on which the computing process is performed, and is considered known or can be calculated). At the same time, it is further assumed that the number of processor cycles required for adding numbers and multiplying them is equal.

If n computing nodes are used for parallel computing within a processor grid of size $n_1 \times n_2$, the time spent by each computing node on computation can be estimated as

$$C_1(n_1, n_2) \cdot \frac{M}{n_1}\Big(2\frac{N}{n_2} - 1\Big) \equiv C_1(n_1, n_2) \cdot \frac{M(2N - n_2)}{n}.$$

Note. Hereafter, it is assumed that the difference in the sizes of $M_{part(k)}$ and $N_{part(l)}$ data blocks on various processes can be ignored and considered equal to $M_{part(k)} \equiv M/n_1$ and $N_{part(l)} \equiv N/n_2$ respectively. This is due to the fact that usually the maximum difference in the size of data blocks distributed across different processes is no more than 1, which means that in the case of solving "large" problems, this difference can be ignored.

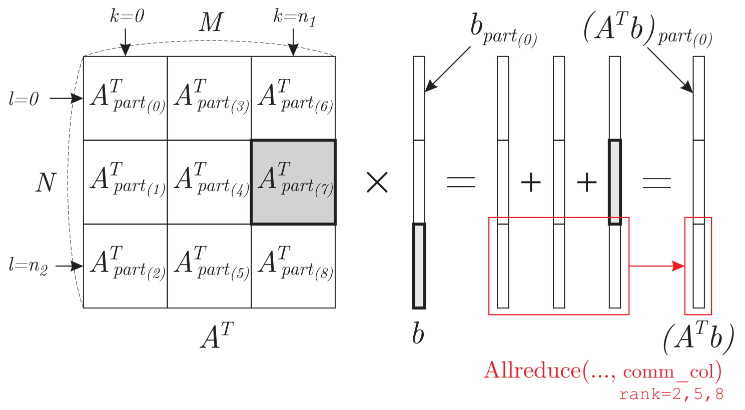
At the same time, it is specifically noted that the coefficient C_1 depends on the sizes n_1 and n_2 of the process grid, since these sizes determine the sizes of the blocks $(Ax)_{part(\cdot)}$ of matrix A that are involved in calculations. Given that in popular programming languages used for parallel programming (for example, C/C++ and Python), matrices (i.e., two-dimensional arrays) are stored line-by-line in memory, the sizes of these matrices will determine the access time to their elements during the software implementation of matrix-vector multiplication.

The overhead time for organizing the interaction of each group of n_2 processes included in each of the rows of the process grid can be estimated as follows. First, the amount of data transmitted by each process is proportional to $\dfrac{M}{n_1}$ (the proportionality coefficient is determined by the type of data transmitted). Secondly, the number of data transfer operations with optimal organization of transfers is proportional to $\log_2 n_2$ (the proportionality coefficient is determined by the implementation of the corresponding function of interaction of computational processes within the framework of the parallel programming technology used — for example, the function of collective interaction of processes `Allreduce()` from MPI). Thus, the overhead time will be proportional to $\dfrac{M}{n_1} \cdot \log_2 n_2$, where the proportionality coefficient will take into account both the already mentioned proportionality coefficients and the technical features of the communication network, as well as the implemented option for distributing processes across computing nodes.

Thus, the total running time of the corresponding parallel algorithm can be estimated using the formula

$$T_n^{Ax} = C_1(n_1, n_2) \cdot \frac{M(2N - n_2)}{n} + C_2(n_1, n_2) \cdot \frac{M}{n_1} \cdot \log_2 n_2. \tag{4}$$

Here, the coefficient C_2 is considered known and 1) depends on the speed of transmission of a unit of information over a communication network, which, in turn, depends on the relative location of computing nodes, in particular, as determined by the sizes n_1 and n_2 of the process grid, as well as on the implemented option for distributing processes across computing nodes; 2) depends on the implementation of the function of collective interaction of processes.



$k=0$
M
$k=n_1$
$l=0$
N
$l=n_2$
$A^T_{part(0)}$ $A^T_{part(3)}$ $A^T_{part(6)}$
$A^T_{part(1)}$ $A^T_{part(4)}$ $A^T_{part(7)}$
$A^T_{part(2)}$ $A^T_{part(5)}$ $A^T_{part(8)}$
A^T
$b_{part(0)}$
b
$(A^Tb)_{part(0)}$
(A^Tb)
Allreduce(..., comm_col)
rank=2,5,8

Figure 3. A parallel algorithm for multiplying a transposed matrix by a vector in the case of a two-dimensional matrix partition into blocks. The figure shows the case of data distribution across nine computing processes that form a two-dimensional 3×3 grid. Taking into account the transposition, it is assumed that the rows of the presented process grid are the columns of the original process grid; similarly, the columns of the presented process grid are the rows of the original process grid

The estimates (3) and (4) will be used in subsection 1.6 to construct a formula for evaluating the effectiveness of the parallelization of the software implementation of the algorithm for solving the problem (1).

1.3. Estimation of the Implementation Time of a Parallel Algorithm for Multiplying a Transposed Matrix by a Vector in the Case of a Two-Dimensional Matrix Partition Into Blocks

The result of multiplying the transposed matrix A^T of size $N \times M$ by the vector b of size M will be a vector (A^Tb), which will be distributed over various computational processes in parts $(A^Tx)_{part(\cdot)}$. At the same time, the distribution structure of this vector for various processes should correspond to the distribution structure of the vector x (see Fig. 1). Part of the vector (A^Tb) can be calculated using the following formula:

$$(A^Tb)_{part(l)} = \sum_{k=0}^{n_1-1} A^T_{part(k \cdot n_2 + l)} b_{part(k)}, \quad l = \overline{0, \dots, n_2 - 1}. \tag{5}$$

An example explaining this formula is provided in Fig. 3.

Each process involved in the calculations can compute the term $A^T_{part(k \cdot n_2 + l)} b_{part(k)}$ for its pair of values (k, l), independently of the similar calculations performed by other processes. That is, the terms in the formula (5) can be calculated in parallel. Then the corresponding terms located on the processes of the column of the process grid with index l should be summed up, and the result – vector $(A^Tb)_{part(l)}$ – should be placed on all processes of this column of the process grid. The organization of the interaction of processes entails overhead costs for receiving and transmitting messages containing the results of intermediate calculations through the physical communication environment of a multiprocessor computing system.

Similarly to the method described in the previous subsection, it is possible to obtain an estimate of the time required for the sequential implementation of this operation.

$$T_1^{A^Tx} = \tilde{C}_1 \cdot N(2M - 1)$$

and parallel:

$$T_n^{A^T x} = \tilde{C}_1(n_1, n_2) \cdot \frac{N \cdot (2M - n_1)}{n} + C_2(n_1, n_2) \cdot \frac{N}{n_2} \cdot \log_2 n_1.$$

The fundamental difference between these formulas and the formulas presented in the previous subsection is that the coefficient C_1 has been replaced by $\tilde{C}_1$. This is due to the fact that the implementation of the multiplication of a transposed matrix by a vector involves sequential access to the elements of the rows of this matrix; however, the transposed matrix is stored in memory by columns. In order to save memory when solving "large" problems, the preliminary transposition of the matrix is not constructive, as it requires additional memory space that may not be available. A different order of access to the elements of the matrix leads to the fact that the coefficient $\tilde{C}_1$ may differ significantly from the coefficient C_1.

1.4. Estimation of the Implementation Time of the Parallel Scalar Product Algorithm of Vectors

Since the size of the vectors in sequential algorithms for solving the problem (1) is either M or N, we will use characteristic vectors of these sizes. Therefore, if the vector x is mentioned, its size will be assumed to be N, and if the vector b is mentioned, its size will be assumed to be M. It is assumed that these vectors are distributed over computational processes according to the scheme shown in Fig. 1.

The result of the scalar product of vectors $x^{(1)}$ and $x^{(2)}$ of size N will be the value $(x^{(1)}, x^{(2)})$. It can be calculated using the following formula:

$$(x^{(1)}, x^{(2)}) = \sum_{l=0}^{n_2-1} \big(x^{(1)}_{part(l)}, x^{(2)}_{part(l)}\big). \tag{6}$$

Each process involved in the calculations can compute the term $\big(x^{(1)}_{part(l)}, x^{(2)}_{part(l)}\big)$ for its value l, which is independent of similar calculations performed by other processes. That is, the terms in the formula (6) can be calculated in parallel. Then the corresponding terms located on the processes of the row of the process grid should be summed up, and the result – the number $(x^{(1)}, x^{(2)})$ – must be placed on all processes in this row of the process grid. The organization of the interaction of processes entails overhead costs for receiving/transmitting messages containing the results of intermediate calculations through the physical communication environment of a multiprocessor computing system.

The running time of a sequential scalar product operation of vectors can be estimated using the formula

$$T_1^{(x^{(1)}, x^{(2)})} = C_1 \cdot (2N - 1),$$

and the running time of the corresponding parallel algorithm (taking into account overhead costs) can be estimated using the formula

$$T_n^{(x^{(1)}, x^{(2)})} = C_1(n_1, n_2) \cdot \frac{2N - n_2}{n_2} + C_2(n_1, n_2) \cdot \log_2 n_2.$$

It is necessary to note the following two features of the algorithm. First, the processes of each row of the process grid perform the same calculations as the processes of the other rows of the process grid. Thus, scalar product calculations are not parallelized over all n computational processes, but only over n_2 processes. This approach is constructive, since distributing the vectors $x^{(1)}$ and $x^{(2)}$ across all n processes (and not just the n_2 processes of the row of the process grid), and then collecting data from all of them, will require additional overhead proportional to $\log_2 n$, which almost always exceeds the gain in reducing the time of direct calculations when solving "large" problems ($C_1(2N-1)/n$ instead of $C_1(2N-1)/n_2$ in the considered version of the parallel algorithm). Secondly, it assumes that the parts of the vector are well localized in the address space of each process. In this regard, the same estimate can be used for the average execution speed of one arithmetic operation C_1 as in subsection 1.2.

Similarly, the result of the scalar product of vectors $b^{(1)}$ and $b^{(2)}$ of size M will be the value $(b^{(1)}, b^{(2)})$. It can be calculated using the following formula:

$$(b^{(1)}, b^{(2)}) = \sum_{k=0}^{n_1-1} \big(b^{(1)}_{part(k)}, b^{(2)}_{part(k)}\big). \tag{7}$$

Each process involved in the calculations can calculate the term $\big(b^{(1)}_{part(k)}, b^{(2)}_{part(k)}\big)$ for its value k is independent of similar calculations performed by other processes. That is, the terms in the formula (7) can be calculated in parallel. Then the corresponding terms located on the processes of the column of the process grid should be summed up, and the result – the number $(b^{(1)}, b^{(2)})$ – must be placed on all processes in this column of the process grid. The organization of the interaction of processes entails overhead costs for receiving/transmitting messages containing the results of intermediate calculations through the physical communication environment of a multiprocessor computing system.

The running time of a sequential scalar product operation of vectors can be estimated using the formula

$$T_1^{(b^{(1)}, b^{(2)})} = C_1 \cdot (2M-1),$$

and the running time of the implementation of the corresponding parallel algorithm (taking into account overhead costs) can be estimated using the formula

$$T_n^{(b^{(1)}, b^{(2)})} = C_1(n_1, n_2) \cdot \frac{2M - n_1}{n_1} + C_2(n_1, n_2) \cdot \log_2 n_1.$$

1.5. Estimation of the Implementation Time of a Parallel Algorithm for Adding/subtracting Vectors or Multiplying/dividing a Vector by a Number

When distributing vectors across computational processes according to the scheme shown in Fig. 1, in the case of adding two vectors $x^{(1)}$ and $x^{(2)}$ of size N, the following calculations will be performed on each process:

$$(x^{(1)} + x^{(2)})_{part(l)} = x^{(1)}_{part(l)} + x^{(2)}_{part(l)}.$$

There will be no need to exchange messages between computational processes, since the storage structure of the resulting vector corresponds to that used in the parallel algorithms of linear algebra operations described earlier.

Therefore, the time of successive operations of adding vectors of the appropriate size can be estimated using the formulas

$$T_1^{x^{(1)}+x^{(2)}} = C_1 \cdot N,$$
$$T_1^{(b^{(1)}+b^{(2)})} = C_1 \cdot M,$$

and the operating time of the implementation of the corresponding parallel algorithms can be estimated using the formulas

$$T_n^{x^{(1)}+x^{(2)}} = C_1 \cdot \frac{N}{n_2},$$
$$T_n^{(b^{(1)}+b^{(2)})} = C_1 \cdot \frac{M}{n_1}.$$

Exactly the same estimates are true for operations of multiplying the corresponding vectors by a number. Therefore, the formulas for estimating the values of $T^{c\cdot x}$ and $T^{c\cdot b}$ are not written out separately and are considered equivalent to those described above.

At the same time, similarly to the previous subsection (subsection 1.4), it should be noted that the processes of each row (column) of the process grid perform the same calculations as the processes of the other rows (columns) of the process grid. Thus, the corresponding calculations are not parallelized over all n computing processes, but only over n_2 (n_1) processes. The motivation for this approach is similar to that given in subsection 1.4.

1.6. Evaluation of the Parallelization Efficiency of a Parallel Algorithm

The parallelization efficiency of the iterative algorithm for solving the problem (1) can be estimated by the parallelization efficiency of one iteration using the following formula:

$$E_n\big(n_1, n_2, \vec{k}, C_1, \tilde{C}_1, C_2, M, N\big) = \frac{\sum\limits_{op\in\left\{Ax;\ A^Tb;\ (x^{(1)},x^{(2)});\ (b^{(1)},b^{(2)});\ x^{(1)}+x^{(2)};\ b^{(1)}+b^{(2)}\right\}} k^{op}T_1^{op}}{\sum\limits_{op\in\left\{Ax;\ A^Tb;\ (x^{(1)},x^{(2)});\ (b^{(1)},b^{(2)});\ x^{(1)}+x^{(2)};\ b^{(1)}+b^{(2)}\right\}} k^{op}T_n^{op}n}. \quad (8)$$

Here:

- n_1, n_2 are determined by the selected Cartesian topology of parallel processes of the two-dimensional grid type;
- M, N are determined by the size of the problem being solved;
- $\vec{k} = \left\{k^{Ax};\ k^{A^Tb};\ k^{(x^{(1)},x^{(2)})};\ k^{(b^{(1)},b^{(2)})};\ k^{x^{(1)}+x^{(2)}};\ k^{b^{(1)}+b^{(2)}}\right\}$ is determined by the iterative algorithm for solving the problem (1) and contains the values of the number of corresponding operations (while calculating the values of $k^{x^{(1)}+x^{(2)}}$ and $k^{b^{(1)}+b^{(2)}}$, the number of multiplications/divisions of vectors of the appropriate size by a number is also taken into account);
- C_1, $\tilde{C}_1$ are determined by the architecture of the processor and computing node;
- C_2 is determined by the communication network, the implemented option for distributing processes across computing nodes, and the selected implementation of the operation for collective interaction of processes of the `Allreduce()` type (at the same time, as mentioned earlier, this parameter also depends on the size (M and N) of the problem and the selected size (n_1 and n_2) of virtual process topologies – they determine the volume and number of messages transmitted over the communication network);
- T_1^{op} and T_n^{op} are defined by formulas written out in subsections 1.2–1.5.

Example 1. For the algorithm 1, the expression (8) for the main iterations ($s \geqslant 2$) is written as

$$E_n = \frac{2 \cdot T_1^{Ax} + 1 \cdot T_1^{A^Tb} + 2 \cdot T_1^{(x^{(1)}, x^{(2)})} + 1 \cdot T_1^{(b^{(1)}, b^{(2)})} + 6 \cdot T_1^{x^{(1)}+x^{(2)}} + 1 \cdot T_1^{b^{(1)}+b^{(2)}}}{\big(2 \cdot T_n^{Ax} + 1 \cdot T_n^{A^Tb} + 2 \cdot T_n^{(x^{(1)}, x^{(2)})} + 1 \cdot T_n^{(b^{(1)}, b^{(2)})} + 6 \cdot T_n^{x^{(1)}+x^{(2)}} + 1 \cdot T_n^{b^{(1)}+b^{(2)}}\big)n},$$

Example 2. For the algorithm 2, the expression (8) for the main iterations ($s \geqslant 2$) is written as

$$E_n = \frac{2 \cdot T_1^{Ax} + 1 \cdot T_1^{A^Tb} + 3 \cdot T_1^{x^{(1)}+x^{(2)}} + 2 \cdot T_1^{b^{(1)}+b^{(2)}}}{\big(2 \cdot T_n^{Ax} + 1 \cdot T_n^{A^Tb} + 3 \cdot T_n^{x^{(1)}+x^{(2)}} + 2 \cdot T_n^{b^{(1)}+b^{(2)}}\big)n}.$$

2. Choosing the Optimal Size of the Virtual Process Topology

The problem of choosing the optimal sizes n_1 and n_2 of the virtual process topology can be set as follows: it is necessary to determine the values of n_1 and n_2 at which the parallelization efficiency of the selected iterative algorithm for solving the problem (1) on the allocated resources of the supercomputer system will be maximum. Such a statement of the problem can be formalized in the following form:

$$(n_1, n_2) = \underset{\substack{n_1, n_2 \\ n_1 \cdot n_2 = n}}{\operatorname{argmax}} E_n(n_1, n_2; \vec{k}, C_1, \tilde{C}_1, C_2, M, N). \tag{9}$$

Here, the first two arguments of the function E_n are specially separated from the remaining arguments by a semicolon to emphasize that these arguments are used for maximization, while the remaining arguments are known parameters.

Therefore, for the practical application of this formula, two questions remain to be solved:

1. How can someone take into account the condition $n_1 \cdot n_2 = n$? Obviously, on the one hand, the formal mathematical solution to the problem (9) in many cases will be non-integers, and on the other hand, the prime number n cannot be decomposed into factors, each of which is different from 1.
2. How does someone define the coefficients C_1, $\tilde{C}_1$, C_2? Taking into account the comments made earlier regarding these constants, it is obvious that the formal evaluation of them can be extremely difficult. For example, we repeat that the coefficient C_2 depends 1) on the technical features of the communication network, 2) on the sizes n_1 and n_2 of the selected virtual process topology, 3) on how well this virtual topology was mapped to the computing resources allocated at the start of the program, 4) on the implementation option of the function of collective interaction of processes in the package used for parallel programming.

The first issue can be solved as follows. It is necessary to impose a convenient limit on the number of computing processes n. Such a natural limitation for many multiprocessor systems is $n = 2^k$, where k is an integer. Thus, in the algorithm formulated below for determining the optimal values of n_1 and n_2, we will assume that n is a power of two, and the values of n_1 and n_2 must be found as integers.

The second issue can be solved as follows. On the allocated computing resources, before starting the main calculations, it is possible to run a preliminary test using matrices and vectors of characteristic sizes M, N, which will determine C_1, $\tilde{C}_1$ and C_2. That is, the corresponding coefficients will be determined automatically for the features of the allocated resources and the selected compiler options that determine the algorithms for implementing the function of collective interaction of processes. Moreover, the corresponding test runs should be made for different values of n_1 and n_2 in order to determine the dependencies of $C_1(n_1, n_2)$, $\tilde{C}_1(n_1, n_2)$ and $C_2(n_1, n_2)$.

Thus, an algorithm is proposed for determining the optimal values n_1 and n_2 of the virtual process topology, designed as Algorithm 3.

3. Numerical Experiments

Computational experiments were conducted on the "Lomonosov-2" supercomputer [12] and were constructed as follows. The sizes M and N, which determine the computational complexity of the problem (1), were chosen in such a way that, on the one hand, the matrix A of the system (1) had significant size, and on the other hand, a part $A_{part_{(\cdot)}}$ of this matrix, which each individual computing process is responsible for storing, fit into the RAM of the computing node. For example, for pairs $(M, N) \in \{(10^5, 10^5), (10^6, 10^4), (10^7, 10^3)\}$, when using the data type `float64` ("double precision") the matrix A requires 298 GB for its storage in memory (while $A_{part_{(\cdot)}}$ requires 1.2 GB when using 64 computing nodes), and for pairs of $(M, N) \in \{(5 \cdot 10^5, 5 \cdot 10^5), (5 \cdot 10^6, 5 \cdot 10^4), (5 \cdot 10^7, 5 \cdot 10^3)\}$ – 1 863 GB (1.8 Tb) ($A_{part_{(\cdot)}}$ – 29 GB when using the same number of computing nodes).

At the same time, computational experiments were conducted for a series of these pairs of (M, N) in order to demonstrate that problems of the same computational complexity can correspond to completely different values of n_1 and n_2, which determine the optimal virtual topology of processes for parallel computing.

The MPI parallel programming technology was used for the software implementation of the Algorithm 3. The program code implementing the pseudocode from Algorithm 3 is not provided in this article, since the pseudocode is designed in such a way that the corresponding software implementation can be recovered from it in an unambiguous way (in the sense of choosing software solutions that may affect the parallelization efficiency). It should be noted that for the software implementation of the function of collective interaction of processes `Allreduce()`, its blocking version was used.

The computation results are shown in Fig. 4. It is perfectly clear that the values of n_1 and n_2, which determine the optimal virtual topology for parallel computing, depend both on the computational complexity of the problem and on the relative sizes of M and N. In particular, the intuitive facts are experimentally confirmed that, for example, for "elongated matrices" the optimal "elongated process topology" is obtained, and the closer the matrix is to the "square" one, the more optimal it is to use the "square process topology" for calculations.

However, there are some non-obvious results. For example, there may be situations where, if the virtual topology is chosen incorrectly, the parallelization efficiency may be close to zero. Although such a result is obtained for extreme cases (for example, $M = 5 \cdot 10^7$, $N = 5 \cdot 10^3$, i.e., the number of equations exceeds the number of unknowns by four orders of magnitude), which are extremely rare in solving applied problems, it is necessary to keep in mind the possibility of such effects in practice. It is also necessary to note the effect of a sharp increase in parallelization efficiency for a limiting process grid of size $n_1 \times n_2 \equiv 1 \times 64$ (noted in Fig. 4 by dotted line).

Remark 1. If someone uses a variant of the function `Allreduce()` implemented using persistent interaction requests from the MPI-4 standard, the results will obviously change. This is due to the following fact. Functionally (in the sense of the result), the operation of the function `Allreduce()` is equivalent to the sequence of launching functions `Allreduce_init()` "+" `start()` "+" `wait()` from the standard MPI-4. Given that the function `Allreduce()` can be im-

Algorithm 3: The pseudocode of the algorithm for determining the optimal size of the virtual topology of processes using the MPI parallel programming technology.

```
Data: $N$, $M$, $\vec{k}$, $n$ – the power of two
Result: $n_1$, $n_2$
comm ← MPI.COMM_WORLD
$n_1 \leftarrow n$; $n_2 \leftarrow 1$
while $n_1 >= 1$ and $n_1 <= n$ do
    comm_cart ← comm.Create_cart(dims=($n_1$, $n_2$), periods=True, reorder=True)
    rank_cart ← comm_cart.Get_rank()
    comm_col ← comm_cart.Split(rank_cart % $n_2$, rank_cart)
    comm_row ← comm_cart.Split(rank_cart // $n_2$, rank_cart)
    // formation of arbitrary matrices of characteristic dimensions
    $A_{part} \leftarrow \text{random}(M/n_1, N/n_2)$; $x_{part} \leftarrow \text{random}(N/n_2)$; $b_{part} \leftarrow \text{random}(M/n_1)$
    // estimation of the value of the coefficient $C_1$
    time_start ← MPI.Wtime()
    $(Ax)_{part} \leftarrow A_{part} \cdot x_{part}$
    time_elapsed ← MPI.Wtime() − time_start
    $C_1 \leftarrow \dfrac{\texttt{time\_elapsed} \cdot n}{M \cdot (2N - n_2)}$
    comm_cart.Allreduce($C_1/n \to C_1$, op=MPI.SUM) // averaging $C_1$
    // estimation of the value of the coefficient $\tilde{C}_1$
    time_start ← MPI.Wtime()
    $(A^T b)_{part} \leftarrow A^T_{part} \cdot b_{part}$
    time_elapsed ← MPI.Wtime() − time_start
    $\tilde{C}_1 \leftarrow \dfrac{\texttt{time\_elapsed} \cdot n}{N \cdot (2M - n_1)}$
    comm_cart.Allreduce($\tilde{C}_1/n \to \tilde{C}_1$, op=MPI.SUM) // averaging $\tilde{C}_1$
    // estimation of the value of the coefficient $C_2$
    comm_cart.Barrier()
    time_start ← MPI.Wtime()
    comm_col.Allreduce($(A^T b)_{part} \to x_{part}$, op=MPI.SUM)
    time_elapsed ← MPI.Wtime() − time_start
    if $n_1 = 1$ then $C_{2temp_1} \leftarrow 0$ else $C_{2temp_1} \leftarrow \dfrac{\texttt{time\_elapsed}}{N/n_2 \cdot \log_2 n_1}$
    comm_cart.Barrier()
    time_start ← MPI.Wtime()
    comm_row.Allreduce($(Ax)_{part} \to b_{part}$, op=MPI.SUM)
    time_elapsed ← MPI.Wtime() − time_start
    if $n_2 = 1$ then $C_{2temp_2} \leftarrow 0$ else $C_{2temp_2} \leftarrow \dfrac{\texttt{time\_elapsed}}{M/n_1 \cdot \log_2 n_2}$
    if $n_1 = 1$ then $C_{2temp_2} \leftarrow 0$; if $n_2 = 1$ then $C_{2temp_1} \leftarrow 0$
    if $n_1 \neq 1$ and $n_2 \neq 1$ then $C_2 \leftarrow (C_{2temp_1} + C_{2temp_2})/2$
    comm_cart.Allreduce($C_2/n \to C_2$, op=MPI.SUM)
    // parallelization efficiency estimation
    $E(n_1, n_2) \leftarrow E_n(n_1, n_2; \vec{k}, C_1, \tilde{C}_1, C_2, M, N)$
    $n_1 \leftarrow n_1/2$; $n_2 \leftarrow n_2 \cdot 2$
end
$(n_1, n_2) \leftarrow \text{argmax}\, E(n_1, n_2)$
```

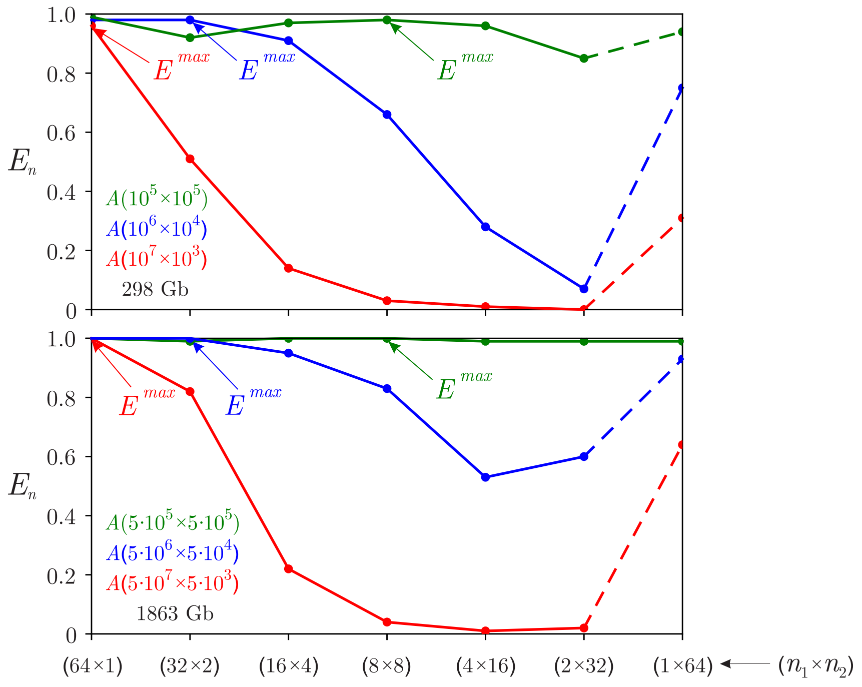


Figure 4. Graphs of the dependence of the parallelization efficiency of Algorithm 1 for matrices of different dimensions, but with the same number of elements, depending on the selected virtual topology when computing on a fixed number of nodes of the "Lomonosov-2" supercomputer [12]

plemented over the same areas of the address space, the same `Allreduce_init()` operation can be performed outside the `while` loop. This can significantly reduce the overhead of receiving and transmitting messages containing the results of intermediate calculations (for more information, see [6]). As a result, the estimate of C_2 may decrease significantly.

Remark 2. It is obvious that the method proposed in the paper for calculating the parallelization efficiency in determining the optimal virtual topology is an estimated one. Therefore, the question arises: how much does the actual parallelization efficiency differ from the estimated one in typical tasks? To answer this question, the following experiment was conducted. An example was chosen for $(M, N) = (8 \cdot 10^4, 8 \cdot 10^4)$. In this case, the matrix A will require 47.8 GB for its storage, as a result of which it will completely fit into the RAM of one computing node (64 GB) of the "Lomonosov-2" supercomputer, which is required to determine the operating time T_1 of the sequential version of the Algorithm 1. For this matrix A and some model right-hand side b_δ of the system (1), a parallel implementation of the Algorithm 1 was launched in two versions – using the functions of collective communication of MPI-processes within the MPI-3 standard and with using the functions of collective communication of MPI-processes within the MPI-4 standard (see previous remark). At the same time, the number of iterations in the Algorithm 1 was forcibly limited to 300 iterations in order to obtain a reasonable counting time for the sequential implementation of the Algorithm $T_1 = 864$ seconds, which is a fairly representative counting time, but at the same time slightly less than the upper limit of 15 minutes, which limits the counting time on the test queue of the "Lomonosov-2" supercomputer. For this example, as for similar examples with a square matrix of higher dimensions, the optimal topology is a "square" one. Therefore, parallel implementations of the algorithm were launched on $n \in \{4, 9, 16, 25, 36, 49, 64\}$ MPI-processes so that $n_1 \equiv n_2 \equiv \sqrt{n}$. The running time T_n for each run was detected, then the speedup of calculations was calculated using the formula $S_n = T_1/T_n$

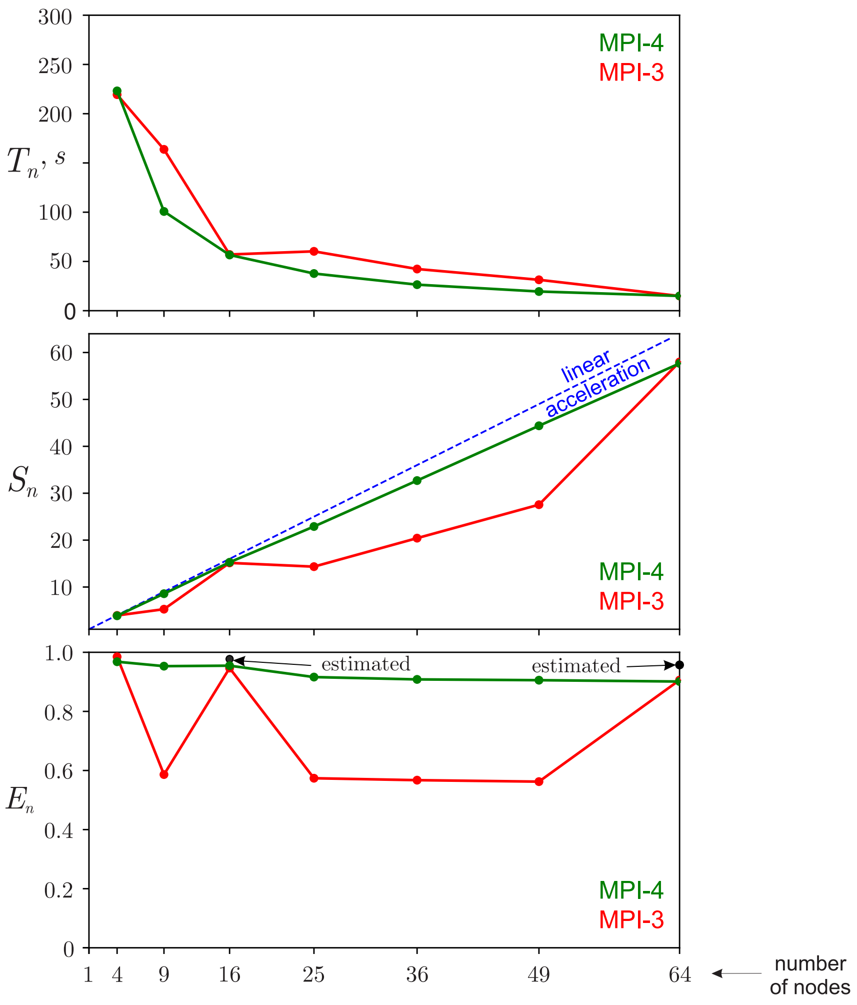


Figure 5. Graphs of the dependence of the counting time, speedup, and parallelization efficiency of the Algorithm 1 depending on the selected number of computing nodes within the framework of the "quadratic process topology". The graphs are typical for a typical launch, rather than the average values for a series of experiments. On the efficiency graph, the points "estimated" are marked separately, the values of which are calculated using the Algorithm 3

and the parallelization efficiency using the formula $E_n = S_n/n$. Based on these data, work schedules, speedup, and efficiency were constructed, as shown in Fig. 5. The key graph here is the graph of the real parallelization efficiency, which demonstrates certain differences between the real parallelization efficiency and the estimated one. However, these differences look insignificant from a practical point of view, which confirms the applicability of the proposed Algorithm 3.

4. Discussion

1. **Large-scale autotuning.** The ideas for adjusting the algorithm parameters to the characteristics of the target computing system, used in the development of the proposed methodology, are consonant with the ideas of autotuning (see, for example, the works [8, 9, 13–15]). However, in these works, the ideas considered mainly relate to "small-scale autotuning" – features of adjusting algorithms to the technical capabilities of "small" computing systems (a multicore processor or a graphics processing unit). The ideas proposed in our work relate

to adjusting the algorithm parameters to the characteristics of a distributed memory supercomputer and have not been previously considered in detail in publicly available sources.

2. **The invariance of the proposed algorithm with respect to the properties of the computing system used.** Such important characteristics of a communication network as latency, data transfer rate, topology of the communication network, and distribution of processes across computing nodes are important in the *a priori* estimation of the coefficient C_2. Also, many technical features of the computing node must be taken into account when *a priori* estimating the coefficients C_1 and $\tilde{C}_1$. In fact, the paper proposes *a posteriori* method for determining these constants based on the results of some preliminary tests, which automatically sets in them the corresponding characteristics of the used communication network and computing nodes. Thus, the proposed algorithm for determining the optimal values of the virtual topology of processes is relevant for any hardware, any parallel programming technology, any compilation options, and any features of the system and application software.
3. **Possible improvements to the proposed algorithm.** In the derivation of all formulas, the assumption was made that the computational complexity of all arithmetic operations is equivalent. If desired, the reader can clarify the output of the corresponding formulas, taking into account that, for example, addition of two numbers requires 1–3 CPU cycles, multiplication – 1–7 cycles, and division – 10–40 cycles. The values of the coefficients C_1 and $\tilde{C}_1$ on a processor-homogeneous computing system should not change much from launch to launch; therefore, they could potentially be calculated once before the start of all experiments, saved and used in the future. However, the distribution of processes across the computing nodes of a supercomputer can vary from launch to launch, even on the same set of computing nodes. Therefore, the coefficient C_2 must be calculated before each run of the computational algorithm implementation. The values of the coefficients C_1 and $\tilde{C}_1$ are averaged over all application processes. Considering that it is supposed to use a computer system that is homogeneous in terms of processors, this should not be a serious limitation and simplification. The coefficient C_2 is calculated as the average value for `Allreduce()` operations performed across rows and columns of the process grid. For greater accuracy of the algorithm, one can do without averaging and use two separate coefficients. Next, the value of the coefficient C_2 is averaged over all application processes, and the impact of this averaging may be significant. Further experiments should demonstrate how significant this factor is in practice and needs to be taken into account in order to obtain more accurate results.
4. **Theoretical assumptions.** Using only powers of two for the values of n_1 and n_2 does not represent a serious limitation of generality. This assumption is often used in practice when evaluating various dynamic characteristics of software implementations.
5. **Applicability of the algorithm.** The algorithm described in this article is designed to solve linear algebra problems using a two-dimensional Cartesian topology of processes, but it can be easily adapted to solve other problems using other topologies.
6. **Features of software implementation of algorithms.** Some statements made in subsections 1.1–1.5 when estimating the implementation time of parallel algorithms using basic linear algebra operations may be incomprehensible to readers who have not previously implemented parallel versions of iterative algorithms for solving systems of linear algebraic equations. For example, the statement may not be obvious (see subsection 1.5), that with

the chosen data storage structure, when adding vectors, there will be no need to exchange messages between different computing processes. Therefore, we provide a link to the work [6], which contains an example of a software implementation of an iterative algorithm from the class of algorithms under consideration (including using the functions of collective interaction of processes from various MPI standards: MPI-3 and MPI-4).

7. **The case of a heterogeneous computing system.** The approach discussed in this article is applicable without additional modifications for computing systems that are homogeneous in terms of core computing – whether they are central processing units (CPUs) or graphics accelerators (GPUs), while the system may be heterogeneous over a communication network. If a supercomputer co-design is required for a more complex heterogeneous system, then a more complex approach beyond the scope of this article may be required to distribute basic computing operations across heterogeneous computers.

Conclusion

The paper demonstrates the fundamental possibility of using the ideas of supercomputing co-design to automatically match the optimal topology of calculations with the features of the problem being solved, the features of the supercomputer system used and parallel programming technology. The proposed methodology for algorithms that are in demand in solving applied problems can increase the efficiency of using supercomputer systems by users.

Acknowledgements

The paper was published with the financial support of the Russian Science Foundation (project 25-11-00181, https://rscf.ru/en/project/25-11-00181/). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [12].

References

1. Antonov, A.S.: MPI and OpenMP Parallel Programming Technologies: Textbook. Moscow University Press (2012)

2. Antonov, A.S., Maier, R.V., Nikitenko, D.A., Voevodin, V.V.: An approach to solving the problem of supercomputer co-design. Lobachevskii J Math. 45(7), 2965–2973 (2024). https://doi.org/10.1134/S1995080224603680

3. Kalitkin, N.N., Kuzmina, L.V.: Improved form of the conjugate gradient method. Mathematical Models and Computer Simulations 4(1), 68–81 (2012). https://doi.org/10.1134/S2070048212010061

4. Kalitkin, N.N., Kuzmina, L.V.: Improved forms of iterative methods for systems of linear algebraic equations. Doklady Mathematics 88(1), 489–494 (2013). https://doi.org/10.1134/S1064562413040133

5. Kindermann, S.: Optimal-order convergence of Nesterov acceleration for linear ill-posed problems*. Inverse Problems 37(6), 065002 (2021). https://doi.org/10.1088/1361-6420/abf5bc

6. Lukyanenko, D.: Parallel algorithm for solving overdetermined systems of linear equations, taking into account round-off errors. Algorithms 16(5), 242 (2023). https://doi.org/10.3390/a16050242

7. Neubauer, A.: On Nesterov acceleration for Landweber iteration of linear ill-posed problems. Journal of Inverse and Ill-posed Problems 25(3), 381–390 (2017). https://doi.org/10.1515/jiip-2016-0060

8. Park, J., Shin, Y., Lee, J., *et al.*: HYPERF: End-to-End Autotuning Framework for High-Performance Computing. Proceedings of the 34th International Symposium on High-Performance Parallel and Distributed Computing (20), 1–14 (2025). https://doi.org/10.1145/3731545.3731588

9. Petrovič, F., Střelák, D., Hozzová, J., *et al.*: A benchmark set of highly-efficient CUDA and OpenCL kernels and its dynamic autotuning with Kernel Tuning Toolkit. Future Generation Computer Systems 108, 161–177 (2020). https://doi.org/10.1016/j.future.2020.02.069

10. Tikhonov, A.N., Goncharsky, A.V., Stepanov, V.V., Yagola, A.G.: Numerical methods for the solution of ill-posed problems. Dordrecht: Kluwer Academic Publishers (1995)

11. Voevodin, V., Antonov, A., Dongarra, J.: AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. Supercomputing Frontiers and Innovations 2(1), 4–18 (2015). https://doi.org/10.14529/jsfi150101

12. Voevodin, V., Antonov, A., Nikitenko, D., *et al.*: Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). https://doi.org/10.14529/jsfi190201

13. Vuduc, R., Demmel, J.W., Yelick, K.A.: OSKI: A library of automatically tuned sparse matrix kernels. Journal of Physics: Conference Series 16(1), 521 (2015). https://doi.org/10.1088/1742-6596/16/1/071

14. van Werkhoven B.: Kernel Tuner: A search-optimizing GPU code auto-tuner. Future Generation Computer Systems 90, 347–358 (2019). https://doi.org/10.1016/j.future.2018.08.004

15. Wu, X., Balaprakash, P., Kruse, M., *et al.*: ytopt: Autotuning scientific applications for energy efficiency at large scales. Concurrency and Computation: Practice and Experience 37(1), e8322 (2023). https://doi.org/10.1002/cpe.8322

DOI: 10.14529/jsfi250403

Supercomputer Methods of Ultrasound Tomographic Imaging in NDT Based on Lamb Waves

Alexander S. Belyaev[1], ***Alexander V. Goncharsky***[1], ***Sergey Y. Romanov***[1], ***Vadim V. Voevodin***[1]

This article concerns the developing of supercomputer methods for solving inverse problems of ultrasonic tomography in application to nondestructive testing of thin plates using Lamb waves. Such problems are computationally expensive, as longitudinal, shear, and other waves propagate in solids, requiring the use of vector elastic models of wave propagation. Iterative methods for solving the inverse problem have been developed. The methods are based on gradient descent methods of minimizing the residual functional. Efficiency of the proposed algorithms is illustrated on model problems. The field of wave tomography, which is currently under development, requires powerful computing resources. Parallel computations in this study have been performed on general-purpose processors of the Lomonosov supercomputer complex. Solving the Helmholtz equation is the core element of the developed algorithms for solving inverse problems of wave tomography. The most demanding computations involve solving linear equations with large-scale sparse matrices using LU-decomposition method. The algorithms were implemented using linear algebra libraries with serial and parallel code. Effectiveness, scalability and performance of the method has been evaluated on CPU computing platforms.

Keywords: supercomputer, mathematical modeling, guided wave tomography, inverse problems, Lamb waves.

Introduction

This paper concerns the developing of developing ultrasonic tomographic imaging methods for non-destructive testing of solids. Immediately after the advent of the first medical X-ray tomographs, the idea of ultrasonic tomographic imaging devices arose. Solving this problem would eliminate the use of ionizing radiation in medical imaging. Ultrasonic tomography opens up fundamentally new possibilities for nondestructive imaging of solids. However, developing wave tomography devices has proven much more complex than X-ray ones, primarily due to the mathematical challenges involved. While the inverse problems in X-ray tomography can be solved using linear mathematical models, wave tomography imaging requires solving nonlinear inverse problems.

Over the past 20 years, breakthrough results in ultrasound tomography have been achieved in soft tissue diagnostics. One of the key challenges in modern medicine is the diagnosis of breast cancer in the early stages of the disease. This problem can be successfully solved using ultrasound tomography [1–5]. A unique feature of breast cancer diagnostics is the ability to interpret tomography data in a scalar wave model, since soft tissues are more than 90% water. Mostly the pressure waves propagate through soft tissues. However, even in this simple model, the inverse problem is nonlinear, and its solution requires the use of supercomputers [6]. Efficient iterative algorithms for interpreting experimental data have been developed using the explicit formulation of the gradient of the residual functional between the calculated and measured wave fields at the detectors [7, 8]. Effectiveness of the developed methods has been evaluated on supercomputers using various computing platforms [9, 10].

Inverse problems of ultrasound tomography in NDT are much more complex, since the mathematical model must describe the propagation of both longitudinal and shear waves. The

[1]Lomonosov Moscow State University, Moscow, Russian Federation

displacement of a point is described by a vector, and a system of second-order differential equations is constructed for the components of the displacement vectors [11–14]. The inverse problem is reduced to minimizing a functional dependent on three unknowns: density and two elasticity coefficients as functions of spatial coordinates. As in the scalar case, a representation for the gradient of the residual functional with respect to these three parameters is obtained in the vector elastic model [13, 15, 16]. Thus, from a mathematical standpoint, in both the scalar and vector models, the objective is to solve a coefficient inverse problem for differential equations. Iterative methods for finding an approximate solution employ the explicit representation for the gradient of the residual functional.

However, in this formulation, the inverse problem is computationally very expensive. For a number of practically important non-destructive testing (NDT) problems, the use of simpler mathematical models appears promising [17]. One such problem is ultrasonic diagnostics of defects in thin plates [18–21]. This study evaluates the supercomputer implementation of the ultrasonic tomographic imaging method for NDT using Lamb waves. Capabilities of the approximate solution method for the inverse problem of detecting surface defects in metal plates are examined in detail. Solving the Helmholtz equation is the core element of the proposed imaging algorithms. A specialized scalar model for thin-plate defect diagnostics allows for determining not only shapes, but also the depth map of defects. Even in this simplified model, a supercomputer is required to implement the developed algorithms in practice. The effectiveness of tomographic image reconstruction algorithms on a CPU platform is discussed in detail.

The article is organized as follows. Sections 1, 4.1, 4.2 are devoted to the formulation and model computations of the direct problem of ultrasound tomography. Sections 2, 3, 4.3, 4.4 are devoted to the formulation and model computations of the inverse problem. Section 5 contains performance analysis of a supercomputer implementation. Conclusion summarizes the study.

1. Statement of the Direct Problem of Ultrasound Tomography in a Vector Elastic Model

The object of tomographic imaging in this study is a thin, homogeneous flat plate, the thickness of which is on the order of the central wavelength of the sounding pulse (Fig. 1). Defect detection in thin plates and pipes is a pressing practical issue [18, 19]. The specificity of this problem is that Lamb waves of different modes propagate in thin plates and pipes. Defects in the imaged sample can be internal, associated with a local change in the parameters of the medium, or surface defects, associated with a local change in the plate thickness. First, we present the formulations of the direct and inverse problems for the case of internal defects.

Figure 1 shows a diagram of a tomographic imaging experiment, including sources and detectors of ultrasonic radiation. The sources sequentially emit ultrasonic sounding pulses, which are recorded by the detectors.

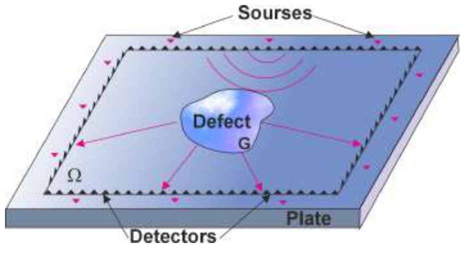


Figure 1. Scheme of the tomographic experiment

We assume that the sounding waves propagate in isotropic, linear, perfectly elastic media described by the Lame parameters and the density of the material. To simulate the wave propagation process in elastic media, the problem is often formulated in terms of particle velocity and stress. The dynamic equations in $\boldsymbol{R}^3$ describing wave propagation in an elastic volumetric plate in the velocity-stress formulation have the form of a first-order hyperbolic system [22]

$$\rho\frac{\partial u_1}{\partial t}=\frac{\partial\sigma_{11}}{\partial x_1}+\frac{\partial\sigma_{12}}{\partial x_2}+\frac{\partial\sigma_{13}}{\partial x_3},\quad \rho\frac{\partial u_2}{\partial t}=\frac{\partial\sigma_{12}}{\partial x_1}+\frac{\partial\sigma_{22}}{\partial x_2}+\frac{\partial\sigma_{23}}{\partial x_3},\quad \rho\frac{\partial u_3}{\partial t}=\frac{\partial\sigma_{13}}{\partial x_1}+\frac{\partial\sigma_{23}}{\partial x_2}+\frac{\partial\sigma_{33}}{\partial x_3},$$

$$\frac{\partial}{\partial t}\begin{pmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{23}\\ \sigma_{13}\\ \sigma_{12}\end{pmatrix}=\begin{pmatrix}\lambda+2\mu & \lambda & \lambda & 0 & 0 & 0\\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0\\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0\\ 0 & 0 & 0 & \mu & 0 & 0\\ 0 & 0 & 0 & 0 & \mu & 0\\ 0 & 0 & 0 & 0 & 0 & \mu\end{pmatrix}\times\begin{pmatrix}\frac{\partial u_1}{\partial x_1}\\ \frac{\partial u_2}{\partial x_2}\\ \frac{\partial u_3}{\partial x_3}\\ \frac{\partial u_2}{\partial x_3}+\frac{\partial u_3}{\partial x_2}\\ \frac{\partial u_3}{\partial x_1}+\frac{\partial u_1}{\partial x_3}\\ \frac{\partial u_1}{\partial x_2}+\frac{\partial u_2}{\partial x_1}\end{pmatrix}+\begin{pmatrix}f_{11}\\ f_{22}\\ f_{33}\\ f_{23}\\ f_{13}\\ f_{12}\end{pmatrix}. \tag{1}$$

Here $\boldsymbol{u}(t,\boldsymbol{r})=(u_1,u_2,u_3)$ is the shear velocity vector, $\boldsymbol{\sigma}(t,\boldsymbol{r})=\sigma_{ij}$ is the stress tensor of size 3×3; $\mu(\boldsymbol{r})$, $\lambda(\boldsymbol{r})$ are the Lame coefficients; $\rho(\boldsymbol{r})$ is the bulk density; $\boldsymbol{f}(t,\boldsymbol{r})=f_{ij}$ is the tensor defining the external force. For these equations, we should also specify the initial conditions of rest at the time $t=0$

$$\boldsymbol{u}(0,x)=0,\ \boldsymbol{\sigma}(0,x)=0. \tag{2}$$

When solving the direct problem of elastic wave propagation, the functions $\mu(\boldsymbol{r})$, $\lambda(\boldsymbol{r})$, $\rho(\boldsymbol{r})$ are known, and the components of the wave field for each point of the region Ω are to be determined. Boundary conditions are applied on the boundary $\partial\Omega$ of the plate Ω. The lateral sides of the boundary $\Gamma\subset\partial\Omega$ are assumed stationary with the following boundary conditions

$$\boldsymbol{u}(t,\boldsymbol{r})\mid_{\Gamma}=0. \tag{3}$$

The rest of the boundary (the upper and lower sides of the plate) $\partial\Omega\setminus\Gamma$ is assumed a free boundary, i.e., no external forces act on $\partial\Omega\setminus\Gamma$, and we apply the conditions

$$\sigma(t,\boldsymbol{r})\,\boldsymbol{n}\mid_{\partial\Omega\setminus\Gamma}=0, \tag{4}$$

where $\boldsymbol{n}$ is the outer normal vector to the boundary at point $\boldsymbol{r}$.

In all the computations presented, the following sounding waveform was used as the external disturbance

$$f_{33}(x_1,x_2,x_3=0,t)=\begin{cases}C\sin\left(\frac{2}{3}\pi\nu_0 t\right)\sin\left(2\pi\nu_0 t\right), & 0<t<3/w_0,\\ 0,\ \text{otherwise},\end{cases} \tag{5}$$

and the remaining components of the force tensor $f_{ij}=0$. Here w_0 is the central frequency of the sounding pulse. The constant C determines the amplitude of the external disturbance. For the simulations presented below, we assume $C=1$ GPa$/\mu s=10^{-6}$ kg/(mm $\cdot\,\mu s^3$). The time dependence of the pulse is a smooth function with a zero time derivative at the boundaries and a zero integral over the time of action. Under these conditions, the pulse spectrum does not contain very low and very high frequencies. The external disturbance is applied at a certain point on the upper surface of the plate.

2. Statement of the Inverse Problem of Ultrasonic Tomographic Imaging of Internal Defects in a Plate in Elastic Vector Models

Let us consider the inverse problem of ultrasonic tomography for the velocity-stress equations for detecting internal defects in a plate. Figure 1 shows a diagram of ultrasonic tomography of a plate. A defect with three unknown parameters of the medium $(\mu(\boldsymbol{r}), \lambda(\boldsymbol{r}), \rho(\boldsymbol{r}))$ occupies a region G. A plate Ω has a known constant thickness d_0 and parameters $(\mu_0, \lambda_0, \rho_0)$ outside the defect G. Ultrasound emitters and detectors are placed around the defect G on the surface of the plate. The emitters are located at points $\boldsymbol{q_j}$ with a total of M emitter positions $j = 1, \ldots, M$. Wave field $\boldsymbol{v}(t, \boldsymbol{r}; \boldsymbol{q_j})$ is measured at the boundary S surrounding the region G. Let $\boldsymbol{V}(t, \boldsymbol{s}; \boldsymbol{q_j})$ denote the experimental data collected for detector position $s \in S$, source position $\boldsymbol{q_j}$, $j = 1, \ldots, M$, over the time interval $[0, T]$. A known function $\boldsymbol{f}(t, \boldsymbol{r})$ describes the sounding pulse. In the inverse problem, the goal is to determine the unknown parameters of the medium $(\mu(\boldsymbol{r}), \lambda(\boldsymbol{r}), \rho(\boldsymbol{r}))$ for $\boldsymbol{r} \in G$, knowing the measured experimental data $\boldsymbol{V}(t, \boldsymbol{s}; \boldsymbol{q_j})$. The displacement function $\boldsymbol{v}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)$ is obtained by solving the main problem (1)–(4) for velocities and stresses, followed by integrating the velocity function using the formula $\boldsymbol{v}(t, \boldsymbol{r}) = \int_0^t \boldsymbol{u}(t, \boldsymbol{r}) dt$. The wave field $\boldsymbol{v}(t, \boldsymbol{s}; \boldsymbol{q_j}; \mu, \lambda, \rho)$ computed for the parameters $(\mu(\boldsymbol{r}), \lambda(\boldsymbol{r}), \rho(\boldsymbol{r}))$ must satisfy the equation $\boldsymbol{v}\left(t, \boldsymbol{s}; \boldsymbol{q_j}; \overline{\mu}, \overline{\lambda}, \overline{\rho}\right) = \boldsymbol{V}(t, \boldsymbol{s}; \boldsymbol{q_j})$, where $\boldsymbol{s} \in S$, for all source positions $\boldsymbol{q_j}$ and for the exact (unknown) values of the parameters $(\overline{\mu}(\boldsymbol{r}), \overline{\lambda}(\boldsymbol{r}), \overline{\rho}(\boldsymbol{r}))$.

Let us introduce the residual functional $\Phi(\mu, \lambda, \rho)$ of the arguments $(\mu(\boldsymbol{r}), \lambda(\boldsymbol{r}), \rho(\boldsymbol{r}))$, which is the difference between the experimental data and the computed data

$$\Phi(\mu, \lambda, \rho) = \sum\nolimits_{j=1}^{M} \frac{1}{2} \int_0^T \int_S (\boldsymbol{v}(t, \boldsymbol{s}, \boldsymbol{q_j}; \mu, \lambda, \rho) - \boldsymbol{V}(t, \boldsymbol{s}; \boldsymbol{q_j}))^2 ds dt. \tag{6}$$

For multiple wave sources, the residual functional is the sum of the values $j = 1, \ldots, M$ obtained for each source. For each fixed source j, the integrals over the time interval $[0, T]$ and over the boundary S are calculated for all the detectors in S. Mathematically, the inverse problem is posed as the problem of finding functions $(\overline{\mu}(\boldsymbol{r}), \overline{\lambda}(\boldsymbol{r}), \overline{\rho}(\boldsymbol{r}))$ that minimize the residual functional (6) $(\overline{\mu}(\boldsymbol{r}), \overline{\lambda}(\boldsymbol{r}), \overline{\rho}(\boldsymbol{r})) : \min\limits_{\mu, \lambda, \rho} \Phi(\mu, \lambda, \rho) = \Phi(\overline{\mu}, \overline{\lambda}, \overline{\rho})$. The functions $(\overline{\mu}(\boldsymbol{r}), \overline{\lambda}(\boldsymbol{r}), \overline{\rho}(\boldsymbol{r}))$ constitute an approximate solution to the inverse problem.

To minimize the residual functional, we use gradient descent methods. A rigorous mathematical formulation for the gradient of the residual functional $\Phi(\mu, \lambda, \rho)$ with respect to parameters (μ, λ, ρ) was obtained in [15] and has the following form:

$$\begin{gathered}
\Phi_\mu(\boldsymbol{r}; \mu, \lambda, \rho) = \sum\nolimits_{j=1}^{M} \int_0^T \boldsymbol{\varepsilon}(\boldsymbol{v}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)) : \boldsymbol{\varepsilon}(\boldsymbol{h}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)) dt, \\
\Phi_\lambda(\boldsymbol{r}; \mu, \lambda, \rho) = \sum\nolimits_{j=1}^{M} \int_0^T \operatorname{div}(\boldsymbol{v}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)) \cdot \operatorname{div}(\boldsymbol{h}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)) dt, \\
\Phi_\rho(\boldsymbol{r}; \mu, \lambda, \rho) = \sum\nolimits_{j=1}^{M} \int_0^T (\boldsymbol{v_t}(t, \boldsymbol{r}; \boldsymbol{q}_j; \mu, \lambda, \rho), \boldsymbol{h_t}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)) dt.
\end{gathered} \tag{7}$$

Here, $\boldsymbol{\varepsilon}(\boldsymbol{v}(t, \boldsymbol{r})) = (D\boldsymbol{v}(t, \boldsymbol{r}) + (D\boldsymbol{v}(t, \boldsymbol{r}))^T)/2$; $D\boldsymbol{v}$ is the Jacobian of $\boldsymbol{v}(t, \boldsymbol{r})$ with respect to $\boldsymbol{r}$; and similarly for $\boldsymbol{h}(t, \boldsymbol{r})$. In the first line, the operator $(A : B) = \sum_{ij} A_{ij} B_{ij}$ denotes the element-wise scalar product of two matrices, and in the third line $(\cdot, \cdot)$ is the scalar product of vectors from $\boldsymbol{R}^3$. Equation (7) uses the displacement functions $\boldsymbol{v}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)$ and $\boldsymbol{h}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)$. The displacement function $\boldsymbol{v}(t, \boldsymbol{r}; \boldsymbol{q_j}; \mu, \lambda, \rho)$ is obtained by solving the main problem (1)–(4)

for velocities and stresses, followed by integrating the velocity function using the formula

$$\boldsymbol{v}(t,\boldsymbol{r}) = \int_0^t \boldsymbol{u}(t,\boldsymbol{r})dt, \tag{8}$$

where the initial conditions of rest are used for $\boldsymbol{u}(t,\boldsymbol{r})$. Function $\boldsymbol{h}(t,\boldsymbol{r};\boldsymbol{q_j};\mu,\lambda,\rho)$ is obtained from the solution of the following adjoint velocity-stress problem (9)–(10) for the given values (μ,λ,ρ) followed by integration of the velocity function using formula (11).

$$\begin{cases} \rho(\boldsymbol{r})\frac{\partial\boldsymbol{\omega}(t,\boldsymbol{r})}{\partial t} - \operatorname{div}\boldsymbol{\sigma}(t,\boldsymbol{r}) = \boldsymbol{u}(t,\boldsymbol{s};\boldsymbol{q_j};\mu,\lambda,\rho) - \boldsymbol{U}(t,\boldsymbol{s};\boldsymbol{q_j}), \\ \frac{\partial\boldsymbol{\sigma}(t,\boldsymbol{r})}{\partial t} = 2\mu(\boldsymbol{r})\frac{\partial\boldsymbol{\varepsilon}(t,\boldsymbol{r})}{\partial t} + \lambda(\boldsymbol{r})\operatorname{div}\boldsymbol{\omega}(t,\boldsymbol{r})\ I. \end{cases} \tag{9}$$

$$\omega(t=T,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho)=0,\quad \sigma(t=T,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho)=0,\quad \omega(t,\boldsymbol{r})\mid_\Gamma=0,\quad \sigma(t,\boldsymbol{r})\boldsymbol{n}\mid_{\partial\Omega\setminus\Gamma}=0. \tag{10}$$

$$\boldsymbol{h}(t,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho) = -\int_t^T \boldsymbol{\omega}(t,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho)dt. \tag{11}$$

Here, $\boldsymbol{\varepsilon}(t,\boldsymbol{r})$=$(D\boldsymbol{\omega}(t,\boldsymbol{r})$+$(D\boldsymbol{\omega}(t,\boldsymbol{r}))^T)/2$; $D\boldsymbol{\omega}$ is the Jacobian of $\boldsymbol{\omega}(t,\boldsymbol{r})$; I is the identity matrix.

For numerical integration over time in formulas (8) in order to obtain $\boldsymbol{v}(t,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho)$, it is convenient to use the formula for integration in reverse time $\boldsymbol{v}(t,\boldsymbol{r}) = \int_0^t \boldsymbol{u}(t,\boldsymbol{r})dt = \int_0^T \boldsymbol{v}(t,\boldsymbol{r})dt - \int_t^T \boldsymbol{v}(t,\boldsymbol{r})dt$, where the value $\int_0^T \boldsymbol{v}(t,\boldsymbol{r})dt$ is computed in advance during solving the main problem (1)–(4). In order to compute the gradient using formulas (7), the functions $\boldsymbol{v}_t(t,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho)$ and $\boldsymbol{h}_t(t,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho)$ are replaced by $\boldsymbol{u}(t,\boldsymbol{s},\boldsymbol{q_j};\mu,\lambda,\rho)$ and $\boldsymbol{\omega}(t,\boldsymbol{r},\boldsymbol{q_j};\mu,\lambda,\rho)$, respectively, which are obtained explicitly from the solution of the main and adjoint problems.

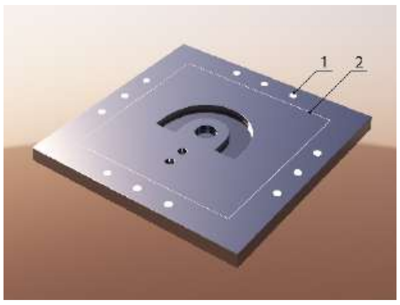
If the lateral boundaries of the plate Ω are located sufficiently far away, it is convenient to replace boundary condition (3) $\boldsymbol{u}(t,\boldsymbol{r})\mid_\Gamma= 0$ with some boundary transparency condition on Γ [23] in the computations of the main and adjoint problems. Iterative gradient-descent methods have proven highly effective in solving inverse coefficient problems. It is expected that the inverse problem in the presented formulation can be solved using high-performance computers.

3. Statement of the Inverse Problem of Ultrasonic Tomography for Detecting Surface Defects in a Plate Using a Scalar Model

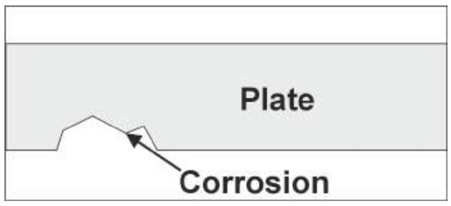
To solve the inverse problem of detecting defects inside a plate rigorously, it is necessary to formulate the inverse problem in a three-dimensional vector model, as described in Section 2. However, such a formulation is associated with a huge amount of computations, which is a demanding task even for HPC systems.

Figure 2a shows a tomographic scheme for examining surface defects on a plate. Let the defect represent a localized change in plate thickness $d(\boldsymbol{r})$; outside the defect, the plate thickness is constant and equal to d_0 (Fig. 2b). Ultrasonic radiation sources 1 are located around the defect. The wave field is recorded on line 2. The case where the defect is located on the side opposite to the sources and detectors (lower side on Fig. 2a) is of greatest interest.

Based on a number of considerations, among the various Lamb wave modes, the A_0 mode [24] is suitable for tomographic imaging. Let us consider a simplified formulation of the inverse problem for detecting surface defects. For a plate thickness that smoothly varies in the region



(a) location of sources 1 and detectors 2



(b) mathematical model of a 2D inverse problem of plate thickness reconstruction $d(\boldsymbol{r})$

Figure 2. Scheme of tomographic diagnostics of surface defects

of defects, the propagation of Lamb waves of A_0 mode is approximately described by a two-dimensional scalar wave model with varying wave velocity in the defect region. We introduce a two-dimensional coordinate system XY on the lower surface of the plate, where the sources and detectors are located. A point $\boldsymbol{r}$ on the surface thus has the coordinates $\boldsymbol{r} = (x, y) \in \boldsymbol{R}^2$. The propagation of vertical oscillations of A_0 mode Lamb waves on the lower surface of the plate is approximately described by a two-dimensional scalar model. Since the wave velocity of A_0 mode depends on the frequency, we use the Helmholtz equations for a certain set of frequencies from the spectrum of the sounding pulse

$$\Delta_r \boldsymbol{u}(w_i, \boldsymbol{r}, \boldsymbol{q_j}) + (w_i / c(\boldsymbol{r}, w_i, d(\boldsymbol{r})))^2 \boldsymbol{u}(w_i, \boldsymbol{r}, \boldsymbol{q_j}) = F\ \delta(\boldsymbol{r} - \boldsymbol{q_j}), \tag{12}$$

where $\boldsymbol{u}(w_i, \boldsymbol{r}, \boldsymbol{q_j})$ is a scalar wave at a point $\boldsymbol{r} \in \boldsymbol{R}^2$ on the lower surface of the plate; w_i is the sounding frequency $(i = 1, \ldots, N)$; $\boldsymbol{q_j} \in \boldsymbol{R}^2$ $(j = 1, \ldots, M)$ is the position of the source on the lower surface of the plate; $c(\boldsymbol{r}, w_i, d(\boldsymbol{r}))$ is the velocity of A_0 mode waves, which depends on the frequency w_i and plate thickness $d(\boldsymbol{r})$ at a point $\boldsymbol{r}$ in accordance with the dispersion relation [24]. F is the amplitude of the sounding wave. Non-reflecting boundary condition is applied at the boundary of the computational domain.

The inverse 2D problem is to determine the thickness $\overline{d}(\boldsymbol{r})$ that minimizes the residual functional $\Phi(d(\boldsymbol{r}))$, which depends on the thickness $d(\boldsymbol{r})$

$$\Phi(d(\boldsymbol{r})) = \sum\nolimits_{j=1}^{M} \sum\nolimits_{i=1}^{N} \int_S (\boldsymbol{u}(w_i, \boldsymbol{s}, \boldsymbol{q_j}, d(\boldsymbol{r})) - \boldsymbol{U}(w_i, \boldsymbol{s}, \boldsymbol{q_j}))^2 ds. \tag{13}$$

Here, $\boldsymbol{U}(w_i, \boldsymbol{s}, \boldsymbol{q_j})$ are the experimental data, which represent vertical oscillations of the A_0 mode Lamb wave at the detector locations $\boldsymbol{s}$. The gradient of the residual functional can be computed explicitly

$$\frac{\delta \Phi(d(\boldsymbol{r}))}{\delta d(\boldsymbol{r})} = \sum_{i=1}^{N} \frac{2c_0^2(w_i)}{c^3(w_i, d(\boldsymbol{r}))} \frac{\delta c(w_i, d(\boldsymbol{r}))}{\delta d(\boldsymbol{r})} \sum_{j=1}^{M} \left(\frac{w_i}{c_0(w_i)} \right)^2 \mathrm{Re}(\boldsymbol{u}(w_i, \boldsymbol{r}, \boldsymbol{q_j}, d(\boldsymbol{r})) \boldsymbol{z}(w_i, \boldsymbol{r}, \boldsymbol{q_j}, d(\boldsymbol{r}))), \tag{14}$$

where $\boldsymbol{z}(w_i, \boldsymbol{r}, \boldsymbol{q}, d(\boldsymbol{r}))$ is the solution to the adjoint problem [25]; $c(w_i, \boldsymbol{d}(\boldsymbol{r}))$ and its derivative is determined from the dispersion relation [24]. Knowing the gradient, we can use iterative gradient-descent methods for minimizing the residual functional in order to solve the inverse problem.

4. Model Computations

4.1. Numerical Method for Solving the Direct Problem of Elastic Wave Propagation in a Solid Body

For the numerical solution of the system of equations of the dynamic elasticity theory (1), a standard explicit finite-difference scheme on staggered grids is used [26–28]. The grid consists of integer and half-integer nodes. We will use the following notation

$$(x_1)_i = ih_1, \quad (x_1)_{i+1/2} = (i+1/2)h_1, \quad (x_2)_j = jh_2, \quad (x_2)_{j+1/2} = (j+1/2)h_2,$$
$$(x_3)_k = kh_3, \quad (x_3)_{k+1/2} = (k+1/2)h_3, \quad t^n = n\tau, \qquad t^{n+1/2} = (n+1/2)\tau,$$

where h_m is the grid step along the coordinate x_m, and τ is the time step. Discretized functions at integer and half-integer grid points are introduced

$$(u_1)^n_{i+1/2,j,k} = u_1((x_1)_{i+1/2},(x_2)_j,(x_3)_k,t^n), \quad (u_2)^n_{i,j+1/2,k} = u_2((x_1)_i,(x_2)_{j+1/2},(x_3)_k,t^n),$$
$$(u_3)^n_{i,j,k+1/2} = u_3((x_1)_i,(x_2)_j,(x_3)_{k+1/2},t^n), \quad (\sigma_{11})^{n+1/2}_{i,j,k} = \sigma_{11}((x_1)_i,(x_2)_j,(x_3)_k,t^{n+1/2}),$$
$$(\sigma_{22})^{n+1/2}_{i,j,k} = \sigma_{22}((x_1)_i,(x_2)_j,(x_3)_k,t^{n+1/2}), \quad (\sigma_{33})^{n+1/2}_{i,j,k} = \sigma_{33}((x_1)_i,(x_2)_j,(x_3)_k,t^{n+1/2}),$$
$$(\sigma_{23})^{n+1/2}_{i,j+1/2,k+1/2} = \sigma_{23}((x_1)_i,(x_2)_{j+1/2},(x_3)_{k+1/2},t^{n+1/2}),$$
$$(\sigma_{13})^{n+1/2}_{i+1/2,j,k+1/2} = \sigma_{13}((x_1)_{i+1/2},(x_2)_j,(x_3)_{k+1/2},t^{n+1/2}),$$
$$(\sigma_{12})^{n+1/2}_{i+1/2,j+1/2,k} = \sigma_{12}((x_1)_{i+1/2},(x_2)_{j+1/2},(x_3)_k,t^{n+1/2}).$$

To construct an explicit finite-difference scheme, the following approximations for time derivatives are used $D_t\,[f]^N_{I,J,K} = \left(f^{N+1/2}_{I,J,K} - f^{N-1/2}_{I,J,K}\right)/\tau$, where f is the differentiated function. Lowercase subscripts denote integer values, while uppercase subscripts are used for both integer and half-integer values. Spatial derivative of f with respect to x_1 is approximated as $D_1\,[f]^N_{I,J,K} = \left(f^N_{I+1/2,J,K} - f^N_{I-1/2,J,K}\right)/\tau$.

Approximations of other spatial derivatives are constructed similarly. The numerical approximation of the system of equations (1) takes the following form:

$$\begin{aligned}
&\rho_{i+1/2,j,k}D_t[u_1]^{n-1/2}_{i+1/2,j,k} = D_1[\sigma_{11}]^{n-1/2}_{i+1/2,j,k} + D_2[\sigma_{12}]^{n-1/2}_{i+1/2,j,k} + D_3[\sigma_{13}]^{n-1/2}_{i+1/2,j,k},\\
&\rho_{i,j+1/2,k}D_t[u_2]^{n-1/2}_{i+1/2,j,k} = D_1[\sigma_{12}]^{n-1/2}_{i,j+1/2,k} + D_2[\sigma_{22}]^{n-1/2}_{i,j+1/2,k} + D_3[\sigma_{23}]^{n-1/2}_{i,j+1/2,k},\\
&\rho_{i,j,k+1/2}D_t[u_3]^{n-1/2}_{i,j,k+1/2} = D_1[\sigma_{13}]^{n-1/2}_{i,j,k+1/2} + D_2[\sigma_{23}]^{n-1/2}_{i,j,k+1/2} + D_3[\sigma_{33}]^{n-1/2}_{i,j,k+1/2},\\
&D_t[\sigma_{11}]^n_{i,j,k} = (\lambda_{i,j,k} + 2\mu_{i,j,k})D_1[u_1]^n_{i,j,k} + \lambda_{i,j,k}D_2[u_2]^n_{i,j,k} + \lambda_{i,j,k}D_3[u_3]^n_{i,j,k} + [f_{11}]^n_{i,j,k},\\
&D_t[\sigma_{22}]^n_{i,j,k} = \lambda_{i,j,k}D_1[u_1]^n_{i,j,k} + (\lambda_{i,j,k} + 2\mu_{i,j,k})D_2[u_2]^n_{i,j,k} + \lambda_{i,j,k}D_3[u_3]^n_{i,j,k} + [f_{22}]^n_{i,j,k},\\
&D_t[\sigma_{33}]^n_{i,j,k} = \lambda_{i,j,k}D_1[u_1]^n_{i,j,k} + \lambda_{i,j,k}D_2[u_2]^n_{i,j,k} + (\lambda_{i,j,k} + 2\mu_{i,j,k})D_3[u_3]^n_{i,j,k} + [f_{33}]^n_{i,j,k},\\
&D_t[\sigma_{23}]^n_{i,j+1/2,k+1/2} = \mu_{i,j+1/2,k+1/2}(D_2[u_3]^n_{i,j+1/2,k+1/2} + D_3[u_2]^n_{i,j+1/2,k+1/2}) + [f_{23}]^n_{i,j+1/2,k+1/2},\\
&D_t[\sigma_{13}]^n_{i+1/2,j,k+1/2} = \mu_{i+1/2,j,k+1/2}(D_3[u_1]^n_{i+1/2,j,k+1/2} + D_1[u_3]^n_{i+1/2,j,k+1/2}) + [f_{13}]^n_{i+1/2,j,k+1/2},\\
&D_t[\sigma_{12}]^n_{i+1/2,j+1/2,k} = \mu_{i+1/2,j+1/2,k}(D_1[u_2]^n_{i+1/2,j+1/2,k} + D_2[u_1]^n_{i+1/2,j+1/2,k}) + [f_{12}]^n_{i+1/2,j+1/2,k}.
\end{aligned} \tag{15}$$

The stability condition for scheme (15) has the form of $c_p\sqrt{\left(\frac{1}{h_1^2} + \frac{1}{h_2^2} + \frac{1}{h_3^2}\right)} \leq 1$, where c_p is the longitudinal wave velocity in the material.

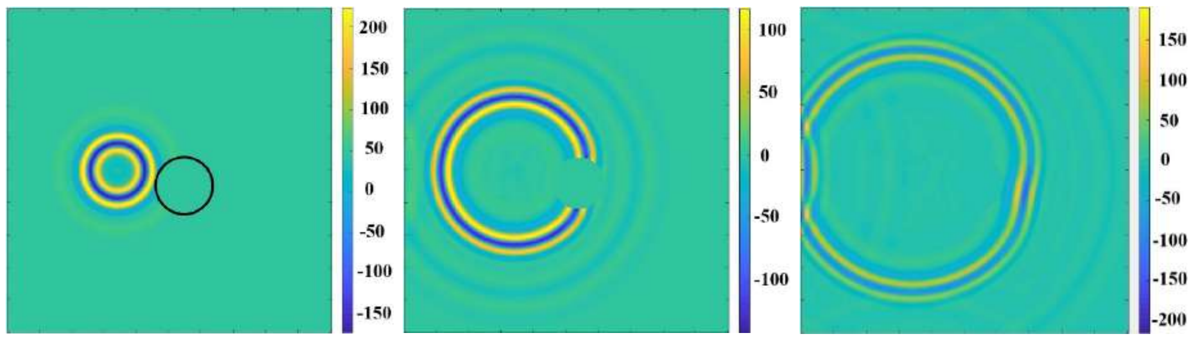
4.2. Model Computations of the Direct Problem of Wave Propagation in Thin Plates

The problem of computing the wave field in thin plates is of particular interest in this work. Due to the complex effects of interference in thin plates, the resulting waves are so-called normal waves or Lamb waves. We consider Lamb waves in plates with a thickness comparable to the wavelength. Lamb waves have different modes, each traveling at a different speed. A distinction is made between symmetric modes (denoted as Si) and asymmetric modes (denoted as Ai) of Lamb waves. In symmetric modes, the upper and lower surfaces of the plate move in opposite directions along the Z-axis perpendicular to the surfaces, while in asymmetric modes the surfaces move in the same direction.

An important feature of Lamb waves is that for a given plate thickness only the zero-order A_0 and S_0 modes are excited for frequencies below a certain threshold frequency. As the frequency increases, additional higher-order modes appear. In tomographic experiments, it is advisable to use the frequencies below this threshold frequency to avoid the appearance of additional modes. The fewer modes are present, the simpler and more reliable are the experimental data obtained. The A_0 mode is excited much more easily than the S_0 mode by striking the plate from above. Furthermore, with this type of pulse excitation, as shown by model computations, the amplitude of the A_0 mode is significantly higher than the amplitude of the S_0 mode. The propagation velocities of the A_0 and S_0 modes in the frequency range under consideration differ by approximately a factor of 2, which allows for reliable separation of these modes in processing the experimental data for solving inverse problems of thin plate diagnostics.

As an example, we present a computation of the wave field in a plate (Fig. 3) with a notch-shaped defect, with a depth smoothly increasing from zero toward the defect center as a cosine function. The parameter values used in the computation effectively excite only the A_0 Lamb wave mode:

- $\omega = 75$ kHz is the central frequency of the pulse;
- $R = 5$ mm is the radius of the circle of force application;
- L= 62.5 cm is the plate length;
- d= 1.0 cm is the plate thickness;
- $a_1 = 106.3$ mm, $b_1 = 156.3$ mm are the coordinates of the center of the force application;
- $L = 8.5$ cm; $h = 0.5$ cm are the diameter and maximum depth of the defect.



(a) 25 μs (b) 50 μs (c) 75 μs

Figure 3. Velocity u_3 on the plate surface at different moments of time

As can be seen from this figure, the defect significantly distorts the wavefront (the defect is highlighted by the black line in Fig. 3a. This distortion appears due to the fact that the residual plate thickness under the defect is 70% of the initial plate thickness, which leads to a decrease in the propagation velocity of the A_0 Lamb waves, since their velocity decreases with decreasing plate thickness.

4.3. Numerical Method for Solving the Inverse Problem of Plate Thickness Reconstruction

The inverse problem of plate thickness reconstruction was solved using a two-dimensional scalar model approximation, described by the Helmholtz equation (12) for a certain set of frequencies. To numerically solve the inverse problem of minimizing the residual functional $\Phi\left(d\left(r\right)\right)$ (13), an iterative process was constructed consisting of the following steps:

1. The initial approximation at the first iteration is $d_1\left(r\right)=d_0=const$, where d_0 is the thickness of the plate without defects.
2. For a given $d_n\left(r\right)$, the direct problem (12) of computing the wave field is solved using the finite difference approximation (16).
3. Using the simulated measured values of the wave field on the detectors $U\left(w_i, s, q_j\right)$ and computed $u\left(w_i, s, q_j, d\left(r\right)\right)$ for a given $d\left(r\right)$, the adjoint problem is solved in finite difference approximation.
4. Using the obtained values $z\left(w_i, r, q_j, d\left(r\right)\right)$ and $u\left(w_i, r, q_j, d\left(r\right)\right)$, the gradient of the residual functional $\Phi'_d\left(d\left(r\right)\right)$ is computed by formula (14).
5. Knowing the gradient, we compute the next iterative approximation using the formula $d_{n+1}\left(r\right)=d_n\left(r\right)\text{-}\gamma\left(n\right)\Phi'_d\left(d\left(r\right)\right)$. The step size at the first iteration is determined based on prior considerations. If the value of the residual functional increases at the next iteration, $\gamma\left(n\right)$ is reduced by a factor of 2. The process returns to step 2.

The Helmholtz equation is discretized on a uniform rectangular grid, and the corresponding derivatives are approximated by finite differences using standard formulas. The discretized Helmholtz equation takes the following form:

$$\frac{u_{i+1j} - 2u_{ij} + u_{i-1j}}{\Delta x^2} + \frac{u_{ij+1} - 2u_{ij} + u_{ij-1}}{\Delta y^2} + \frac{\omega}{c^2(\omega,\ d(r_{ij}))} u_{ij} = F_{i,j}. \qquad (16)$$

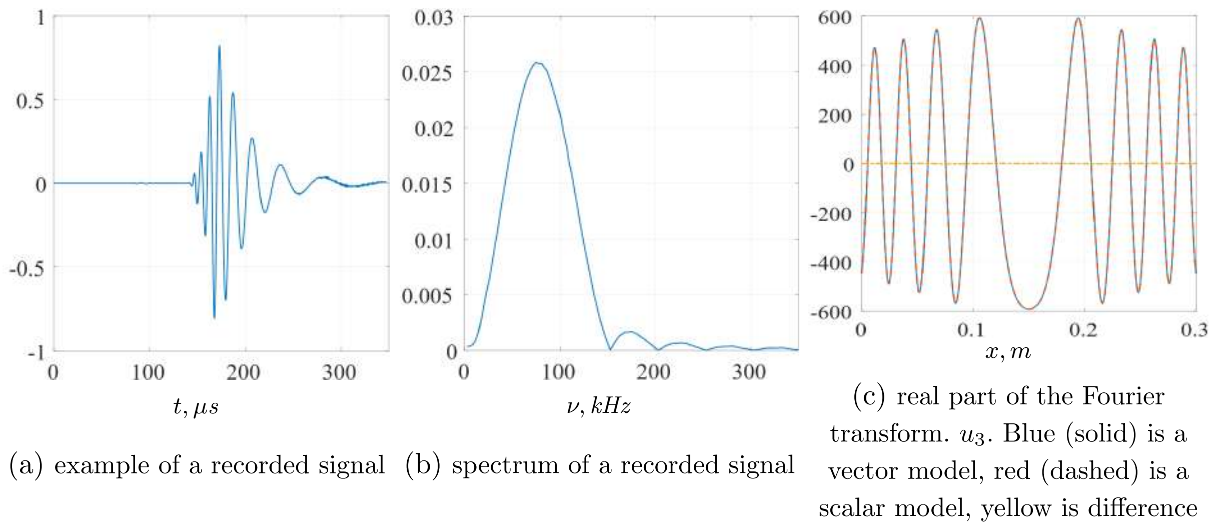
The system of algebraic equations is constructed as the indices $i,\ j$ run through all the values within the workspace. The matrix of this system is sparse and stored in CSR format. The Eigen library is used for sparse matrix manipulation and solving the problem. We will discuss the implementation details in Section 5.

4.4. Model Computations of the Inverse Problem of Plate Thickness Reconstruction

A series of computational experiments were conducted on solving the inverse problem of surface defect imaging. The experiment involved solving the forward problem using a three-dimensional vector wave model and recording the detector data. These data were then used as simulated experimental data to solve the inverse problem for a two-dimensional scalar wave equation.

The direct problem was solved for a three-dimensional thin steel plate (Fig. 2) with a thickness d and a size of $L \times L$. Free boundary conditions were applied on the upper and lower boundaries. The size of the computational domain and the locations of the sources and detectors were chosen such that the wave reflected from the boundary of the computational domain arrived at the detector sufficiently later than the wave reflected from the defect. A square workspace is located in the center of the computational domain. Defects, which are cylindrical depressions on the underside of the plate, are located within the workspace. Sources and detectors are located at the edges of the workspace on the upper side of the plate.

The sources generate sounding pulses defined by formula (5). The detectors are uniformly distributed around the perimeter of the workspace and record the vertical component of the displacement velocity $u_3(t, r)$ on the upper side of the plate. Data is recorded with the same time step as in the forward problem calculation. For each source-detector pair, the data recorded is a function of time. For each detector, the data undergoes a Fourier transform, and the result is used as $U(w, s; q_j)$ to solve the inverse problem for the Helmholtz equation. An example of a recorded signal is shown in (Fig. 4a), and a plot of the Fourier transform modulus for one detector is shown in (Fig. 4b).



(a) example of a recorded signal (b) spectrum of a recorded signal (c) real part of the Fourier transform. u_3. Blue (solid) is a vector model, red (dashed) is a scalar model, yellow is difference

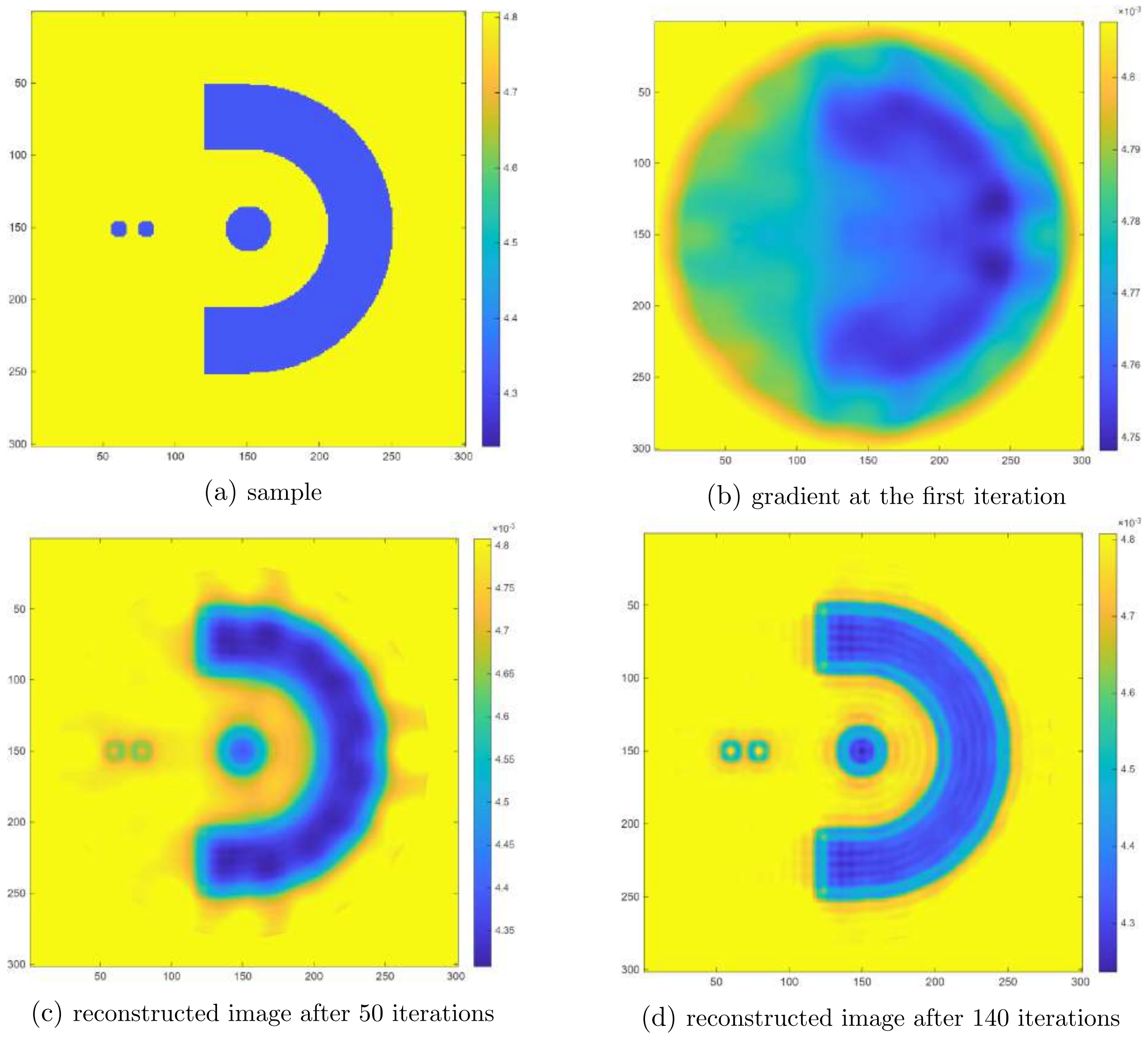
Figure 4. Signal and its spectrum and comparison u_3 in vector model with u_3 in scalar model

A key aspect in solving the inverse problem in a scalar two-dimensional wave model described by the Helmholtz equation is the correct selection of the amplitude and phase of the source perturbation and the wave propagation velocity for each harmonic. These parameters cannot be obtained explicitly with the required accuracy, even if the source and medium parameters in the forward three-dimensional problem for an elastic vector medium are known. In a real physical experiment, these parameters are unknown or approximate, making the case even worse.

In order to determine the missing parameters, data fitting was performed on a single-source problem with standardly positioned detectors on a defect-free plate. The vector model was used to compute the wave field at the detectors, perform a Fourier transform, and obtain complex-valued data at the detectors for the selected frequency. The phase, velocity, and amplitude of the harmonics in the scalar model were determined by minimizing the norm of the difference between the complex-valued data at the detectors for a given frequency in the scalar and vector problems.

Figure 4c shows the data fitting error between the two models. Blue line represents the data from the vector model, red line represents the data from the scalar model, and yellow line represents their difference. The root-mean-square fitting error is approximately 0.02%. The procedure for determining these parameters was performed for each frequency used in solving the inverse problem.

To demonstrate the algorithm for solving the inverse problem of plate thickness imaging, a computational experiment was conducted with 12 sources located 10 cm inwards from the midpoints of the sides of a 30×30 cm square computational domain. 1200 detectors were placed equally spaced around the perimeter of the square. The computations used 41 frequencies in the range of 23–147 kHz. The simulated sample object contains a collection of defects in the form of cylindrical depressions.



(a) sample

(b) gradient at the first iteration

(c) reconstructed image after 50 iterations

(d) reconstructed image after 140 iterations

Figure 5. Simulation experiment

Figure 5 shows the sample, the gradient at the first iteration, and the reconstructed image after 50 and 140 iterations. The results demonstrate that defects are localized with high accuracy, and their shape is precisely reconstructed. A large number of model simulations were performed with defects of varying shapes and depths. These simulations demonstrated that the developed algorithm reliably detects both point-like and extended defects, precisely reconstructs their shape and depth, and has high spatial resolution.

5. Performance Analysis of a Supercomputer Implementation for Solving Inverse Problems of Ultrasound Tomography Using the Lamb Wave

Solving the inverse problems under consideration requires huge computational resources. During the iterative solution process, the Helmholtz differential equation (12) must be solved for both the direct and the adjoint problem at each iteration, for each source position, and for each frequency used. Typical numbers of sources and frequencies can vary from a few to several dozen. For the model computations presented in Section 4.4, 12 sources and 41 frequencies were used, in which case the Helmholtz equation must be solved $12 \times 41 \times 2 = 984$ times for every iteration.

A finite-difference approximation was used for the Helmholtz equation, resulting in a system of linear algebraic equations (SLAE), the solution of which is the most computationally expensive and time-consuming task. Third-party linear algebra libraries were used for SLAE solution. The SLAE matrix has a size equal to the number of grid points and is highly sparse, with a fill-factor of approximately 0.0008%. The matrix is stored in CSR format. For the model computations from Section 4.4, the matrix size was approximately $6.4 \cdot 10^5 \times 6.4 \cdot 10^5$ with the number of nonzero elements approximately 3.2×10^6. Note that the matrix in this problem is not symmetric.

An inverse problem solution algorithm was implemented using C++ language plus MPI for parallelization, and Eigen library [29] to fill in the matrix, the right-hand side, and solving the SLAE. In this Section, we analyze the performance of the developed application for solving inverse problems. The computations were performed in the "compute" section of the Lomonosov-2 supercomputer [30]. In this section, each node contains a single 14-core Intel Xeon E5-2697 v3 CPU with 64 GB of RAM. The application was run on 164 processes. Intel MPI 2017 was used to compile and run the distributed MPI version. For this performance analysis, the program was configured to run for ∼5 minutes, this was achieved by reducing the number of iterations. The iterations are essentially identical, so the overall picture in this case matches the program's behavior when using the number of iterations used for real-life calculations.

First, it was necessary to determine the optimal number of processes per node. This program is quite memory-intensive, and in such cases, it is sometimes beneficial to occupy only a part of processor cores in order to reduce memory contention. Experiments with 8 to 14 processes per node were performed in the "compute" section, with 10 launches of every experiment performed in order to obtain enough statistics. A smaller number of processes per node was not considered in this case because it involves too many compute nodes, which was considered inexpedient.

Table 1 shows the minimum execution times obtained. We consider the minimum value, not average or mean, because supercomputers often experience different external factors that can slow down a user program (such as influence of other users' applications running concurrently, the overload of distributed file system, OS noise, etc.). This leads to periodic anomalous or simply slower executions, which have notably longer execution times and are of no interest in our case. But we note that the average execution time behaves generally similarly to the minimum time. Considering the minimum time allows us to estimate how quickly a program can potentially execute under the chosen parameters, without the influence of external factors.

Table 1. Program execution time with different numbers of processes per node

Processes per node	8	9	10	11	12	13	14
Minimum execution time, s	360	360	294	222	222	228	228

It can be seen that the program executes the slowest at 8–9 processes per node, then the execution time begins to decrease, reaching a minimum at 11–12 processes per node. This is presumably due to the fact that fewer compute nodes are required as the number of processes per node increases, and, as a result, the amount of data transferred between nodes decreases (i.e., the communication decreases). The execution time then increases slightly, likely due to the aforementioned effect of memory contention between processes on the node. Thus, the minimum execution time is achieved at 11–12 processes per node. However, it should be taken into account that when running user programs on a supercomputer, a queuing system is used, and the more nodes a program requires, the longer it will usually wait in the queue for the required number of nodes to become available. Given that the difference in execution time between the versions with 11–12 and 14 processes per node is small (a slowdown of $\sim$3%), but fewer nodes are used (difference in two nodes, or 16.7%), it is recommended to run this program on 14 processes per node, thereby utilizing all available processor cores. This will both speed up the job's progress through the queue and reduce the overall resource consumption in terms of node-hours, thereby saving resources for other users' jobs.

Next, we analyzed the program's performance, using 14 processes per node. First, we examined the MPI usage in the program. mpiP 3.5 profiling tool [31] was used for this purpose. Analysis of its results showed that program processes spend 3–10% of their execution time on MPI, which can be considered as a good result for a program running on 164 processes and 12 compute nodes. It should be noted that MPI usage characteristics do not differ significantly for individual processes, i.e., there is no significant imbalance in the program.

Approximately 70–75% of the execution time spent on MPI is accounted for by just two `MPI_Allreduce` calls, with almost all of the remaining time accounted for by two `MPI_Bcast` calls. It is interesting to note that both `MPI_Allreduce` calls are responsible for transferring a tiny amount of data – less than 0.1% of the total amount of transferred data (while `MPI_Bcast` calls are responsible for transferring the rest of the data). Therefore, if there is a need to optimize MPI usage, these `MPI_Allreduce` calls can clearly be singled out as the main candidates, since they account for the majority of the time, but they transfer very little data. One could assume that the reason for their long execution is related to imbalance caused by some processes reaching this MPI operation earlier and simply waiting for the other processes to start, but this is not always the case. In some cases, all involved processes spent significant time performing these operations. The exact reasons for this behavior are currently unknown; however, the total time spent on MPI in the current program configuration is quite small, so further detailed analysis and optimization in this direction is not of great interest at the moment (this may change in the future if a larger number of processes are involved, likely leading to the increase of MPI share in execution time).

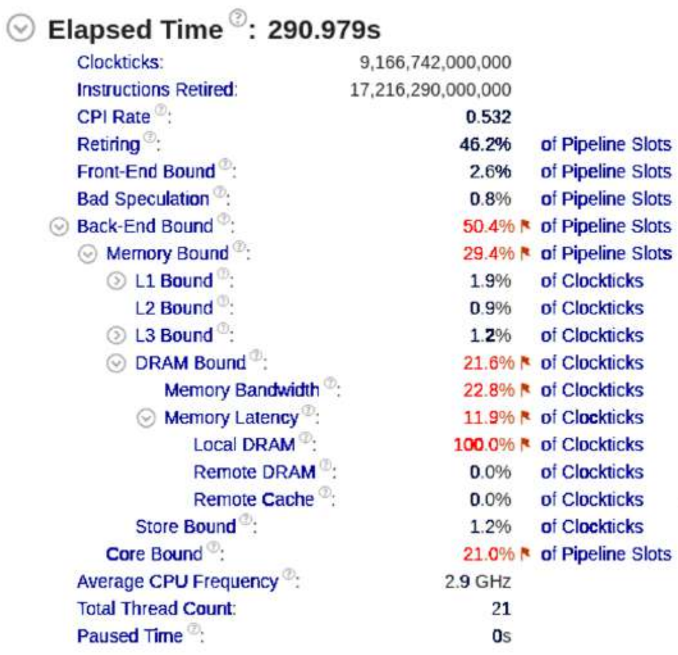
After studying MPI usage, an analysis of the execution efficiency of one individual process was conducted. The program is generally well-balanced, so the overall results for one process presented below hold true for others as well. The analysis was performed using Intel VTune 2019.5 and Intel Advisor 2019.5.

Initially, the most general performance characteristics of parallel programs were studied:

- The average number of cycles per instruction (CPI) is ∼0.5, i.e., two instructions are executed per clock cycle on average, which is high for real-world HPC applications.
- The utilization of processor cores is also quite high, at 85–90%.
- The Retiring metric, which roughly shows the percentage of time the CPU was fully utilized by performing useful operations, is of interest as well. This metric is calculated using a Top-down approach proposed by Intel [32], the goal of which is to identify the dominant bottlenecks in an application performance. For this program, the value of this metric is 45–55%, which indicates a fairly efficient HPC program (according to Intel, an expected range of Retiring value for well-tuned HPC programs is 30–70% [32]).

Intel VTune allows collecting various useful information using the Top-down approach. However, Hyperthreading is enabled on compute nodes in the “compute” partition, making some Top-down metrics unavailable in VTune. Therefore, to analyze these metrics, we conducted special launches of the program in the “pascal” partition of the Lomonosov-2 supercomputer. The nodes in this partition are equipped with a different processor – a 12-core Intel Xeon Gold 6126. However, the comparison showed that the overall picture provided by Top-down approach almost does not change across runs in different partitions, so here we show a more complete list of metrics after running in the “pascal” partition.

Figure 6 shows the Top-down metrics for this program provided by Intel VTune. It can be seen that there are no issues with data prefetch or instruction preparation (Bad Speculation and Front-End Bound metrics, respectively). In this case, the application is classified as Back-End Bound (50%), meaning that processor slots were frequently only partially occupied, with the main reason for this being memory management (Memory Bound = 29%). It should be noted here that Core Bound in the screenshot is just slightly lower than Memory Bound, and this is the only noticeable difference from what is observed in the “compute” section (being of primary interest for us), where Core Bound is 10%, being a small value. Therefore, we do not consider this issue, since it does not occur when running in “compute”.



```
Elapsed Time: 290.979s
  Clockticks:                          9,166,742,000,000
  Instructions Retired:               17,216,290,000,000
  CPI Rate:                                        0.532
  Retiring:                                      46.2%    of Pipeline Slots
  Front-End Bound:                                2.6%    of Pipeline Slots
  Bad Speculation:                                0.8%    of Pipeline Slots
  Back-End Bound:                                50.4%    of Pipeline Slots
    Memory Bound:                                29.4%    of Pipeline Slots
      L1 Bound:                                   1.9%    of Clockticks
      L2 Bound:                                   0.9%    of Clockticks
      L3 Bound:                                   1.2%    of Clockticks
      DRAM Bound:                                21.6%    of Clockticks
        Memory Bandwidth:                        22.8%    of Clockticks
        Memory Latency:                          11.9%    of Clockticks
          Local DRAM:                           100.0%    of Clockticks
          Remote DRAM:                            0.0%    of Clockticks
          Remote Cache:                           0.0%    of Clockticks
      Store Bound:                                1.2%    of Clockticks
    Core Bound:                                  21.0%    of Pipeline Slots
  Average CPU Frequency:                         2.9 GHz
  Total Thread Count:                                 21
  Paused Time:                                        0s
```

Figure 6. Top-down metrics obtained by Intel VTune

In terms of memory usage, this program exhibits no issues with any cache levels (L1/L2/L3 Bound values are very small). However, some problems with RAM can be noted. These include mainly memory bandwidth issues (the program reaches maximum bandwidth to RAM, likely causing a slowdown) as well as some memory latency issues (the program has to wait for data from RAM, as the retrieval process takes notable time). The values for these metrics are not very high, meaning the issues are apparently not that significant, but they are still present.

The presence of performance issues with RAM is as well confirmed by studying the values of performance monitoring counters also provided by VTune. They show that when accessing L3 cache, there are 2.5 misses for every hit on average (and L3 miss necessitates an access to RAM), which is a very poor indicator. Furthermore, one L3 cache miss occurs for every 200 memory accesses, meaning the L3 is used quite heavily. This suggests that memory access is organized not in the most efficient manner, leading to fairly frequent RAM accesses and, consequently, waiting for data from memory.

The vectorization level using Intel Advisor was also studied. It revealed that the program is not vectorized at all: only 0.1% of the execution time is spent in vectorized code fragments. Furthermore, vectorization does not use the newer AVX instructions, only SSE and SSE2. This, however, is currently not particularly significant given that almost all the code is scalar.

A separate study of the most computationally intensive fragments (those that consume the most time during program execution) was conducted. A general analysis on the level of modules revealed that the program spends >90% of its execution time within its own code, with another ∼5% executing MPI functions, 1.5% interacting with the OS kernel (vmlinux), and ∼1% calling the libm math library. Within the program itself, almost all of the time is spent executing the code from the aforementioned Eigen library.

Thus, the following conclusions can be drawn from the analysis performed. The program demonstrates relatively high performance, with MPI operations taking up only a small portion of the program's execution time – ∼5% of the total runtime, which is quite low for a program with 164 processes. Certain performance issues with RAM usage were noted; they do not significantly slow down the program but do represent a potential area for optimization. Vectorization is essentially absent, which is also a clear candidate for optimization (if this area is further to be explored, a detailed analysis of the non-vectorized code fragments will be required to determine the feasibility of vectorization). However, everything discussed regarding memory and vectorization applies to the usage of external Eigen library, as almost all execution time not spent on MPI is spent working with it, which significantly complicates the possibility of making changes and optimizations. One of the possible solutions in this case is to switch to using other libraries for solving SLAE like MKL.

Conclusion

This study examines the challenges of supercomputer implementation of ultrasonic tomographic imaging of internal and surface defects for nondestructive testing of solids. It has been shown that the imaging problem of ultrasonic imaging of thin plates using Lamb waves is a coefficient inverse problem, i.e., it involves determining unknown coefficients in differential equations describing wave propagation in scalar and vector models.

It has been shown that a scalar two-dimensional model is quite suitable for detecting surface defects. A simplified formulation of the inverse problem assumes that the propagation of Lamb waves of mode A_0 is approximately described by the Helmholtz equation. The experimental data were obtained from a numerical solution of the direct problem in a 3D elastic model in time domain. An explicit expression for the residual functional gradient with respect to the plate thickness was formulated for a range of frequencies. Iterative methods have been developed to solve the inverse problem. The methods are based on gradient descent methods to minimize the residual functional. Efficiency of the proposed algorithms is illustrated on model problems.

Within this simplified model, tomographic imaging technique makes it possible to reconstruct the shape, size, and depth of defects. Solving the Helmholtz equation is the core element of the developed algorithms for solving inverse problems of wave tomography. The most demanding computations involve solving linear equations with large-scale sparse matrices using LU-decomposition method. The algorithms were implemented using linear algebra Eigen library. The performance analysis of the solution of the inverse problem in the scalar formulation of the Helmholtz equation (with Eigen library used for solving SLAE) was performed on the Lomonosov-2 supercomputer.

Acknowledgements

This research was supported by Russian Science Foundation grant No. 25-11-00089, https://rscf.ru/en/project/25-11-00089/. This research was carried out at the Lomonosov Moscow State University. The results in Section 2 were obtained within the framework of the state assignment of Lomonosov Moscow State University "Development of Methods for Solving Inverse Problems of Diagnostics and Synthesis in Wave Models". The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. Ruiter, N.V., Zapf, M., Hopp, T., *et al.*: USCT data challenge. Proc. SPIE. Medical Imaging: Ultrasonic Imaging and Tomography 10139, 101391N (2017).

2. Duric, N., Littrup, P., Sak, M., *et al.*: A novel marker, based on ultrasound tomography, for monitoring early response to neoadjuvant chemotherapy. J. Breast Imaging 2(6), 569–576 (2020). `https://doi.org/10.1093/jbi/wbaa084`

3. Wiskin, J., Borup, D., Andre, M., *et al.*: Three-dimensional nonlinear inverse scattering: Quantitative transmission algorithms, refraction corrected reflection, scanner design, and clinical results. J. Acoust. Soc. Am. 133(5), 3229–3229 (2013). `https://doi.org/10.1121/1.4805138`

4. Lucka, F., Pérez-Liva, M., Treeby, B.E., Cox, B.T.: High resolution 3D ultrasonic breast imaging by time-domain full waveform inversion. Inverse Problems 38(2), 025008 (2022). `https://10.1088/1361-6420/ac3b64`

5. Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Low-frequency ultrasonic tomography: Mathematical methods and experimental results. Moscow University Physics Bulletin 74(1), 43–51 (2019). `https://doi.org/10.3103/S0027134919010090`

6. Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Supercomputer technologies in tomographic imaging applications. Supercomputing Frontiers and Innovations 3(1), 41–66 (2016). https://doi.org/10.14529/jsfi160103

7. Goncharsky, A.V., Kubyshkin, V.A., Makan, I.I., *et al.*: Supercomputer simulations in designing medical ultrasound tomographic imaging devices. Lobachevskii Journal of Mathematics 45(7), 3038–3050 (2024). https://doi.org/10.1134/S199508022460393X

8. Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Multistage Iterative Method to Tackle Inverse Problems of WaveTomography. Supercomputing Frontiers and Innovations 9(1), 87–107 (2022). https://doi.org/10.14529/jsfi220106

9. Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Computational Efficiency of Iterative Methods for Solving Inverse Problems. In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds.) Supercomputing. RuSCDays 2023. Lecture Notes in Computer Science, vol. 14388, pp. 35–46. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-49432-1_3

10. Bazulin, E.G., Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Parallel CPU- and GPU-algorithms for inverse problems in nondestructive testing. Lobachevskii Journal of Mathematics 39(4), 486–493 (2018). https://doi.org/10.1134/S1995080218040030

11. Yang, P., Brossier, R., Metivier, L., Virieux, J.: A review on the systematic formulation of 3-D multiparameter full waveform inversion in viscoelastic medium. Geophys. J. Int. 207, 129–149 (2016). https://doi.org/10.1093/gji/ggw262

12. Virieux, J., Asnaashari, A., Brossier, R., *et al.*: An introduction to full waveform inversion. Society of Exploration Geophysicists 6, R1-1–R1-40 (2014). https://doi.org/10.1190/1.9781560803027.entry6

13. Pan, W., Innanen, K.A.: Elastic full-waveform inversion: density effects, cycle-skipping, and inter-parameter mapping. CREWES Research Report 28, 62.1–62.18 (2016).

14. Li, Y., Gu, H.: Full waveform inversion for velocity and density with rock physical relationship constraints. Journal of Applied Geophysics 167, 106–117 (2019). https://doi.org/10.1016/j.jappgeo.2019.04.005

15. Lechleiter, A., Schlasche, J.W.: Identifying Lamé parameters from time-dependent elastic wave measurements. Inverse Problems in Science and Engineering 25(1), 2–26 (2017). https://doi.org/10.1080/17415977.2015.1132713

16. Plessix, R.E.: A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. Geophys. J. Int. 167, 495–503 (2006). https://doi.org/10.1111/j.1365-246X.2006.02978.x

17. Bazulin, E.G., Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Inverse problems of ultrasonic tomography in nondestructive testing: Mathematical methods and experiment. Russian Journal of Nondestructive Testing 55(6), 453–462 (2019). https://doi.org/10.1134/S1061830919060020

18. Huthwaite, P., Simonetti, F.: High-resolution guided wave tomography. Wave Motion 50(5), 979–993 (2013). https://doi.org/10.1016/j.wavemoti.2013.04.004

19. Rao, J., Ratassepp, M., Fan, Z.: Guided Wave Tomography Based on Full Waveform Inversion. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control 63(5), 737–745 (2016). https://doi.org/10.1109/TUFFC.2016.2536144

20. Zhao, X., Rose, J.L.: Ultrasonic guided wave tomography for ice detection. Ultrasonics 67, 212–219 (2016). https://doi.org/10.1016/j.ultras.2015.12.005

21. Tong, J., Lin, M., Wang, X., *et al.*: Deep learning inversion with supervision: A rapid and cascaded imaging technique. Ultrasonics 122, 106686 (2022). https://doi.org/10.1016/j.ultras.2022.106686

22. Landau, L.D., Lifshitz, E.M.: Course of Theoretical Physics, Vol. 7: Theory of Elasticity. Pergamon, Oxford (1995).

23. Engquist, B., Majda, A.: Absorbing boundary conditions for the numerical simulation of waves. Math. Comput. 31, 629–629 (1977). https://doi.org/10.1090/s0025-5718-1977-0436612-4

24. Viktorov, I.A.: Lamb's Ultrasonic Waves. Soviet Physics. Acoustics 11(1), 1–18 (1965).

25. Natterer, F., Sielschott, H., Dorn, O., *et al.*: Fréchet derivatives for some bilinear inverse problems. SIAM J. Appl. Math. 62(6), 2092–2113 (2002). https://doi.org/10.1137/S0036139901386375

26. Virieux, J.: P-SV Wave Propagation in Heterogeneous Media: Velocity-Stress Finite-Difference method. Geophysics 51(4), 889–901 (1986). https://doi.org/10.1190/1.1442147

27. Lisitsa, V.V.: Numerical Methods and Algorithms for Calculating Wave Seismic Fields in Media with Local Complicating Factors. Doctoral Thesis in Physics and Mathematics (Trofimuk Institute of Petroleum Geology and Geophysics SB RAS, Novosibirsk, 2017). https://www.dissercat.com/content/chislennye-metody-i-algoritmy-rascheta-volnovykh-seismicheskikh-polei-v-sredakh-s-lokalnymi, accessed: 2023-07-10

28. Belyaev, A.S., Goncharsky, A.V., Romanov, S.Y.: Development of Numerical Algorithms for Solving the Direct Problem of Propagation of Ultrasonic Waves in Thin Plates. Numerical Methods and Programming 24(3), 275–290 (2023). https://doi.org/10.26089/NumMet.v24r320 (in Russian)

29. Eigen as a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. https://eigen.tuxfamily.org/, accessed: 2025-11-14

30. Voevodin, Vl., Antonov, A., Nikitenko, D., *et al.*: Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). https://doi.org/10.14529/jsfi190201

31. Vetter, J., Chambreau, C.: mpiP: Lightweight, Scalable MPI Profiling (2005). http://gec.di.uminho.pt/Discip/MInf/cpd1415/PCP/MPI/mpiP_%20Lightweight,%20Scalable%20MPI%20Profiling.pdf

32. Description of Top-Down approach. https://www.intel.com/content/www/us/en/docs/vtune-profiler/cookbook/2023-0/top-down-microarchitecture-analysis-method.html, accessed: 2025-10-15

DOI: 10.14529/jsfi250404

Efficient Algorithm Based on the Woodbury Formula for Modeling Multi-port Antenna Systems

***Rufina M. Tretiakova*[1,2], *Alexey V. Setukha*[1,2], *Ilya A. Mass*[1,2]**

This paper presents an efficient computational approach for calculating the characteristics of multi-port antenna systems using boundary integral equations. Modeling of antenna radiation is based on finding surface currents placed on the antenna structural elements. Numerical solution of integral equations for unknown currents is carried out by the Galerkin method using RWG basis functions. The antennas excitation is modeled by placing a system of lumped ports. A key challenge addressed is the calculating of mutual coupling characteristics (mutual impedance matrices, S-parameters) for various conditions of ports loading. This requires multiple solutions of a system of linear equations. In each case, the system matrix undergoes a change in blocks consisting of elements responsible for the interaction of the antenna ports with each other and with surface currents. To overcome this, an algorithm based on the Woodbury formula is developed, significantly reducing computational costs by leveraging the low-rank nature of port-related matrix modifications. The method's effectiveness is demonstrated for both wire and patch antenna arrays, showing substantial speedups – approximately proportional to the number of ports for direct solvers and significant gains for iterative solvers using mosaic-skeleton approximations while maintaining solution accuracy.

Keywords: matrix methods, numerical algorithms, computational electrodynamics, antenna radiation, boundary integral equations, Woodbury formula.

Introduction

The Method of Moments (MoM) approach allows for the simulation of the performance of antennas with metal and dielectric parts. In case of metallic antennas, the electromagnetic field has an integral representation via surface currents on their metal parts, and the problem of antenna radiation can be treated as the scattering problem of some primary field generated by the antenna itself. One common mathematical model for such fields is the lumped port model, also known as the delta-gap model [2]. With a known primary field, MoM reduces antenna simulation to a system of surface integral equations [2, 13]. Some common performance characteristics, such as impedance, S-parameters, and the Voltage Standing Wave Ratio (VSWR), can be expressed in terms of electric currents, which are the solution of these integral equations.

MoM holds an advantage over volume-discretization techniques, as it does not require meshing the entire problem space, only the surfaces. Furthermore, it inherently satisfies the radiation condition at infinity.

In multi-port systems, mutual coupling between ports significantly affects the overall radiation characteristics. Matrices of mutual impedance and S-parameters are central to the analysis of this coupling. These matrices are computed through a series of numerical simulations, where individual ports are activated sequentially as active sources while the others are terminated with passive loads (e.g., matched feeder lines). This approach requires one simulation for each active port, leading to specific computational challenges.

A well-known challenge of MoM is that discretizing the integral equations yields linear systems with large dense matrices. Solving these systems is the dominant computational cost of the numerical simulation.

[1]Research Computing Center, Lomonosov Moscow State University, Moscow, Russian Federation
[2]Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow, Russian Federation

For antenna systems, the main part of the system matrix describes the electromagnetic interaction between surface discretization elements (e.g., mesh cells). However, some rows and columns are modified according to the current active port. Thus, computing the complete mutual coupling matrices involves solving a series of linear systems, each with a modified system matrix. Such matrices can be represented as the sum of a fixed matrix (from the structure) and a low-rank correction per port configuration. A naive approach, solving each modified system independently, results in a computational cost proportional to the number of ports.

To address this inefficiency, we propose an algorithm based on the Woodbury matrix identity [3]. This identity allows for the efficient computation of the inverse of a matrix after it has been modified by a low-rank update. For a system of n equations, a direct solution via LU decomposition requires $\Omega = O(n^3)$ operations and iterative solution requires $\Omega = O(n^2 I_0)$ operations (I_0 is the number of iterations). Solving for m ports naively would therefore scale as $m\Omega$. Our algorithm leverages the Woodbury formula to obtain the exact solution for all port configurations by solving only one linear system with m right-hand sides. This requires $\Omega + O(nm^2 + m^3)$ operations, making the time for the multi-port solution comparable to that of a single simulation when $m \ll n$.

For systems with a moderate number of unknowns (up to approximately 40,000), we employ a direct solver. For larger problems, we use an iterative solver – the Generalized Minimal Residual Method (GMRES) [7] – accelerated by performing matrix-vector multiplications in the mosaic-skeleton format [10]. This format is a data-sparse approximation that operates on a compressed representation of the dense matrix [11]. The proposed method applies both direct and iterative solvers. We present computational examples of antenna arrays with numerous feed points, demonstrating the significant efficiency gains achieved by our method.

The article is organized as follows. The first section contains the statement of the problem, the boundary integral equation method, the numerical scheme and the description of antenna model. The algorithm based on the Woodbury matrix identity is also described in this section. The second section includes the testing of the developed algorithm on two examples: wire antenna array and patch antenna array. The discussion of the results is given in the conclusion section.

1. Mathematical Model

We consider a perfectly conducting antenna system. The antenna system includes a set of radiating elements connected to a feeder. Each such element can operate either for radiation or for signal reception.

Let Σ denote the union of all perfectly conducting surfaces forming the antenna structure. We consider the problem of antenna system radiation as a scattering problem. The primary field is created by a system of antenna ports. The antenna excitation is described in subsection 1.3.

Let us consider the general formulation of the problem of scattering a given primary field for monochromatic wave [13]. The electric and magnetic fields are sought in the form:

$$\vec{E}(x,t) = \mathbf{E}(x)e^{-\mathbf{i}\omega t},\ \vec{H}(x,t) = \sqrt{\frac{\varepsilon_0}{\mu_0}}\mathbf{H}(x)e^{-\mathbf{i}\omega t}, \tag{1}$$

where ε_0 and μ_0 are the electric and magnetic constants, ω is the angular frequency. The problem is reduced to the spatial components of the electric and magnetic fields $\mathbf{E}(x)$ and $\mathbf{H}(x)$.

The surrounding medium is assumed to be isotropic, homogeneous, and without conductivity. The electromagnetic field everywhere outside the antenna structural elements is described by Maxwell's equations. In the monochromatic case they have the following form:

$$\begin{aligned} \operatorname{rot}\mathbf{E} &= \mathbf{i}\omega k\mathbf{H}, \\ \operatorname{rot}\mathbf{H} &= -\mathbf{i}\omega k\mathbf{E}, \end{aligned} \tag{2}$$

here k is the wave number, defined by the formula $k = \omega\sqrt{\varepsilon_0\mu_0}$.

The magnetic field can be excluded from equation (2). The equations for the electric field are:

$$\Delta\mathbf{E} + k^2\mathbf{E} = 0, \quad \operatorname{div}\mathbf{E} = 0, \tag{3}$$

provided that the magnetic field has the form

$$\mathbf{H} = \frac{\operatorname{rot}\mathbf{E}}{\mathbf{i}k}. \tag{4}$$

We assume that the total electric and magnetic fields have the form [2, 13]

$$\mathbf{E}_{tot} = \mathbf{E}_{inc} + \mathbf{E}, \quad \mathbf{H}_{tot} = \mathbf{H}_{inc} + \mathbf{H}. \tag{5}$$

Here $\mathbf{E}_{inc}(x)$ and $\mathbf{H}_{inc}(x)$ are the primary electric and magnetic fields created by the antenna excitation, $\mathbf{E}(x)$ and $\mathbf{H}(x)$ are the unknown secondary electric and magnetic fields. These secondary fields must satisfy equations (3)–(4) outside the antenna structural elements. The total field must satisfy the boundary condition on the antenna surface Σ:

$$n \times \mathbf{E}_{tot} = 0, \quad x \in \Sigma. \tag{6}$$

Also the secondary fields have to satisfy the Sommerfeld radiation condition at infinity:

$$\mathbf{E}(x) = O\left(\frac{1}{|x|}\right), \quad \frac{\partial\mathbf{E}(x)}{\partial|x|} - ik\mathbf{E}(x) = o\left(\frac{1}{|x|}\right), \quad |x| \to \infty. \tag{7}$$

$$\mathbf{H}(x) = O\left(\frac{1}{|x|}\right), \quad \frac{\partial\mathbf{H}(x)}{\partial|x|} - ik\mathbf{H}(x) = o\left(\frac{1}{|x|}\right), \quad |x| \to \infty. \tag{8}$$

1.1. Method of Moments

In this subsection we describe the integral representation for the electric field and the integral equation system [2].

The electric field:

$$\mathbf{E}(x) = \frac{\mathbf{i}}{k_0}\mathcal{K}[\Sigma, \boldsymbol{g}](x), \quad x \in \mathbb{R}^3 \setminus \Sigma, \tag{9}$$

where $\mathcal{K}[\Sigma, \boldsymbol{g}]$ is the electric field operator:

$$\mathcal{K}[S, \boldsymbol{g}](x) = \operatorname{rot}\operatorname{rot}\int\limits_S \boldsymbol{g}(y)F(x-y)d\sigma_y. \tag{10}$$

In formula (10), S is random surface, $\boldsymbol{g}$ is a tangential vector field defined on the surface S, F is the Green's function $F(x-y) = \frac{e^{\mathbf{i}k|x-y|}}{|x-y|}$.

In expression (9), $\boldsymbol{g}$ is the unknown surface current.

The integral operator $\mathcal{K}$ in (10) is defined everywhere outside the integration domain S. The field defined by this operator for a sufficiently smooth function $\boldsymbol{g}$ has boundary values on both sides of the surface S, for which the formula [15] is:

$$\mathcal{K}^{\pm}[S,k,\boldsymbol{g}](x) = \mathcal{K}[S,k,\boldsymbol{g}](x) \pm 2\pi\, \boldsymbol{n}(x)\, \mathrm{Div}\, \boldsymbol{g}(x), \quad x \in S, \tag{11}$$

where x is a point of smoothness of the surface, not lying on its edge, $\boldsymbol{n}(x)$ is the unit vector of the positive normal at point x, the signs $\pm$ correspond to the boundary values from the side of the normal vector (from the positive side of the surface) and from the opposite side (from the negative side of the surface), respectively. Here $\mathcal{K}[S,k,\boldsymbol{g}](x)$ is the direct value of the operator on the surface. It arises if the value of the point $x \in S$ is substituted directly into the expression:

$$\mathcal{K}[S,\boldsymbol{g}](x) = \int\limits_{S} \mathrm{rot}\,_x \mathrm{rot}\,_x (\boldsymbol{g}(y)F(x-y))d\sigma_y. \tag{12}$$

In this case the hypersingular integral on the right-hand side is a Hadamard finite part integral.

We substitute expression (9) into the boundary condition (6) and apply the formula (11) for the boundary values of the integral operator (11) to obtain the boundary integral equation for the unknown function $g \in \Sigma$:

$$\frac{\mathbf{i}}{k_0}\boldsymbol{n}(x) \times \mathcal{K}[\Sigma,\boldsymbol{g}](x) = -\boldsymbol{n} \times \mathbf{E}_{inc}(x), \quad x \in \Sigma. \tag{13}$$

Note that in equation (13), the operator $\mathcal{K}$ is defined by the formula (12) with the hypersingular integral on the right-hand side.

We solve these equations and substitute the obtained currents into formulas (9) and (4) to calculate the values of the secondary electric and magnetic fields at an arbitrary point outside the surface Σ.

1.2. Numerical Method

For the numerical solution of equation (13) we employ a widely used numerical scheme of the Galerkin method with piecewise linear basis functions (RWG functions). This numerical scheme was first described in [6] and its modern version is described in [2, 13].

The surface Σ is approximated by a conformal system of triangular cells. Let $\tilde{\Sigma} = \bigcup_{i=1,N} \sigma_i$ be the approximation of the original total surface Σ.

The approximation of the unknown currents g is a linear combination of RWG basis functions [6].

$$\boldsymbol{g}(x) = \sum_{i=1,M} g_i \boldsymbol{v}_i(x) \tag{14}$$

Each basis function $\boldsymbol{v}_i(x)$, $i = 1,...,M$ is associated with an edge common to two cells σ_i^1 and σ_i^2 and characterizes the flux of the vector field through this edge. If an edge is common to exactly two mesh cells, then one basis function from the list of functions with indices $i = 1,...,M$ is associated with this edge. If an edge is a junction of $p > 2$ cells $\sigma_{j_1},...,\sigma_{j_p}$ (these cells are numbered in a clockwise order), then $p-1$ basis functions are associated with this edge. Each basis function corresponds to a pair of cells $(\sigma_{j_1},\sigma_{j_2})$, $(\sigma_{j_{p-1}},\sigma_{j_p})$. Thus an ordered pair of cells σ_i^1 and σ_i^2 is associated with each basis function $\boldsymbol{v}_i(x)$, $i = 1,...,M$ (see [14]). Let s_i^1 and s_i^2 be the areas of cells σ_i^1 and σ_i^2. We also denote the vertices that lie opposite to the common edge of cells σ_i^1 and σ_i^2 as C_i^1 and C_i^2 for each basis function $\boldsymbol{v}_i(x)$.

The Bubnov–Galerkin method is used for the numerical solution of the boundary integral equations $(\boldsymbol{v}_i, \mathcal{K}\boldsymbol{v}_j)_{\tilde\Sigma} = (\boldsymbol{v}_i, \mathcal{K}[\tilde\Sigma, \boldsymbol{v}_j])_{\tilde\Sigma}$. The scalar product of two vector functions is understood as the integral over the surface Σ: $(\boldsymbol{g}, \boldsymbol{f})_{\tilde\Sigma} = \int_{\tilde\Sigma}(\boldsymbol{g}, \boldsymbol{f})d\sigma$.

The scalar product with the electric operator is defined by the following expression:

$$(\boldsymbol{v}_i, \mathcal{K}\boldsymbol{v}_j)_{\tilde\Sigma} = \int\limits_{\sigma_i^1\cup\sigma_i^2}\int\limits_{\sigma_j^1\cup\sigma_j^2} \left(k^2\boldsymbol{v}_i(x)\cdot\boldsymbol{v}_j(y) - D_iD_j\right) F(x-y)d\sigma_y d\sigma_x =$$

$$= \sum_{p=1,2}\sum_{q=1,2}\frac{(-1)^{p+q}}{s_i^p s_j^q}\int\limits_{\sigma_i^p}\int\limits_{\sigma_j^q}\left(k^2(C_i^p - x)\cdot(C_j^q - y) - 4\right)\frac{e^{\mathbf{i}kr}}{r}d\sigma_y d\sigma_x. \quad (15)$$

To compute the integral in expression (15), we use formulas with analytical extraction and integration of the singularity [2]. For regular integrals we developed an adaptive integration procedure based on Gaussian quadrature with subdivision of cells into smaller ones and accuracy control.

The system of boundary integral equations (13) is reduced to a system of linear algebraic equations.

$$\frac{\mathbf{i}}{k_0}\sum_{j=1}^{M}(\boldsymbol{v}_i, \mathcal{K}\boldsymbol{v}_j)_{\tilde\Sigma}\; g_j = -(\boldsymbol{v}_i, \mathbf{E}_{inc})_{\tilde\Sigma}, \quad i = 1, ..., M. \quad (16)$$

After the system (16) is solved for the variables g_i, $i = 1, ..., M$, the surface currents $\boldsymbol{g}$ are computed using formula (14).

1.3. Antenna Model

1.3.1. Antenna excitation

We use the well-known delta-gap model [2]. A port is a small gap of width d between certain mesh edges. There is a potential difference in this gap. A port is called active if a voltage U created by an EMF ϵ is applied across it. A port is called passive if a resistance R is connected across it. It is possible to specify both voltage and resistance simultaneously, but such cases are not considered in this work. A port is approximated as a line consisting of one or several consecutive edges. Let Θ denote the set of indices i for which the edge between cells σ_i^1 and σ_i^2 lies on the port line.

The primary field $\mathbf{E}_{inc}$ is defined as

$$\mathbf{E}_{inc} = \frac{U}{d}\boldsymbol{e}_0, \quad (17)$$

where $\boldsymbol{e}_0$ is the unit vector indicating the current direction. The voltage on each edge from the set of port edges Θ satisfies the formula

$$U = \epsilon - R(\boldsymbol{e}_0, \boldsymbol{g}_c), \quad (18)$$

where $\boldsymbol{g}_c$ is the value of the surface current on this edge.

In case of an active port the EMF ϵ is given and the resistance $R = 0$. The scalar product of the primary field and the basis function is the following:

$$(\boldsymbol{v}_i, \mathbf{E}_{inc})_{\tilde\Sigma} = \frac{U}{d}(\boldsymbol{v}_i, \boldsymbol{e}_0) = \frac{U}{d}\int\limits_{D}\frac{(C_i - x, \boldsymbol{e_0})}{s_i}d\sigma = \frac{U}{d}\int\limits_{D}\frac{2}{L}d\sigma = 2U, \quad i \in \Theta, \quad (19)$$

D is the part of the surface Σ where the gap is located. If the i-th edge does not belong to the set Θ, then $(\boldsymbol{v}_i, E_{inc})_{\tilde{\Sigma}} = 0$. That is, the right-hand side is a vector consisting of two values: 0 or $2U$. In case of a passive port $(\boldsymbol{v}_i, E_{inc}) = 0$ for $i \notin \Theta$, as well for the active port.

Now let us consider the passive port edges

$$(\boldsymbol{v}_i, \mathbf{E}_{inc})_{\tilde{\Sigma}} = 2U = -2R(\boldsymbol{e}_0, \boldsymbol{g}_c) = \, = -2R \sum_{k\in\Theta} g_k(\boldsymbol{e}_0, \boldsymbol{v}_i) = -4R \sum_{k\in\Theta} g_k, \quad i \in \Theta. \tag{20}$$

This term is not included into the right-hand side. It is added to the system of equations as a sparse matrix B where $b_{ik} = -4R$ if $i, k \in \Theta$.

1.3.2. Current and impedance

Important antenna characteristics are the port current, input impedance, S-parameters, and standing wave ratio (VSWR). The input impedance of an antenna is the complex ratio of the voltage to the current at its feed point. S-parameters (scattering parameters) describe how electromagnetic power propagates through a multi-port network, quantifying reflection and transmission coefficients. The VSWR is a measure of the impedance mismatch between the antenna and its feed line, defined as the ratio of the maximum to the minimum voltage of the standing wave along the line.

First we consider a single antenna. The current through the port is the flux of the vector g through the line consisting of all port edges

$$Y = (\boldsymbol{e}_0, \boldsymbol{g}_c) = 2 \sum_{k\in\Theta} g_k. \tag{21}$$

We denote Z as the port impedance, S as the S-parameter, η as the VSWR, and R_0 as the matching impedance (a given value). According to [5], the following formulas are used for these characteristics:

$$Z = \frac{U}{Y}, \tag{22}$$

$$S = \frac{Z - R_0}{Z + R_0}, \tag{23}$$

$$\eta = \frac{1 + |S|}{1 - |S|}. \tag{24}$$

Now we consider an antenna system consisting of m active ports. Each port consists of a set of edges Θ_m. In this case, the calculation of the antenna system characteristics is performed according to the following scheme. Each port is sequentially considered active with an EMF $\epsilon = 1$ V, while the others are set as passive with the same resistance R_0. The currents Y_{ij} are computed:

$$Y_{ij} = 2 \sum_{k\in\Theta_i} g_k, \quad \epsilon \neq 0 \text{ on } \Theta_j.$$

As a result, a current matrix Y is obtained. The mutual impedance matrix Z, S-parameter matrix, and VSWR values are calculated using the formulas [5]:

$$Z = Y^{-1}, \tag{25}$$

$$S = \left(\frac{Z}{R_0} + I\right)^{-1} \left(\frac{Z}{R_0} - I\right), \tag{26}$$

$$\eta_i = \frac{1 + |S_{ii}|}{1 - |S_{ii}|}. \tag{27}$$

Here Y, Z and S are complex matrices of size $m \times m$, and η is a real vector of size m.

1.3.3. Optimization of calculations

Now let us consider an antenna system with a number of ports $m > 1$, for which it is required to compute the mutual impedance, S-parameter and VSWR of all ports. In the resulting system of equations, the main part of the matrix is independent of the ports and is assembled according to formula (16). However, when passive ports are present, the addition term (20) appears. If the problem is solved "head-on", the matrix is calculated m times and the system is solved m times. To avoid an increase in the computational complexity of the problem, we used a well-known formula from linear algebra, namely the Woodbury formula, which is a generalization of Sherman–Morrison formula [3].

Let us introduce the notation: $A \in \mathbb{C}^{n\times n}$, $U \in \mathbb{C}^{n\times m}$, $V \in \mathbb{C}^{k\times n}$, and $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix.

$$B = A + UV \ \Rightarrow B^{-1} = A^{-1} - A^{-1}U(I_m + VA^{-1}U)^{-1}VA^{-1}. \tag{28}$$

If the rank of the correction is $m \ll n$, then this formula has an asymptotic complexity of $O(nm^2 + m^3)$, which is insignificant compared to complexity of matrix invertion $O(n^3)$. So we can get all the required antenna characteristics with $O(n^3)$ instead of $O(mn^3)$ in case of sequentially solving the problems.

However for large linear systems, constructing the inverse matrix A^{-1} is impossible. The problem is solved approximately using one or another iterative method. To apply the Woodbury formula to the iterative method for solving the problem described above, we do the following. For each port we form an indicator vector $u_p = \{0, 1\}^n$, $p \in \{1, \ldots, m\}$, where ones are only in those rows that correspond to the edges of this port. If this port is the only passive one, and its input impedance is R_0, then the system matrix is equal to

$$B = A - 4R_0 u_p u_p^T. \tag{29}$$

Since the ports do not intersect, $U = [u_1, \ldots, u_m]$ is a matrix with linearly independent columns. Let us split U into u_p and $\tilde{U}_p = [u_1, \ldots, u_{p-1}, u_{p+1}, \ldots, u_m]$. If port p is the only active one, then the system takes the form

$$(A - 4R_0\, \tilde{U}_p \tilde{U}_p^T)y = -2u_p. \tag{30}$$

The construction of the solution for all ports is carried out according to Algorithm 1. This algorithm is applicable to both direct and iterative methods for solving the system $AY = U$. Thus, instead of solving m systems with different matrices, it is sufficient to solve one system with m right-hand sides and compute the currents using the specified formula.

Algorithm 1 Computation of currents using the Woodbury formula

```
1: Solve $AY = U$
2: $X \leftarrow Y * 2R_0$
3: for $p = 1$ to $m$ do
4:     $x_p, \tilde{X}_p \leftarrow X$
5:     $g_p \leftarrow x_p - \tilde{X}_p \left(I_{m-1} + \tilde{U}_p^T \tilde{X}_p\right)^{-1} \tilde{U}_p^T x_p$
6: end for
7: return $g$
```

2. Results

The calculations were performed using the authors' program "edmpis" [4]. This program allows to solve the problems of electromagnetic wave scattering and antenna radiation not only for perfectly conducting but also for dielectric objects. For small-scale problems (up to 40,000 equations), a direct LU-decomposition method from LAPACK library is employed. For large-scale problems, the matrix is computed using the mosaic-skeleton approximation method [1, 8], and the system of linear algebraic equations is solved using the minimal residual method optimized for multiple right-hand sides [9].

The "edmpis" program uses MPI and OpenMP libraries to accelerate computations. When using the direct LU solver, matrix assembly is accelerated using OpenMP. When using the iterative solver with matrix approximation, MPI is used, and to a lesser extent, OpenMP. The testing of acceleration of parallel program was performed in [8].

It should be noted that the fast multipole algorithm is considered to be the classical method for compression of dense matrices in the electrodynamic problems. We use the mosaic-skeleton approximation because of its universality and good compression. The testing of acceleration of mosaic-skeleton approximation [12] shows almost linear speedup with the increase of MPI processes.

The calculations were performed on the cluster of the Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences. The main cluster specifications are provided below.
Computational nodes "normal":

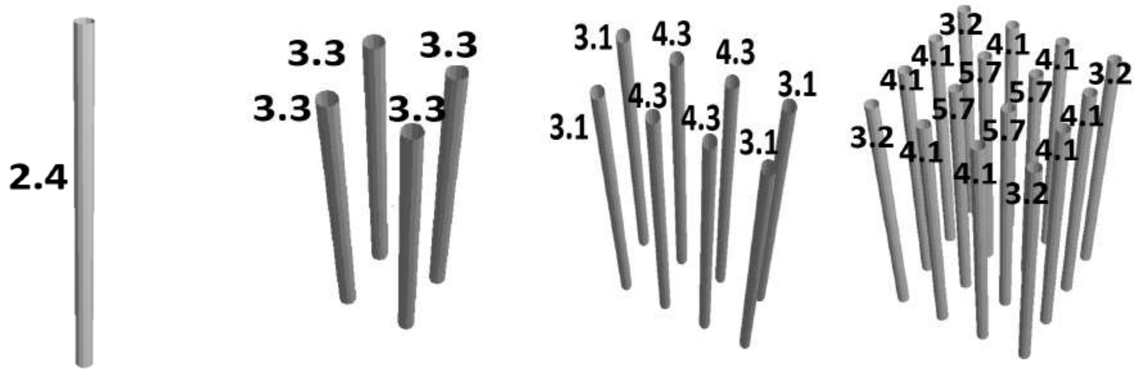
- 40 cores (two 20-core Intel Xeon Gold 6230@2.10GHz processors);
- 384 GB RAM;
- 480 GB SSD storage;
- Operating system: SUSE Linux Enterprise Server 15 SP6 (x86_64).

Computational nodes "short":

- 24 cores (two 12-core Intel Xeon Silver 4214@2.20GHz processors);
- 128 GB RAM;
- 480 GB SSD storage;
- Operating system: SUSE Linux Enterprise High Performance Computing 15 SP4 (x86_64).

2.1. Wire Antenna

As a simple example we consider a thin wire with a radius of 1 cm and a length of 50 cm. Such a wire can be considered as a linear antenna. The calculation was performed on a single "short" type node using 24 cores. In this case the system of linear equations was solved by a direct method.



(a) wire1x1 (b) wire2x2 (c) wire4x2 (d) wire4x4

Figure 1. Antenna arrays composed of wire antennas and their VSWR

Figure 1 shows a single antenna (“strip1x1”) and three antenna systems obtained by copying this antenna: “strip2x2”, “strip4x2”, “strip4x4”. The numbers indicate the calculated VSWR values of the antenna system elements in the case where this element is active and the other elements are passive.

Table 1. Program execution time for wire antennas

Geometry	m	N_{eq}	T_W	T_0
wire1x1	1	1788	5.35	5.31
wire2x2	4	7152	31.03	131.48
wire4x2	8	14304	90.77	730.25
wire4x4	16	28608	317.01	5169.51

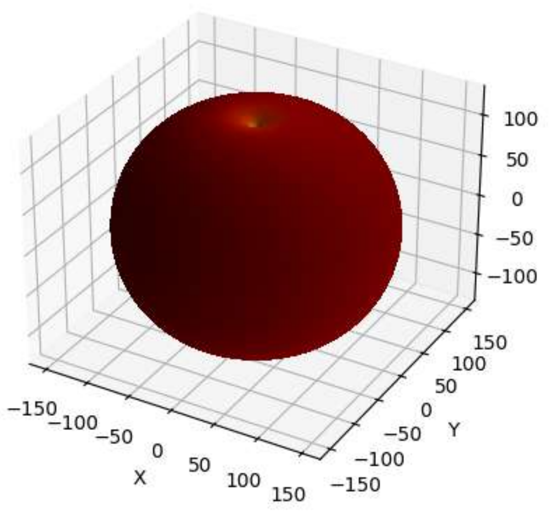
Table 1 shows the program execution time for calculations without optimization (T_0) and with optimization using the Woodbury formula (T_W). Time is measured in seconds. N_{eq} is the number of equations. It can be noted that the optimization sped up the program by a factor of m as expected.

For illustration, Fig. 2 shows the radiation patterns of the antenna systems at a frequency of 0.3 GHz for one of the port loading variants. In all cases, the only active antenna is the one in the lower left corner of Fig. 1. The radiation patterns reflect the dependence of the radiation power asymptote at large distances on the direction to the receiver:

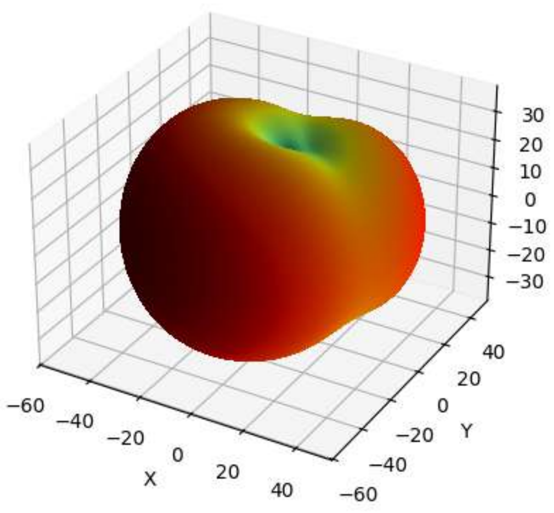
$$F = \lim_{R\to\infty} \frac{4\pi|\boldsymbol{E}(x)|^2}{R^2}, \quad x = R\boldsymbol{\tau}, \tag{31}$$

where $\boldsymbol{\tau}$ is a unit vector indicating the direction to the receiver.

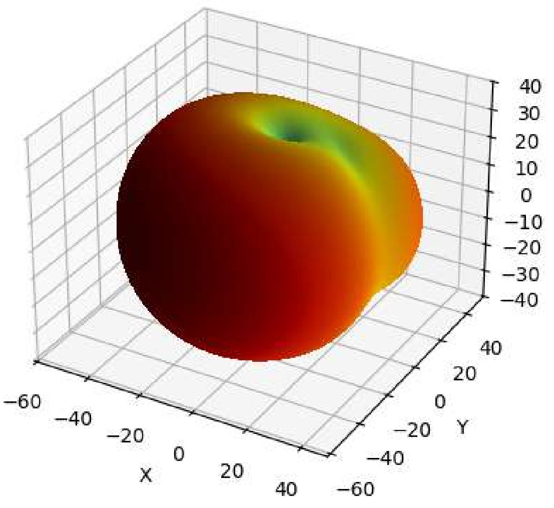
The radiation patterns, VSWR, and other characteristics calculated by the direct method and using the Woodbury formula coincide with a relative accuracy of 10^{-5}. This is due only to rounding errors, since the Woodbury formula allows for an exact, not an approximate, solution.



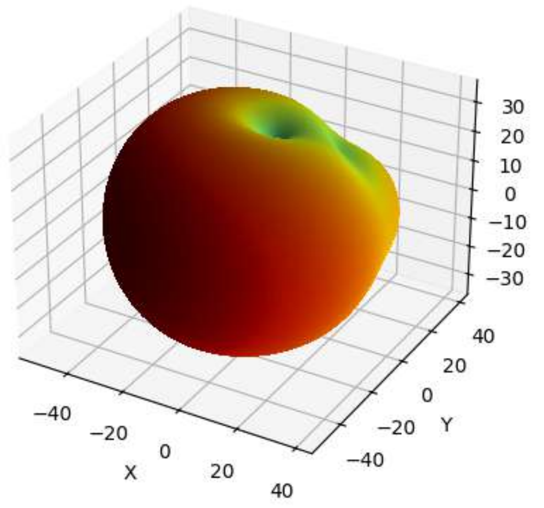
(a) wire1x1



(b) wire2x2



(c) wire4x2



(d) wire4x4

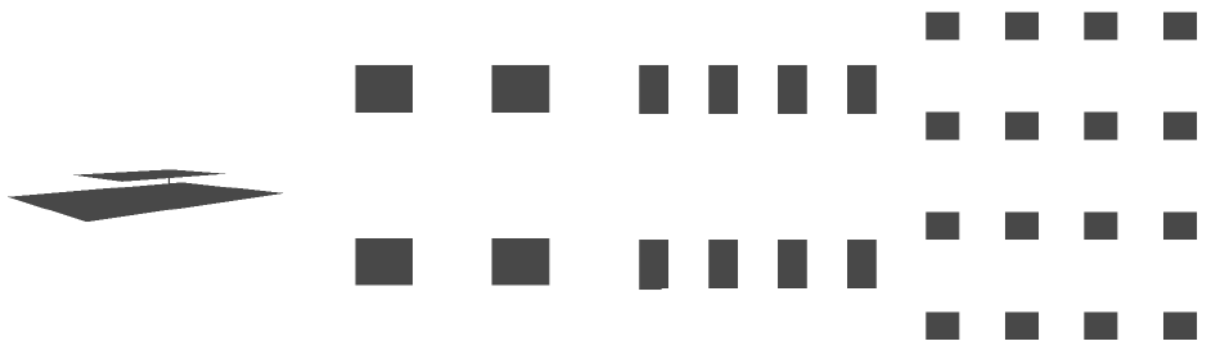
Figure 2. Radiation patterns of antenna arrays composed of wire antennas at a frequency of 0.3 GHz

2.2. Patch Antenna

Another example is a system of patch antennas. Each patch antenna consists of two plates. The excitation of the antenna is defined by a narrow strip connecting the antenna patches, where a lumped port is placed.

The calculation was performed on a single "normal" type node using 40 cores. The problem was solved by an iterative method with an accuracy of 10^{-2}, which influenced the difference between the solution with optimization and without it.

Figure 3 shows a single patch antenna and three antenna arrays obtained by copying this antenna: "patch2x2", "patch4x2", "patch4x4". Unlike the previous example, the VSWR of the patch antennas is mostly independent of the array configuration and the position of the specific antenna. The VSWR of the active antenna is approximately 1.7 for all patch antennas in these



(a) patch1x1 (b) patch2x2 (c) patch4x2 (d) patch4x4

Figure 3. Antenna arrays composed of patch antennas

systems. The mutual influence of patch antennas on each other is significantly less than that of strip antennas. For the mutual impedances the relation $\frac{Z_{ij}}{Z_{ii}} \simeq 10^{-2}$, $i \neq j$ is typical.

Figure 4 shows the radiation patterns of the antenna arrays at a frequency of 4.25 GHz for one of the loading variants. In all cases the only active antenna is the one in the lower left corner. It can be observed how the increase in the number of antennas in the system increases the noise level of the radiation pattern.

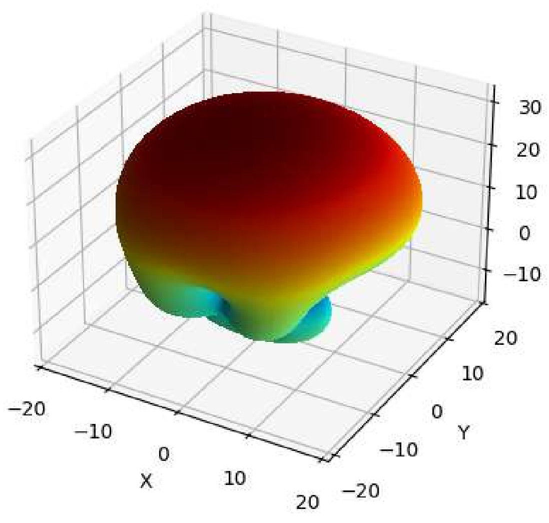
Table 2. Program execution time for patch antennas

Geometry	m	N_{eq}	T_W	T_0	I_W	I_0
patch1x1	1	8879	38.18	38.16	323	323
patch2x2	4	35516	181.32	610.9	1589	2592
patch4x2	8	71032	506.92	2530.08	3285	6306
patch4x4	16	142064	1829.72	9987.38	7071	14380

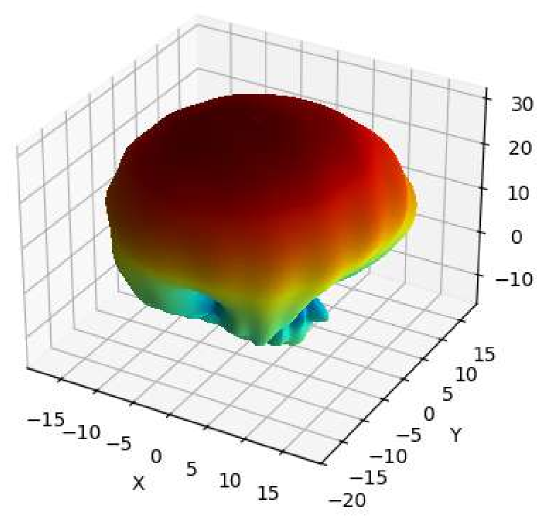
Table 2 shows the program execution time for calculations without optimization and with optimization using the Woodbury formula. I_W is the number of GMRES iterations in the optimized program, I_0 is the total number of GMRES iterations when solving m systems in the program without optimization. The radiation patterns, VSWR, and other characteristics coincide with the accuracy of 10^{-2}, which corresponds to the accuracy of the iterative method.

Conclusion

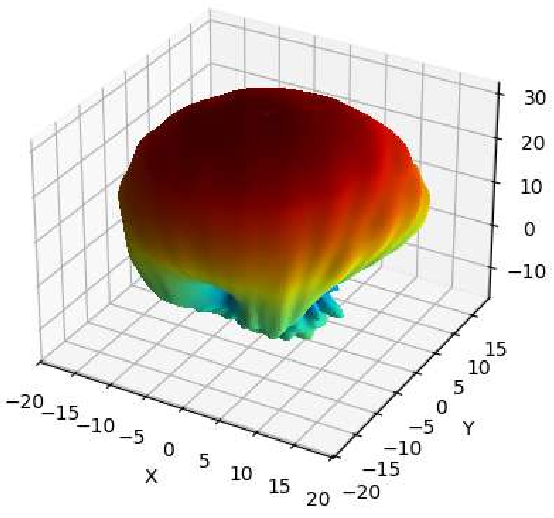
The purpose of this paper was to accelerate the calculation of characteristics of mutual coupling of elements in antenna arrays by the optimization with the Woodbury formula. The numerical experiment demonstrated that in case of direct method the execution time of the optimized program differs from the execution time of the program without optimization by approximately a factor of m, where m is the number of antennas in the system (calculation of the linear antenna system). In the case of iterative method with a low-rank approximation of the system matrix (example with the patch antenna system), optimization also achieves a significant acceleration of computations, but with a smaller factor. For instance, the computation time for a system of 16 antennas decreased by approximately 5.5 times.



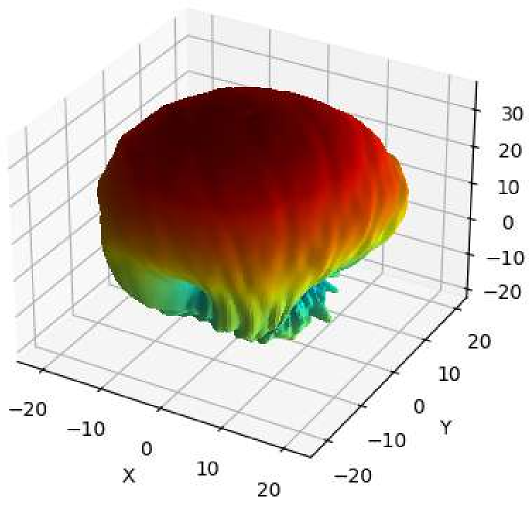
(a) patch1x1



(b) patch2x2



(c) patch4x2



(d) patch4x4

Figure 4. Radiation patterns of antenna arrays composed of patch antennas at a frequency of 4.25 GHz

The difference in the acceleration of computations when applying direct and iterative methods is as follows. In the direct algorithm, an LU decomposition of the matrix is constructed using the LAPACK library, after which the problem is solved for one or m right-hand sides in approximately the same time. For the patch antenna first the matrix is approximated and then the GMRES solver is called. When using optimization, the matrix approximation is performed once instead of m times. Also, the used iterative solver is optimized for problems with multiple right-hand sides: the Krylov basis built for previous right-hand sides is used for the next one. However in this example the total number of iterations increases with the number of ports compared to the case of a single port, but with a smaller factor than the number of ports. When using optimization, the number of iterations decreases by 1.5 to 2 times compared to the total number of iterations without optimization. It is known that the right-hand side vectors u_p,

$p = 1, \ldots, m$, are orthogonal to each other, but this information is insufficient to predict the behavior of the iterative method on an arbitrary antenna array.

To sum up, the developed method sufficiently accelerates the calculation of characteristics of an antenna system with many active excitation elements.

Acknowledgements

The work was supported by the Moscow Center of Fundamental and Applied Mathematics at INM RAS (Agreement with the Ministry of Education and Science of the Russian Federation No. 075-15-2025-347).

References

1. Aparinov, A.A., Setukha, A.V., Stavtsev, S.L.: Low rank methods of approximation in an electromagnetic problem. Lobachevskii Journal of Mathematics 40(11), 1771–1780 (2019). https://doi.org/10.1134/S1995080219110064
2. Gibson, W.: The method of moments in electromagnetics. CRC Press (2021)
3. Hager, W.: Updating the inverse of a matrix. SIAM Review 31(2), 221–230 (1989). https://doi.org/10.1137/1031049
4. Mass, I., Setukha, A., Tretiakova, R.: Combination of methods of volume and surface integral equations in problems of electromagnetic scattering by small thickness structures. Lobachevskii Journal of Mathematics 45(7), 3107–3120 (2024). https://doi.org/10.1134/S1995080224603837
5. Pozar, D.M.: Microwave engineering: theory and techniques. John Wiley & Sons (2021)
6. Rao, S., Wilton, D., Glisson, A.: Electromagnetic scattering by surfaces of arbitrary shape. IEEE Transactions on Antennas and Propagation 30(3), 409–418 (1982). https://doi.org/10.1109/TAP.1982.1142818
7. Saad, Y., Schultz, M.: Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing 7(3), 856–869 (1986). https://doi.org/10.1137/0907058
8. Setukha, A.V., Stavtsev, S.L., Fetisov, S.N., Mukhin, A.N.: Application of mosaic-skeleton approximations of matrices in electromagnetic scattering problems. Computational Mathematics and Mathematical Physics 65(7), 1691–1708 (2025). https://doi.org/10.1134/S0965542525700617
9. Sukmanyuk, S., Zheltkov, D., Valiakhmetov, B.: Generalized minimal residual method for systems with multiple right-hand sides. arXiv preprint arXiv:2408.05513 (2024)
10. Tyrtyshnikov, E.: Incomplete cross approximation in the mosaic-skeleton method. Computing 64, 367–380 (2000). https://doi.org/10.1007/s006070070031

11. Tyrtyshnikov, E.E.: A brief introduction to numerical analysis. Springer Science & Business Media (1997)

12. Valiakhmetov, B., Tyrtyshnikov, E.: MSk-the package for a dense matrix approximation in the mosaic-skeleton format. In: Russian Supercomputing Days: Proceedings of the International Conference, Moscow, Russia. pp. 20–27 (2023). `https://doi.org/10.29003/m3478.978-5-317-07070-0`

13. Volakis, J.L., Sertel, K.: Integral Equation Methods for Electromagnetic. SciTech, Raleigh, NC (2012)

14. Yla-Oijala, P., Taskinen, M., Sarvas, J.: Surface integral equation method for general composite metallic and dielectric structures with junctions. Progress In Electromagnetics Research 52, 81–108 (2005). `https://doi.org/10.2528/PIER04071301`

15. Zakharov, E.V., Ryzhakov, G.V., Setukha, A.V.: Numerical solution of 3D problems of electromagnetic wave diffraction on a system of ideally conducting surfaces by the method of hypersingular integral equations. Differential Equations 50(9), 1240–1251 (2014). `https://doi.org/10.1134/S0012266114090110`

DOI: 10.14529/jsfi250405

GPU-based Large-eddy Simulation of Mixed-phased Clouds

Evgeny V. Mortikov[1,2], *Elizaveta M. Gashchuk*[1,2], *Andrey V. Debolskiy*[1,2,3]

We discuss the development of the large-eddy simulation (LES) model of the atmospheric boundary layer with embedded two-moment bulk cloud microphysics scheme well-suited for massively-parallel heterogeneous supercomputers based on GPU (Graphics Processing Units) architecture. To evaluate the LES model and its computational efficiency, we consider the numerical setup corresponding to the development of an intense Arctic cold-air outbreak case. It is shown that the dynamic closure approach for calculation of subgrid scale fluxes, applied to both heat and moisture transport, allows to correctly reproduce moist convective boundary layers with mixed-phased clouds even with coarse grid resolution. Implementation of state-of-the-art microphysics scheme for GPU systems not only led to significant speedup of the computations, but in general improved the multi-GPU scaling of the model.

Keywords: large-eddy simulation, boundary layer turbulence, cloud-resolving modeling, computational performance, graphics processing unit.

Introduction

The atmospheric processes are characterized by a large spectra of motions. With a ratio of largest-to-smallest scales approaching ten orders of magnitude [46] they range from planetary waves (10^8 m) and cyclones to convective and cloud processes and finally to microscale turbulence (10^{-2} m). The rise in supercomputer performance in general coincides with grid refinement in state-of-the-art large-scale numerical weather prediction (NWP) and climate models, allowing to resolve increasingly larger span of scales and, in turn, to improve model predictions.

While cloud-resolving simulations are still (and will be in coming decades) a complex computational challenge for global atmospheric models [39], where convective processes are parameterized using RANS (Reynolds-Averaged Navier–Stokes) closures [34], the large-eddy simulation approach allows to explicitly reproduce the largest and most energy-containing scales contributing to both turbulence mixing and cloudy convection owing to the self-similarity of turbulence in the inertial range. This allows to study inherently linked turbulence, cloud-scale circulations and microphysical processes.

Unfortunately, the LES approach with available HPC resources in most cases is limited to diurnal studies and spatial scales reaching at best the order of 10 km. Current developments, see review [40], in numerical models and their optimization for supercomputers seem to bring the gap closer between NWP (at least for regional scale models) and LES. However, there is still a long way ahead, with the largest uncertainties related to representation of land surface-atmosphere interaction processes and the lack of turbulence closures suited for the so-called "grey-zone" resolution, where both LES and RANS underlying assumptions may fail.

One of our overall goals is the development of LES models, which are able to reproduce the atmospheric boundary layer (ABL) turbulence even with coarse resolution approaching "grey-zone" restrictions, while allowing to study the interaction of microscale processes and mesoscale organization of cloud clusters [42] with feasible simulations in domains of the order of

[1]Research Computing Center, Lomonosov Moscow State University, Moscow, Russia
[2]Moscow Center of Fundamental and Applied Mathematics, Moscow, Russia
[3]A. M. Obukhov Institute of Atmospheric Physics, Moscow, Russia

10–100 km. The former sets stringent requirements for the subgrid/subfilter closure, while the evident prerequisite for the latter is an efficient implementation of the model on supercomputers. GPU-accelerated HPC systems may provide a significant increase in computational performance of numerical models, e.g., see [3], where 150× speedup compared to CPU (Central Processing Unit)-only code was achieved in large-eddy simulation of pollutant dispersion and [40], where a LES model supplemented with single-moment ice microphysics scheme was implemented using CUDA.

In this paper we discuss the implementation of a large-eddy simulation model of the ABL with embedded bulk two-moment microphysics scheme capable of reproducing complex mixed-phase cloud processes on GPU-based architecture. We assess both the efficiency of GPU implementation and the sensitivity of the LES model to grid resolution in reproducing an intense Arctic cold-air outbreak case. Bulk microphysics schemes have some notable problems due to simplification in representing collection and sedimentation processes, compared with much computationally demanding bin-based or spectral models [26]. Despite this, they are commonly used in both cloud-resolving LES and large-scale NWP and climate models. In particular, we consider the two-moment cloud microphysics scheme [41, 43], which with some modifications is used in DALES [21] and PALM [29] atmospheric boundary layer LES codes, COSMO [1] and ICON [16] NWP models. In this regard, the results obtained in this study may be useful for improving such models as well.

The paper is structured as follows. In Section 1 we give an overview of the LES governing equations and the subgrid turbulence closure. The cloud microphysics scheme for liquid-phase and mixed-phases processes is presented in Section 2. The numerical aspects of the model and its implementation for supercomputers are given in Section 3. In Section 4 we assess the LES model and its efficiency on GPU architecture, followed by summary and conclusions.

1. Governing Equations and Subgrid-scale Model

We consider the dynamics of a stratified atmosphere governed by the Navier–Stokes equations in the Boussinesq approximation, which is described by a coupled system comprising momentum, continuity, heat and moisture transport equations:

$$\frac{\partial \overline{u}_i}{\partial t} + \frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{\partial \tau_{ij}}{\partial x_j} - \frac{1}{\rho_0}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} + \varepsilon_{ij3} f \overline{u}_j + \overline{F}_i, \tag{1}$$

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0, \tag{2}$$

$$\frac{\partial \overline{\Theta}}{\partial t} + \frac{\partial \overline{u}_i \overline{\Theta}}{\partial x_i} = -\frac{\partial h_{\Theta,i}}{\partial x_i} + \chi_\Theta \frac{\partial^2 \overline{\Theta}}{\partial x_i \partial x_i} + \overline{\Phi}_\Theta + \overline{F}_\Theta, \tag{3}$$

$$\frac{\partial \overline{q}}{\partial t} + \frac{\partial \overline{u}_i \overline{q}}{\partial x_i} = -\frac{\partial h_{q,i}}{\partial x_i} + \chi_q \frac{\partial^2 \overline{q}}{\partial x_i \partial x_i} + +\overline{\Phi}_q + \overline{F}_q, \tag{4}$$

where $\overline{\mathbf{u}} = (\overline{u}_1, \overline{u}_2, \overline{u}_3) \equiv (\overline{u}, \overline{v}, \overline{w})$ denotes the wind velocity vector with components aligned along the coordinates $\mathbf{x} = (x_1, x_2, x_3) \equiv (x, y, z)$, respectively, $\overline{p}$ is the pressure and ρ_0 is the reference air density, ν and $\chi_{[\Theta,q]}$ stand for coefficients of molecular kinematic viscosity and diffusivity, t is time. The term $\varepsilon_{ij3} f \overline{u}_j$ accounts for the Coriolis acceleration, where ε_{ijk} is the Levi-Civita symbol (alternating tensor), $f = 2\Omega \sin\phi$ – Coriolis parameter for latitude ϕ and Ω is the angular speed of Earth's rotation. The vector $\overline{F}_i$ corresponds to external forces acting on the flow, $\overline{\Phi}_{[\Theta,q]}$ are tendencies due to cloud microphysical processes and $\overline{F}_{[\Theta,q]}$ are any other sinks or sources of heat and moisture. For stratified atmosphere $\overline{F}_i$ includes the buoyancy force $\overline{\mathbf{F}}_b = \alpha g \overline{\Theta}_v \cdot \mathbf{e}_3$, where

$$\overline{\Theta}_v = \overline{\Theta}_p \left[1 + \left(\frac{R_v}{R_d} - 1 \right) \overline{q}_v - \overline{q}_{\text{liquid}} - \overline{q}_{\text{solid}} \right] \tag{5}$$

is the virtual potential temperature equivalent to potential temperature $\overline{\Theta}_p$ in dry air, $\overline{q}_{[v,\text{liquid},\text{solid}]}$ are the water vapor, liquid water and solid water mixing ratios, respectively, $R_{[d,v]}$ are the specific gas constants for dry air and water vapor, α is the thermal expansion coefficient, g is the acceleration due to gravity and $\mathbf{e}_3$ is the unity vector in the vertical z direction (positive upwards). We consider prognostic equations (3), (4) for conservative (in terms of wet adiabatic processes, e.g., condensation/evaporation) variables – the total water content $\overline{q} = \overline{q}_v + \overline{q}_{\text{liquid}} + \overline{q}_{\text{solid}}$ and the liquid/solid water potential temperature $\overline{\Theta}$:

$$\overline{\Theta} = \overline{\Theta}_p - \frac{L_v}{c_p \Pi} \overline{q}_{\text{liquid}} - \frac{L_s}{c_p \Pi} \overline{q}_{\text{solid}}, \tag{6}$$

where $\Pi = (p_h/p_0)^{R_d/c_p}$ is the Exner function with hydrostatic pressure p_h non-dimensionalized by reference value p_0, c_p is the heat capacity of dry air at constant pressure, $L_{[v,s]}$ are constants attributed to latent heat of vaporization and sublimation, respectively. The liquid $\overline{q}_{\text{liquid}}$ and "solid" (i.e., ice) $\overline{q}_{\text{solid}}$ water content mixing ratios are calculated by the cloud microphysics part of the model, which is described in subsequent section.

The $\overline{(\cdot)}$ in the system of equations (1)–(4) denotes spatial filtering applied in the LES approach, $\overline{a}(\mathbf{x}, t) = F_{\overline{\Delta}} a(\mathbf{x}, t)$, where $\overline{\Delta}$ is the filter width and a is any scalar variable or vector component. The subfilter or subgrid-scale (as the filter width is related to the grid spacing of the discrete system) stress terms τ_{ij} and scalar fluxes $h_{a,i}$ are expressed as:

$$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j, \tag{7}$$

$$h_{a,i} = \overline{u_i a} - \overline{u}_i \overline{a}, \tag{8}$$

where, e.g., $a = [\Theta, q]$. The system of equations (1)–(4) requires turbulence closure and the subfilter terms have to be defined in terms of resolved (filtered) variables.

We use the well-known dynamic Smagorisnky eddy viscosity model [38], which implies that the anisotorpoic part τ^a_{ij} of τ_{ij} tensor is aligned with the strain rate of the resolved scales:

$$\tau^a_{ij} = \tau_{ij} - \frac{1}{3} \delta_{ij} \tau_{kk} \approx \tau^{\text{smag}}_{ij} = -2 K_m \overline{S}_{ij}, \tag{9}$$

where $\overline{S}_{ij}$ denotes the strain rate tensor of the filtered velocity field:

$$\overline{S}_{ij} = \frac{1}{2} \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right). \tag{10}$$

Here the eddy viscosity coefficient K_m is defined as:

$$K_m = \left(C_s \overline{\Delta} \right)^2 \left| \overline{S} \right|, \tag{11}$$

where C_s is the Smagorinsky coefficient and $\left| \overline{S} \right| = \sqrt{2 S_{ij} S_{ij}}$ is the tensor norm.

The subgrid/subfilter-scale fluxes of temperature Θ and total water content q are expressed in the same vein using the eddy diffusivity hypothesis:

$$h_{\Theta,i} = -K_{\Theta,h}\frac{\partial\overline{\Theta}}{\partial x_i} = -K_m Pr_{\rm sgs}^{-1}\frac{\partial\overline{\Theta}}{\partial x_i}, \tag{12}$$

$$h_{q,i} = -K_{q,h}\frac{\partial\overline{q}}{\partial x_i} = -K_m Sc_{\rm sgs}^{-1}\frac{\partial\overline{q}}{\partial x_i}, \tag{13}$$

where $Pr_{\rm sgs}$ and $Sc_{\rm sgs}$ are the subgrid Prandtl and Schmidt numbers, respectively, which we assume may be distinct for heat and moisture fluxes.

The dynamic procedure [27] based on the Germano identity [15] is applied for calculation of the Smagorinsky coefficient, $C_s \equiv C_s(\mathbf{x},t)$, and $Pr_{\rm sgs}(\mathbf{x},t)$, $Sc_{\rm sgs}(\mathbf{x},t)$ dependent on spatial coordinates and time. The Lagrangian averaging along the flow pathlines, as proposed in [5, 30], is used for solving the resulting minimization problem. Assuming that exponentially decaying weights are used in the averaging, the problem reduces to simple relaxation-transport equations, which may be efficiently solved with first-order approximation in time, see [50] for details. The dynamic procedure defines subgrid Prandtl $Pr_{\rm sgs}(\mathbf{x},t)$ and Schmidt numbers $Sc_{\rm sgs}(\mathbf{x},t)$ without additional *ad hoc* assumptions on the flow dynamics, and is known to improve the model performance in simulations of stably stratified atmospheric boundary layers on coarse grids [9, 25].

2. Cloud Microphysics

The cloud microphysics model is based on the two-moment bulk scheme, proposed in [41, 43]. In the liquid-phase only case it assumes separation of droplet spectrum by radii threshold r_{th} in two parts, corresponding to cloud droplets $r < r_{th}$ and rain droplets $r \geq r_{th}$. In the two-moment approach only the first two moments of each part of spectra are predicted. The system of equations (1)–(4) is supplemented with prognostic equations for number concentrations of cloud and rain droplets, $\overline{N}_c$ and $\overline{N}_r$, and mixing ratios of cloud and rain water content, $\overline{q}_c$ and $\overline{q}_r$:

$$\frac{\partial\overline{N}_a}{\partial t} + \frac{\partial\overline{u}_i\overline{N}_a}{\partial x_i} = -\frac{\partial h_{N_a,i}}{\partial x_i} + \chi_{N_a}\frac{\partial^2\overline{N}_a}{\partial x_i\partial x_i} + \overline{\Phi}_{N_a}, \tag{14}$$

$$\frac{\partial\overline{q}_a}{\partial t} + \frac{\partial\overline{u}_i\overline{q}_a}{\partial x_i} = -\frac{\partial h_{q_a,i}}{\partial x_i} + \chi_{q_a}\frac{\partial^2\overline{q}_a}{\partial x_i\partial x_i} + \overline{\Phi}_{q_a}, \tag{15}$$

where $a = [c, r]$ and $\overline{q}_{\rm liquid} = \overline{q}_c + \overline{q}_r$. The terms Φ in the right-hand side stand for different cloud microphysics processes. For liquid part of cloud model these are:

$$\overline{\Phi}_{N_c} = \left.\frac{\partial\overline{N}_c}{\partial t}\right|_{\rm act} + \left.\frac{\partial\overline{N}_c}{\partial t}\right|_{\rm evap} + \left.\frac{\partial\overline{N}_c}{\partial t}\right|_{\rm auto} + \left.\frac{\partial\overline{N}_c}{\partial t}\right|_{\rm accr} + \left.\frac{\partial\overline{N}_c}{\partial t}\right|_{\rm sed}, \tag{16}$$

$$\overline{\Phi}_{N_r} = \left.\frac{\partial\overline{N}_r}{\partial t}\right|_{\rm auto} + \left.\frac{\partial\overline{N}_r}{\partial t}\right|_{\rm slf/brk} + \left.\frac{\partial\overline{N}_r}{\partial t}\right|_{\rm evap} + \left.\frac{\partial\overline{N}_r}{\partial t}\right|_{\rm sed}, \tag{17}$$

$$\overline{\Phi}_{q_c} = \left.\frac{\partial\overline{q}_c}{\partial t}\right|_{\rm cond} + \left.\frac{\partial\overline{q}_c}{\partial t}\right|_{\rm evap} + \left.\frac{\partial\overline{q}_c}{\partial t}\right|_{\rm auto} + \left.\frac{\partial\overline{q}_c}{\partial t}\right|_{\rm accr} + \left.\frac{\partial\overline{q}_c}{\partial t}\right|_{\rm sed}, \tag{18}$$

$$\overline{\Phi}_{q_r} = \left.\frac{\partial\overline{q}_r}{\partial t}\right|_{\rm auto} + \left.\frac{\partial\overline{q}_r}{\partial t}\right|_{\rm accr} + \left.\frac{\partial\overline{q}_r}{\partial t}\right|_{\rm evap} + \left.\frac{\partial\overline{q}_r}{\partial t}\right|_{\rm sed}. \tag{19}$$

The tendencies correspond to activation of cloud droplets (act), represented by Twomey-type parameterizations linking cloud condensation nuclei with prescribed dry aerosol number concentration, condensation (cond) and evaporation (evap) of cloud and rain droplets, autoconversion (auto) of cloud droplets to rain, accretion (accr), merging and splitting of rain droplets (slf/brk), with the latter affecting only the number concentration $\overline{N}_r$. Their formulation is given in [41, 43, 44] and assumes that droplet spectra follows a gamma distribution and that the distribution's slope and shape parameters may be estimated based on predicted bulk characteristics. The sedimentation tendencies (sed) are calculated using a simple upwind scheme.

We use the extension of the two-moment scheme for mixed-phase clouds as described in [41], where additional prognostic equations are considered for both number concentrations and mixing ratios of three hydrometeors: cloud ice, snow and graupel, e.g., $a = [i, s, g]$ in (14) and (15), $\overline{q}_{\text{solid}} = \overline{q}_i + \overline{q}_s + \overline{q}_g$. The microphysics scheme assumes power-law relations for diameter-mass and velocity-mass relations for each category, as well as generalized gamma particle size distributions.

The mixed-phase scheme accounts for primary ice production processes – ice nucleation following [37], homogeneous/heterogeneous freezing [4, 7] of cloud and rain droplets, and considers ice multiplication parameterization by the Hallett-Mossop processes, which occurs due to riming of ice particles. The tendencies in equations for number concentrations and mixing ratios of ice, snow and graupel hydrometeors also include deposition (with ventilation coefficients for spherical particles), riming, aggregation and self-collection of snow, partial conversion of snow and ice crystals to graupel, collection of snow by graupel, sublimation, evaporation, melting and its enhancement due to collisions with liquid droplets in temperature range above freezing. The collection processes between different categories are formulated using a generalized approach suggested by [55], where the use of gamma-distribution for particle size and mass-diameter power-law relations allows to evaluate collection integrals analytically in terms of Gamma-functions in a unified manner.

The sedimentation tendencies of all hydrometeors are non-conservative and included in equations for liquid/solid water potential temperature $\overline{\Theta}$ and total water content $\overline{q}$:

$$\overline{\Phi}_\Theta = -\frac{L_v}{c_p\Pi}\left(\left.\frac{\partial \overline{q}_c}{\partial t}\right|_{\text{sed}} + \left.\frac{\partial \overline{q}_r}{\partial t}\right|_{\text{sed}}\right) - \frac{L_s}{c_p\Pi}\left(\left.\frac{\partial \overline{q}_i}{\partial t}\right|_{\text{sed}} + \left.\frac{\partial \overline{q}_s}{\partial t}\right|_{\text{sed}} + \left.\frac{\partial \overline{q}_g}{\partial t}\right|_{\text{sed}}\right), \tag{20}$$

$$\overline{\Phi}_q = \left.\frac{\partial \overline{q}_c}{\partial t}\right|_{\text{sed}} + \left.\frac{\partial \overline{q}_r}{\partial t}\right|_{\text{sed}} + \left.\frac{\partial \overline{q}_i}{\partial t}\right|_{\text{sed}} + \left.\frac{\partial \overline{q}_s}{\partial t}\right|_{\text{sed}} + \left.\frac{\partial \overline{q}_g}{\partial t}\right|_{\text{sed}}. \tag{21}$$

The time-advancement of the cloud microphysics model supports both the sequential-tendency splitting and parallel splitting approaches [11]. In the former during a single time step all tendencies associated with different processes (e.g., for the liquid-phase part of the model these are the terms appearing in the right-hand side in (16)–(19)) are calculated sequentially taking into account the tendencies evaluated beforehand. In the latter case, which will be used in GPU performance analysis in the subsequent section, all tendencies are calculated using the same state variables as they are given at the beginning of the time step. The subgrid terms $h_{N_a,i}$ and $h_{q_a,i}$ in (14) and (15) are evaluated using the eddy diffusivity of total water content $K_{q,h}$, i.e., the dynamic procedure is applied to conservative scalars $\overline{\Theta}$ and $\overline{q}$ only.

The cloud model supports some further simplifications. In particular, assuming that $\overline{N}_{[c,i]}$ represent fixed quantities or distributions and, moreover, that any supersaturation is removed in cloud or ice water content instantaneously. In this case, $\overline{q}_c$ and $\overline{q}_i$ are diagnostic parameters evaluated using the saturation adjustment procedure by considering excess water content $\overline{q}_v - \overline{q}_{sat}$ ($\overline{q}_v > \overline{q}_{sat}$) above saturation mixing ratio $\overline{q}_{sat}$ over water and ice surfaces. This partition of excessive vapor between cloud droplets and ice crystals depends on absolute temperature of air $\overline{T} \equiv \overline{\Theta}_p \Pi$, see [51] for details.

The implementation of the two-moment bulk microphysics scheme was verified using extensive datasets of LES intercomparison studies, in particular, simulations of trade wind cumulus convection [45], precipitating cumulus-topped boundary layers [52] and mixed-phase stratiform Arctic clouds [36].

3. Numerical Implementation

The large-eddy simulation model with embedded two-moment mixed-phase microphysics bulk scheme was implemented as part of the DNS-, LES- and RANS- unified C/C++ code developed at the Lomonosov Moscow State University and the Marchuk Institute of Numerical Mathematics of the Russian Academy of Science [32, 33, 35]. The unified code is designed for numerical modeling of geophysical turbulent flows, includes an extensive set of LES and RANS subgrid closures [17, 18, 50], and was used in studies of the atmospheric boundary layer [9, 20, 25, 47, 48] and for large-eddy and direct numerical simulation (DNS) of urban canopy [18, 49, 53] and channel flows [2, 24].

The code makes use of hybrid MPI/OpenMP/CUDA approach for CPU and GPU computations, however, any possible optimizations due to OpenMP are omitted from this study. The MPI CPU-only implementation was used in large scale simulations of turbulent flows with the grid size of the order of 10^8 cells, e.g., see [57, 58]. The code supports 1D, 2D or 3D spatial decomposition of the computational grid among MPI-processes with common optimizations for improving scaling on HPC systems. This, in particular, includes options for combining MPI data transfers for a number of arrays (e.g., vector or tensor components) or increasing the width of the grid halo region [10] for reducing MPI communications latency. The latter allows to lower the number of calls to MPI functions but at the cost of additional computational overhead, which may be negligible when the size of the problem on MPI-process is comparatively small. The bulk cloud microphysics model was implemented supporting both CPU and GPU computations with additional optimizations for tracer transport equations proposed in [12, 33] for HPC systems.

The numerical method is based on conservative in momentum and energy second-order finite-difference approximation [31] of governing equations (1)–(4) on rectangular grids with staggered arrangement of nodes. The projection method [6] is applied for the time advancement of momentum equations (1) coupled with incompressibility constraint (2). Explicit third-order Adams–Bashforth scheme is used for the approximation in time of scalar transport equations (3), (4), (14), (15) and for the calculation of non divergence-free intermediate velocity at the first step of the projection method.

Biconjugate gradient stabilized (BiCGstab) iterative method with a V-cycle geometric multigrid preconditioner is used for solving the finite-difference approximation of the Poisson equation. On each grid in the multigrid sequence smoothing iterations are performed by successive upper relaxation method (SOR) for red-black ordering of nodes. The projection onto coarse grid and the prolongation operator onto a fine grid correspond to a bilinear interpolation consistent with the averaging operator used in finite-difference stencils.

In the LES subgrid closures we define the filter width $\overline{\Delta}$ as equal to the geometric mean of grid cell widths, $\overline{\Delta} = (\Delta_x \Delta_y \Delta_z)^{1/3}$. The dynamic procedure requires explicit definition of test filter $F_{\widehat{\Delta}}$, which is chosen the same as the ones used in [19, 50], while the filter width ratio α is calculated according to [28]. The filtering operations incur considerable additional MPI communications and the filters are applied for each dimension of the 3D field sequentially to reduce the number of computations. Due to high computational cost (see analysis in [50, 54]) the dynamic procedure for the evaluation of Smagorinsky constant $C_s(\mathbf{x}, t)$, the subgrid Prandtl $Pr_{\text{sgs}}(\mathbf{x}, t)$ and Schmidt numbers $Sc_{\text{sgs}}(\mathbf{x}, t)$ is applied only every three integration time steps.

4. LES Model Evaluation

4.1. Cold-air Outbreak Case

We consider the numerical setup proposed in the COMBLE (Cold-Air Outbreaks in the Marine Boundary Layer Experiment) model-observation intercomparison project [13, 14, 23, 56], which aims to foster studies of key cloud microphysics and aerosol processes interactions and, in particular, evaluate LES and single-column models (SCM) capabilities in reproducing an intense supercooled cold-air outbreak (CAO) case observed over the Norwegian sea on 13 March 2020. Here the transition from ice surface to open water results in formation of a growing convective boundary layer (CBL) with mixed-phase clouds and stratocumulus to cumulus transition.

The setup emulates transition of air mass in Lagrangian frame of reference (over ~ 1000 km distance) with development of Arctic convective cloud features under strong CAO conditions with spatially variable forcing converted to time-varying surface temperature and time- and height-varying geostrophic wind for a horizontally homogeneous domain. Periodic boundary conditions are set in horizontal directions: $\phi(x = L_x, t) \equiv \phi(x = 0, t)$ and $\phi(y = L_y, t) \equiv \phi(y = 0, t)$, where ϕ is any scalar variable or vector component. The surface layer momentum, sensible and latent heat fluxes are calculated using MOST (Monin–Obukhov Similarity Theory) approximation in each surface grid cell using Businger–Dyer stability functions [9]. The momentum and thermal roughness values are kept fixed, with values close to the ones obtained in simulations with dynamic water surface roughness using Charnock parameterization (see analysis of sensitivity to roughness parameterizations in [23]).

The size of the domain is $L_x \times L_y \times L_z \equiv 25{\times}25{\times}7$ km^3. At the top of the domain free-slip and zero flux boundary conditions are applied to horizontal components of momentum and scalars, respectively. We follow the COMBLE part I configuration, which fixes cloud droplet number concentration and for mixed-phase simulations prescribes diagnostic ice formation with homogeneous freezing of drops also included.

The LES and SCM intercomparison results are available in [23] and on the project website [22] (and include the MSU/INM LES model). In this paper we focus on evaluation of LES subgrid model and the efficiency of GPU-computations. Note that we do not consider any long-wave radiation heating to remove any influence of external packages and libraries (i.e., MSU/INM LES model uses the RRTMG library for long-wave radiation transfer in cloudy atmosphere [8], which depending on temporal and vertical discretization of the radiation scheme may take up to 90% of computational time) on the performance of the LES model with embedded cloud microphysics scheme.

4.2. Grid Resolution Sensitivity

The COMBLE case represents a suitable scenario to assess model performance and sensitivity to grid resolution and subgrid closure, as it includes strong turbulence convection and precipitation events with complex mixed-phase cloud interactions all having noticeable effect on distribution of water content in the boundary layer.

As a reference case we use grid size of $256 \times 256 \times 140$ cells, similar to the one proposed in the intercomparison project, but with uniform grid steps in each direction, i.e., $\Delta x = \Delta y = 100$ m and $\Delta z = 50$ m. Contrary to the COMBLE setup specification, no grid refinement towards the surface is used, and this way we exclude any influence of non-uniform grid steps on the LES filtering. Starting from the reference case we successively coarsen the resolution and study how the LES model reproduces the horizontally averaged and domain-averaged quantities. The horizontal grid step is increased up to $\Delta x = \Delta y = 800$ m, while the vertical grid step is increased up to $\Delta z = 400$ m, i.e., eight-fold compared to the reference computational grid.

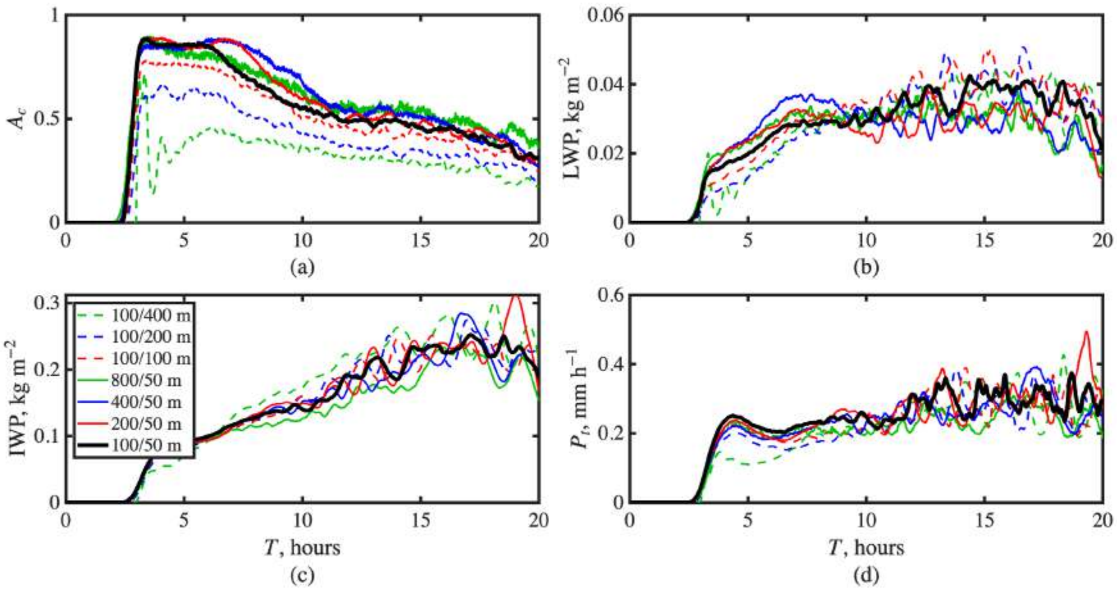


Figure 1. Domain-averaged cloud characteristics: (a) cloud area fraction A_c, (b) liquid water path, (c) ice water path and (d) total precipitation P_t with the decrease in horizontal (solid lines) and vertical (dashed lines) resolution of the large-eddy simulation model. The bold black line denotes the reference simulations on the finest computational grid

Figure 1 shows the domain-averaged characteristics of the key cloud properties: the cloud area fraction (Fig. 1a) A_c, the distribution of total water content (Fig. 1b and Fig. 1c) and the total precipitation rate P_t (Fig. 1d), which is related to ice-phase sedimentation, with negligible contribution from the liquid-phase. The total water content is given in terms of horizontally-averaged values of liquid water path (LWP) and ice water path (IWP):

$$\mathrm{LWP}(x,y) = \int_0^{L_z} \rho_0 q_{\mathrm{liquid}} dz \equiv \int_0^{L_z} \rho_0 \left(q_{\mathrm{c}} + q_{\mathrm{r}}\right) dz,$$

$$\mathrm{IWP}(x,y) = \int_0^{L_z} \rho_0 q_{\mathrm{solid}} dz \equiv \int_0^{L_z} \rho_0 \left(q_{\mathrm{i}} + q_{\mathrm{s}} + q_{\mathrm{g}}\right) dz.$$

Here and hereafter we drop the LES-filtering notation, i.e., $\overline{(\cdot)}$, when denoting the model fields to make the presentation of results more concise.

The decrease in vertical resolution (dashed lines in Fig. 1) has the most effect on cloudy boundary layer characteristics. In particular, it significantly reduces the cloud area fraction, damps the liquid-phase and, to a lesser extent, the ice-phase water content. This in turn affects the precipitation intensity at the early stages of cold-air outbreak, which is mostly due to ice and snow hydrometeors. At the later stages (after 10 hours from the start of simulations, when the cloud top reaches around 2000 m) the boundary layer is sufficiently resolved even with the coarsest grid step of 400 m, and the water content and precipitation estimates almost match the reference case. Notably, the increase of grid steps (solid lines in Fig. 1) in the horizontal directions has a minor influence even on A_c and any other domain-averaged quantities, with only notable excessive condensation during the initial growth of convective boundary layer.

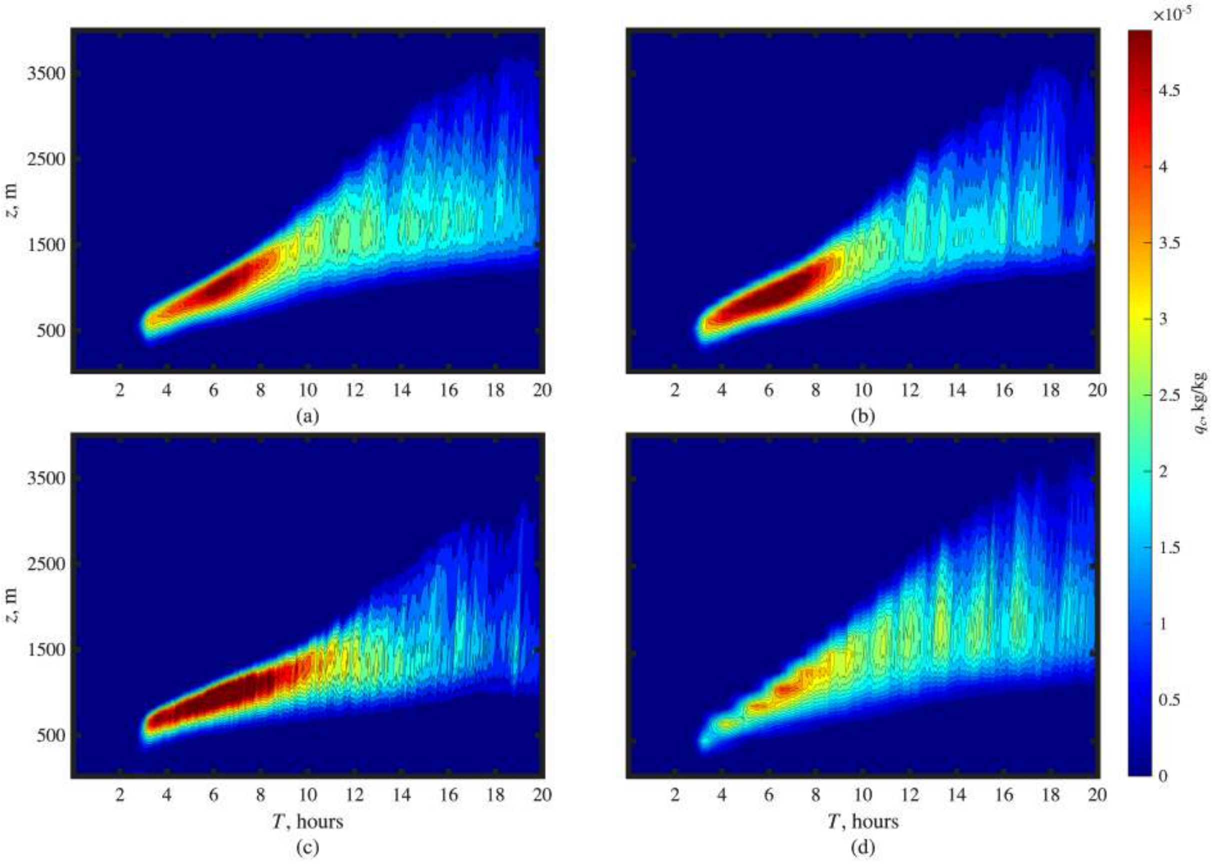


Figure 2. Horizontally-averaged cloud droplet water content, q_c, depending on the grid resolution: (a) reference case with $\Delta_x = \Delta_y = 100$ m, $\Delta z = 50$ m, (b) two-fold decrease in horizontal and vertical resolution, (c) coarsening in horizontal direction to $\Delta x = \Delta y = 800$ m, (d) coarsening in vertical direction to $\Delta z = 200$ m

The horizontally averaged cloud droplet (q_c), ice (q_i) and snow combined with graupel ($q_s + q_g$) water content depending on the resolution of the LES model are shown in Figs. 2, 3 and 4, respectively. The two-fold increase of grid steps in both horizontal and vertical resolution, see Figs. 2–4b, only slightly influences the results. Approaching the coarsest resolution in horizontal directions, $\Delta x = \Delta y = 800$ m (Figs. 2–4c), shows that even though the convective boundary layer growth rate is reduced, resulting in more pronounced saturation and formation of cloud droplets, the ice formation mechanisms and the overall ice-phase water content remains only slightly affected, including the distribution with height of precipitating snow and graupel. When the vertical grid step approaches 200 m (Figs. 2–4d) the entrainment layer at the initial stages is under-resolved suppressing the formation of liquid-phase cloud. This also results in a more thicker distribution of ice water content, in part suppressing conversion to snow and precipitation before and during the transition from stratocumulus to cumulus.

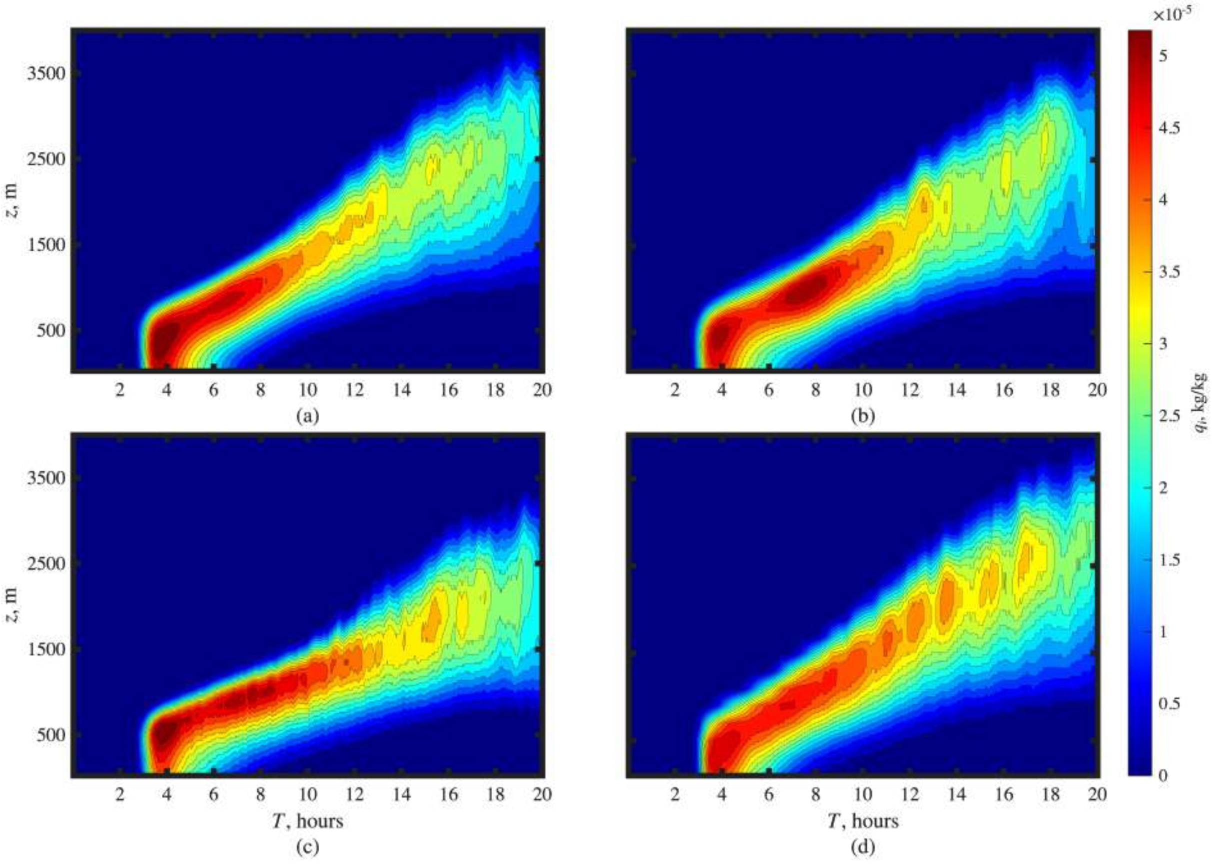


Figure 3. Horizontally-averaged ice water content, q_i, depending on the grid resolution as given in Fig. 2

Our findings show that the developed LES model is capable to reproduce the CAO case characteristics even with highly coarse horizontal resolution of 800 m, while the maximum CBL depth is around 3 km. The horizontal grid resolution only slightly affects the distribution of water content if the turbulence dynamics and the structure of CBL are even barely reproduced. This shows good promise in using the LES model with the dynamic closure approach, applied to momentum, and both heat and moisture transport, to study complex mixed-phase cloud dynamics. On the other hand, as the cloud formation processes are strongly tied with entrainment/detrainment of air, the vertical resolution appears to be more important. While the grid step of 400 m in the vertical direction appears too coarse and unable to resolve the initial growth of boundary layer, the modeling results during the stratocumulus to cumulus transition appear sensitive to even finer resolutions up to 100 m in domain-averaged and horizontally-averaged characteristics, implying also the sensitivity to subgrid closure.

4.3. Computational Performance

To assess the computational performance of the LES model with embedded two-moment bulk microphysics scheme on GPUs, we consider two distinct numerical setups proposed in the COMBLE intercomparison project [23]: the *liquid-only* setup, where only processes related with cloud and rain droplets occur and any ice formation or transport of ice water content is excluded,

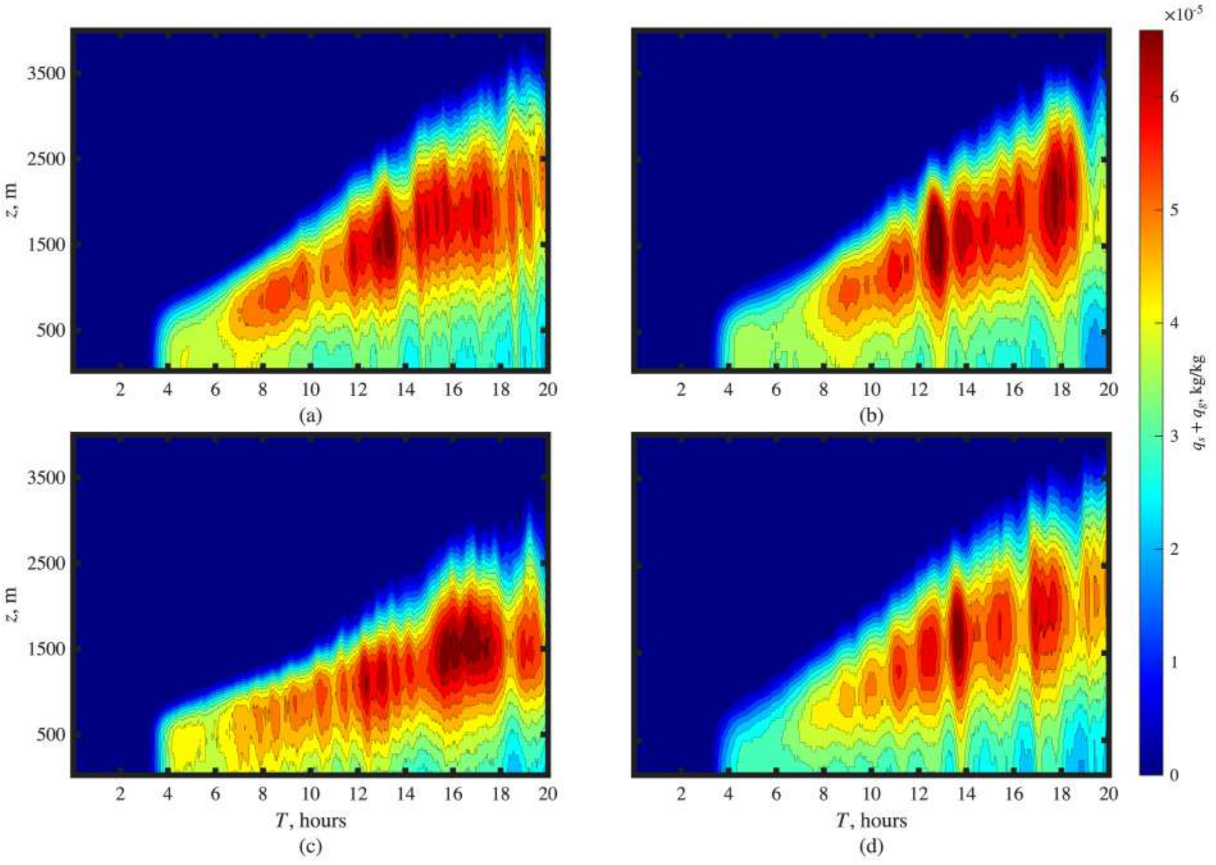


Figure 4. Horizontally-averaged snow and graupel water content, $q_s + q_g$, depending on the grid resolution as given in Fig. 2

and the *mixed-phase* setup with all the liquid- and solid-phase hydrometeors. For the *liquid-only* microphysics we use the saturation adjustment procedure, i.e., the q_c is diagnostic, while the number concentration of cloud droplets, N_c, is fixed. All numerical tests were performed on the Lomonosov-2 supercomputer on nodes with an Intel Xeon Gold 6142 CPUs and NVIDIA V100 GPUs.

To verify the implementation on GPU of the cloud microphysics scheme, the simulations of COMBLE scenario were compared with those obtained on central processing units. Turbulent convective cloud systems are highly nonlinear and we do not expect that the trajectories in both runs would match each other even with minor implementation differences, e.g. order of finite-precision arithmetic operations. The computational grid in these simulations was $128 \times 128 \times 75$ cells, which corresponds to horizontal resolution $\Delta x = \Delta y = 200$ m and a vertical grid step of around $\Delta z \sim 90$ m. Figure 5 shows the difference between the averaged in horizontal directions vertical profiles of ice and snow water content in CPU and GPU runs for the *mixed-phase* case. While the observed differences appear substantial in the later half of simulations, where strong precipitation events occur, we stress that they are within the ensemble spread of LES model for this CAO case. This is also evident in comparison of CPU-GPU implementation results for domain-averaged quantities – liquid and ice water paths shown in Fig. 6, where the relative differences are much smaller.

The two-moment cloud microphysics scheme accounts for circa 50% of total computational time on single CPU core. This is the most computationally demanding part of model time-

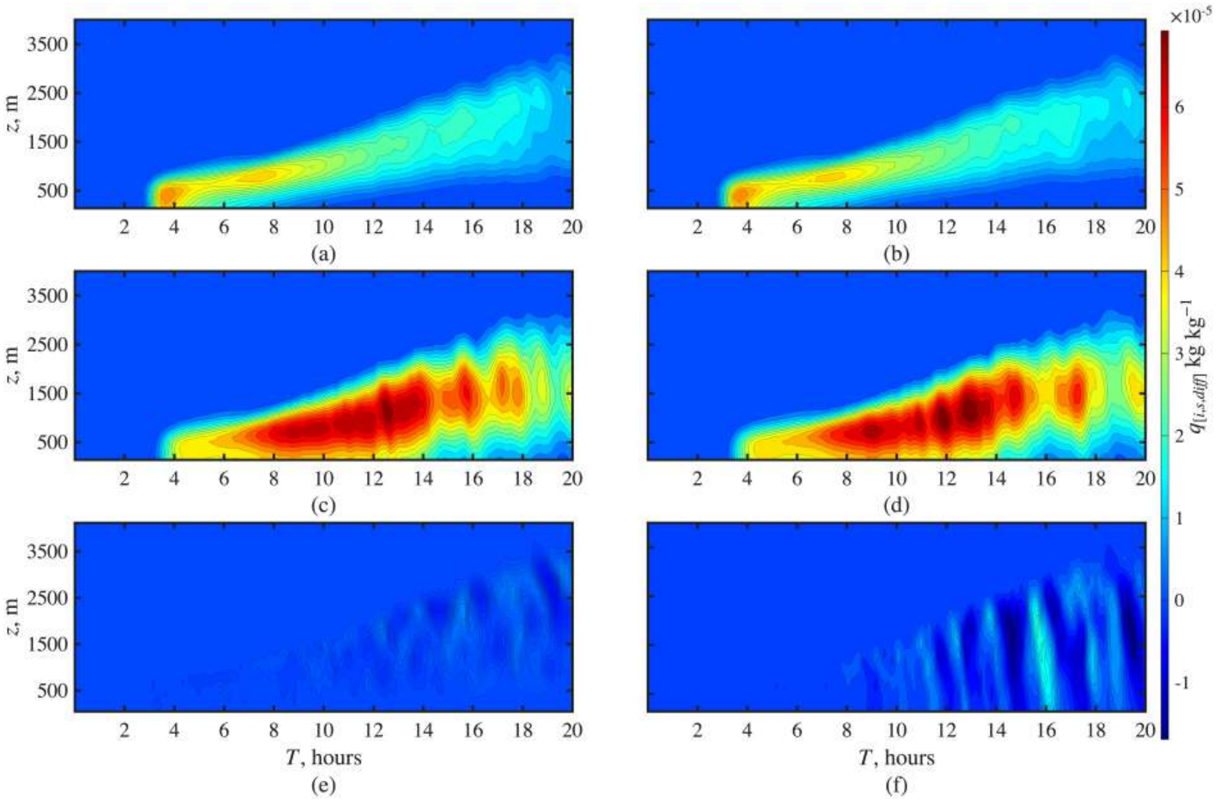


Figure 5. Time evolution of horizontally-averaged vertical profiles for ice, q_i, (a, b) and snow, q_s, (c, d) water content and their respective differences (e, f) between CPU- (a, c) and GPU-based (b, d) implementations in *mixed-phase* case

step, projection method used for solving equations (1) and (2) takes the second place and the dynamic LES closure occupies the third place. This warrants the next part of our study – investigating to what extent the porting of cloud microphysics to GPU architecture could improve the computational performance of the large-eddy simulation model of cloudy boundary layers.

To facilitate these tests, we performed a set of numerical experiments with two-moment cloud microphysics scheme for a range of grid sizes: from $128 \times 128 \times 75$ up to $512 \times 512 \times 75$ cells. Note that for *mixed-phase* case the largest grid size is slightly lower ($576 \times 256 \times 75$ cells) because of increased memory demand – storing additional solid-phase hydrometeor fields and their tendencies makes model reach the 16 GB VRAM limit of V100 devices.

The speedup estimates of running the LES model on GPUs as compared with CPU-only implementation are given in Figs. 7 and 8 for *liquid-only* and *mixed-phase* simulations, respectively. Figure 7 shows the speedup of the entire model (Fig. 7a) and its individual components (Fig. 7b, c and d) for the *liquid-only* case, comparing performance of single GPU with single CPU core. With the increase in grid size (N) the speedup increases for the surface layer flux scheme (Fig. 7b: surface layer), the projection method for solving the momentum equations with incompressibility constraint (Fig. 7c: momentum eq.), and, in particular, the implementation of the BiCGstab algorithm with multigrid preconditioner for solving the finite-difference Poisson equation (Fig. 7c: poisson eq.). The opposite behavior is observed for other components (Fig. 7b, c, d), including the cloud microphysics scheme. This dependence of GPU performance on grid resolution is evident for the *mixed-phase* simulations as well (Fig. 8). In both cases the microphysics scheme (Fig. 7b and Fig. 8b), while being one of the most time-consuming,

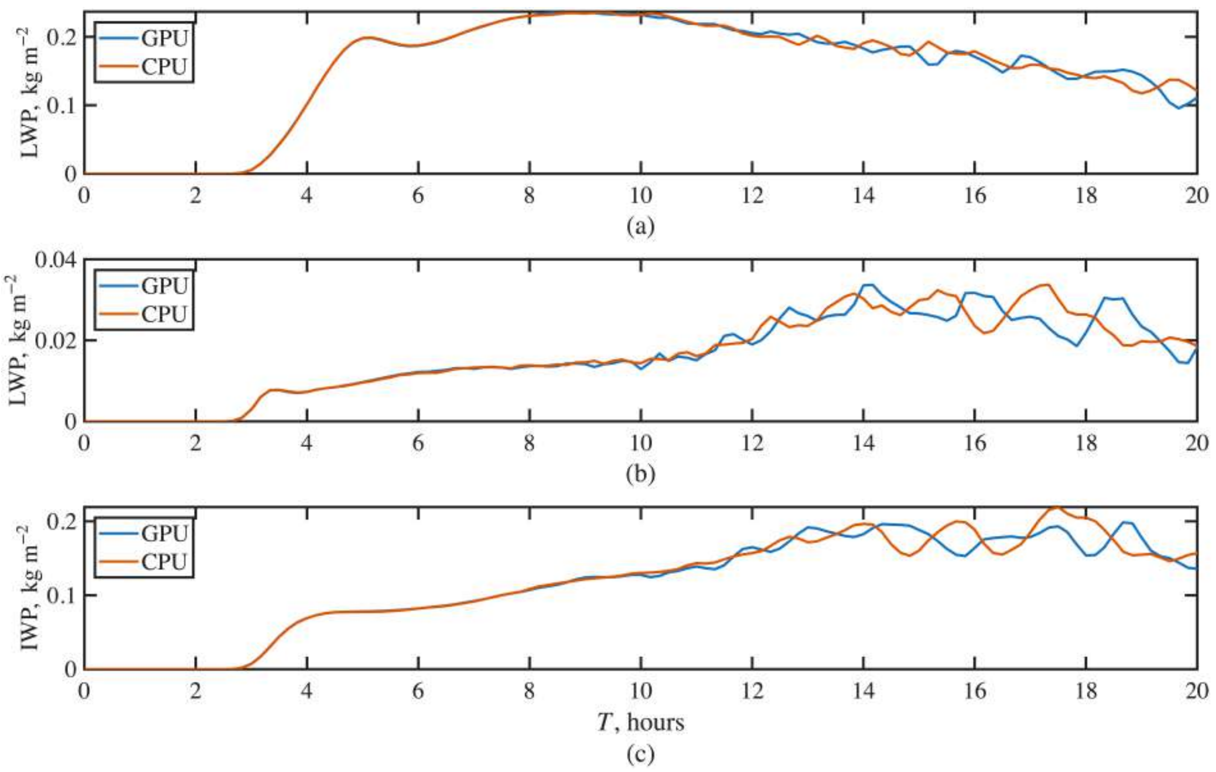


Figure 6. Differences between CPU- and GPU-based implementation of the LES model for liquid water path (LWP) in (a) *liquid-only* and (b) *mixed-phase* cases, (c) ice water path (IWP)

achieves the most significant reduction in run-time over CPU implementation with the speedup in *mixed-phase* well over 200 times compared to single CPU core. Notably, the overall model performance is hindered by less efficient GPU-implementation of explicit in time numerical methods for solving advection-diffusion type equations for liquid/solid water potential temperature Θ, total water content q, mixing ratios q_a and number concentrations N_a of all the hydrometeors, where $a = [c, r, i, s, g]$. Optimization of tracer transport algorithms (see analysis in [12]) seems especially relevant for the *mixed-phase* simulations, as the two-moment microphysics scheme may involve solving ten additional transport equations.

Next, we focus on MPI-scalability of the LES model, shown in Figs. 9 and 10. This evaluation is important for both CPU- and GPU-based implementations. For the former, efficient MPI-scaling is crucial to shorten single GPU-card-to-core gap, and, for the latter, memory constraints on the GPU (compared to CPU) require the use of multiples of devices for very large grids. In numerical tests we enlarged the computational domain in each horizontal direction up to 100 km, resulting in a grid comprising $512 \times 512 \times 75$ cells.

The LES model (Fig. 9a) and almost all of its components (Fig. 9b, c, d, e, f) exhibit a near-linear speedup with increase in the number of MPI processes for CPU-based implementation. The only notable exceptions are the microphysics scheme (Fig. 9b: microphysics) and computations related to evaluation of virtual potential temperature (Fig. 9c: state eq.). This could be attributed to additional MPI communications performed in these parts of the model.

For the MPI-CUDA hybrid implementation we bind each GPU to a separate MPI process, while the speedup is calculated relative to the run time achieved on 2 MPI processes due to insufficient memory available on a single GPU. The results for the GPU-based implementation (Fig. 10) show that MPI-scaling is less efficient, compared with baseline CPU results. In par-

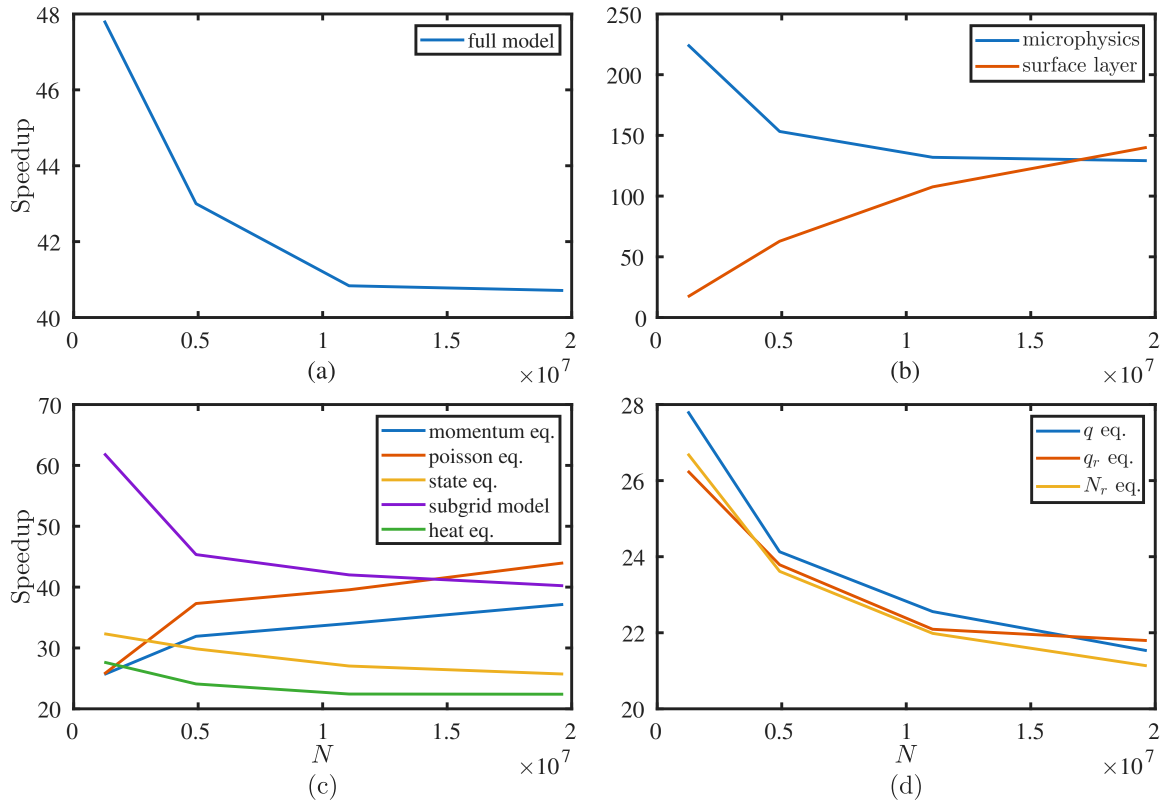


Figure 7. Single GPU over single CPU core implementation speedup of the LES model and its components for *liquid-only* simulations depending on the number of grid cells N

ticular, additional communications with CPU-GPU data transfers thwart the scalability of the dynamics part of the model (Fig. 10c: momentum eq. and poisson eq.). This is primary related to communications and grid coarsening applied in the multigrid method (Fig. 10c: poisson eq.). The reason for poor MPI scalability for the surface flux layer calculations (Fig. 10b: surface layer) appears to be rather small workload, since those calculations are performed only with 2D surface layer data. On the other hand, almost 4-times speedup of the model on 16 GPUs (compared with run time on 2 GPUs) is due to highly-efficient MPI-scaling of the transport equations and, especially, the microphysics scheme.

Some further improvements on MPI-scaling could be expected on novel NVIDIA GPU devices interconnected via high-bandwidth communication links called NVLink. This allows for effective memory exchange between GPUs without utilising CPU RAM (Random Access Memory). For instance, NVLink connection between two A100 GPUs provides 600 GB/s bidirectional bandwidth. Moreover, peer-to-peer exchange between the GPUs on the different computational nodes can be performed if they are connected using RDMA (Remote Direct Memory Access) supported communication links: Infiniband or RoCE – RDMA over Converged Ethernet. Such CPU-excluding data transfers can be implemented in LES models with the communication libraries introduced by coprocessor suppliers (e.g., NVIDIA or AMD). Particularly, NCCL (NVIDIA Collective Communications Library) provides optimized multi-GPU and multi-node communication primitives.

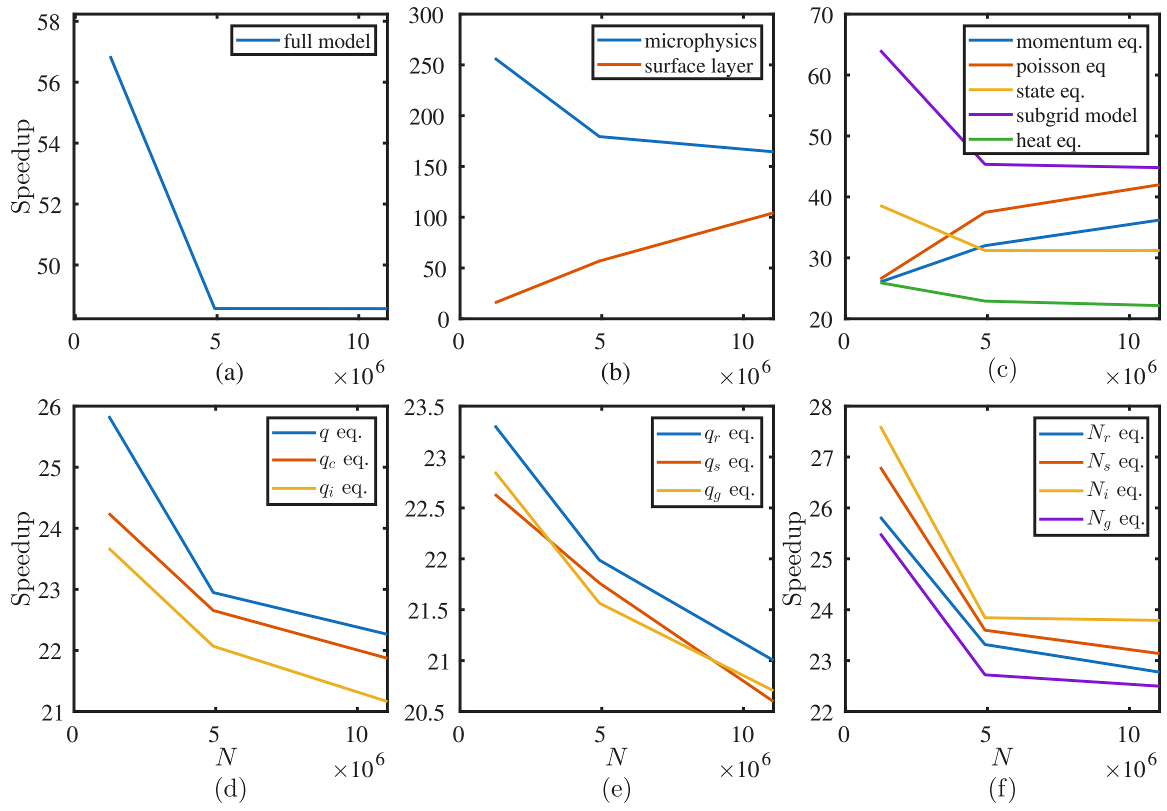


Figure 8. Single GPU over single CPU core implementation speedup of the LES model and its components for *mixed-phase* simulations depending on the number of grid cells N

Conclusions

In this paper we discussed the development of the large-eddy simulation model supplemented with two-moment bulk cloud microphysics scheme for GPU-based HPC systems. The model is based on the dynamic approach for calculation of subgrid scale fluxes, applied to both heat and moisture transport. The analysis of large-eddy simulation of an intense cold-air outbreak case in the Arctic showed little sensitivity in reproducing bulk characteristics of mixed-phase cloudy convection to horizontal grid resolution. On the other hand, the results highlight stronger dependence on vertical resolution even in cases where the CBL dynamics could be expected to be well-resolved by the LES model. Here the grid coarsening led to pronounced reduction in cloud cover, weakened cloud formation during the stratocumuls to cumulus transition, but overall only slightly affected domain-averaged water content and precipitation estimates. In this regard future studies of cloud transition mechanisms during CAO should take into account the possible significant sensitivity to vertical grid resolution and subgrid closure in the LES approach.

The performance evaluation tests demonstrated that GPU implementation provides up to 200 times higher computing performance than single CPU core (or more than 10 times per a CPU node) for computationally demanding cloud microphysics schemes. Numerical solution of tracer transport equations represents one of the main bottlenecks in GPU implementation, relative to other components of the model, achieving a performance comparable to only 1.5 CPU nodes. With the overall estimated threefold speedup of the LES model on graphics processing unit compared to CPU node, adopting code to GPU allows to expand possible range of domain

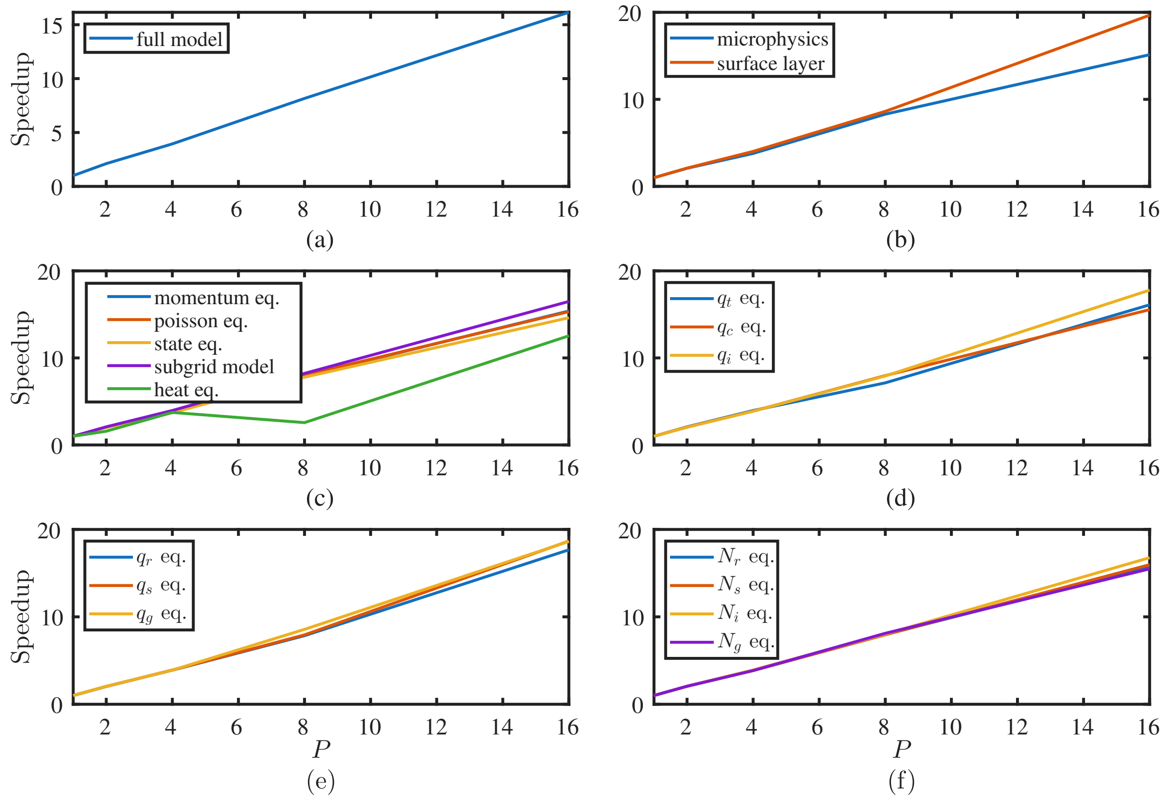


Figure 9. MPI-scalability of the CPU-based implementation of the LES model and its components, P is the number of MPI processes

and grid sizes. This is a prerequisite for studies of complex mixed-phase cloud processes and cloud organizations.

One of the main concerns is related to inefficient MPI multi-GPU scaling of the LES model, which requires further improvement. While the cloud microphysics scheme achieves even super-linear speedup when using 16 GPUs, the overall efficiency of the LES model is hampered by the implementation of multigrid method for solving finite-difference approximation of the Poisson equation, which heavily relies on data coarsening to achieve higher convergence rates and involves a significant amount of communications. These findings suggest the necessity for further optimization of MPI data transfers in GPU-based LES models. In this regard, such improvements could be achieved by excluding CPU RAM in MPI communications, using specialized technologies (e.g., NVLink) and libraries (e.g. NCCL) for device-to-device high bandwidth link between state-of-the-art GPUs.

Acknowledgements

The research is carried out using the equipment of shared research facilities of HPC computing resources at Lomonosov Moscow State University. This study was supported by the Russian Science Foundation (project 25-77-20011; development of the LES model and adoption to multi-GPU computations) and by the Ministry of Education and Science of the Russian Federation (the program of the Moscow Center of Fundamental and Applied Mathematics under agreement No. 075-15-2025-345; implementation evaluation and performance experiments analysis).

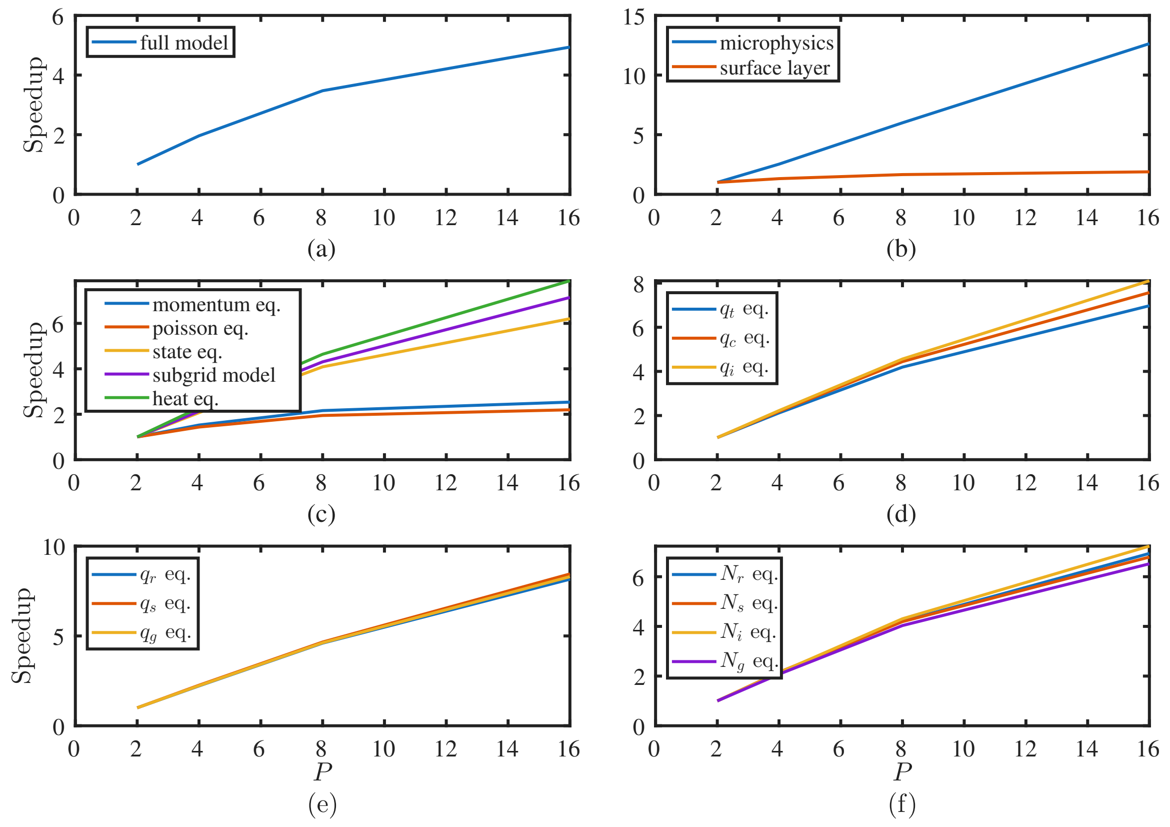


Figure 10. MPI-scalability of the GPU-based implementation of the LES model and its components, P is the number of MPI processes

References

1. Baldauf, M., Seifert, A., Förstner, J., *et al.*: Operational convective-scale numerical weather prediction with the cosmo model: Description and sensitivities. Monthly Weather Review 139, 3887–3905 (12 2011). `https://doi.org/10.1175/MWR-D-10-05013.1`

2. Bhattacharjee, S., Mortikov, E., Debolskiy, A., *et al.*: Direct numerical simulation of a turbulent channel flow with Forchheimer drag. Boundary-Layer Meteorology 185(2), 259–276 (2022). `https://doi.org/10.1007/s10546-022-00731-8`

3. Bieringer, P.E., Piña, A.J., Lorenzetti, D.M., *et al.*: A Graphics Processing Unit (GPU) Approach to Large Eddy Simulation (LES) for Transport and Contaminant Dispersion. Atmosphere 12(7), 890 (2021). `https://doi.org/10.3390/atmos12070890`

4. Bigg, E.K.: The formation of atmospheric ice crystals by the freezing of droplets. Quarterly Journal of the Royal Meteorological Society 79(342), 510–519 (1953). `https://doi.org/10.1002/qj.49707934207`

5. Bou-Zeid, E., Meneveau, C., Parlange, M.: A scale-dependent Lagrangian dynamic model for large eddy simulation of complex turbulent flows. Physics of Fluids 17(2), 025105 (2005). `https://doi.org/10.1063/1.1839152`

6. Brown, D.L., Cortez, R., Minion, M.L.: Accurate projection methods for the incompressible Navier–Stokes equations. Journal of Computational Physics 168(2), 464–499 (2001). `https://doi.org/10.1006/jcph.2001.6715`

7. Cotton, R.J., Field, P.R.: Ice nucleation characteristics of an isolated wave cloud. Quarterly Journal of the Royal Meteorological Society 128(585), 2417–2437 (2002). `https://doi.org/10.1256/qj.01.150`

8. Debolskiy, A.V., Mortikov, E.M., Poliukhov, A.A.: Computational efficiency and adaptation of radiation transfer module within Large-eddy simulation model. Numerical Methods and Programming 26(4), 515–533 (2025). `https://doi.org/10.26089/NumMet.v26r434`

9. Debolskiy, A.V., Mortikov, E.V., Glazunov, A.V., Lüpkes, C.: Evaluation of surface layer stability functions and their extension to first order turbulent closures for weakly and strongly stratified stable boundary layer. Boundary-Layer Meteorology 187(1), 73–93 (2023). `https://doi.org/10.1007/s10546-023-00784-3`

10. Ding, C., He, Y.: A ghost cell expansion method for reducing communications in solving PDE problems. In: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing. p. 50. SC ’01, Association for Computing Machinery, New York, NY, USA (2001). `https://doi.org/10.1145/582034.582084`

11. Donahue, A., Caldwell, P.: Impact of physics parameterization ordering in a global atmosphere model. Journal of Advances in Modeling Earth Systems 10(2), 481–499 (2018). `https://doi.org/10.1002/2017MS001067`

12. Gaschuk, E.M., Ezhkova, A.A., Onoprienko, V.A., *et al.*: Passive tracer transport in ocean modeling: implementation on GPUs, efficiency and optimizations. Lobachevskii Journal of Mathematics 44(8), 3040–3058 (2023). `https://doi.org/10.1134/S1995080223080152`

13. Geerts, B., Giangrande, S., Mcfarquhar, G., *et al.*: “COMBLE” Beginnings: The Cold-Air Outbreaks in the Marine Boundary Layer Experiment. Bulletin of the American Meteorological Society 104, 327–330 (03 2025). `https://doi.org/10.1175/BAMS-D-21-0044.A`

14. Geerts, B., Giangrande, S.E., McFarquhar, G.M., *et al.*: The COMBLE Campaign: A study of marine boundary layer clouds in Arctic cold-air outbreaks. Bulletin of the American Meteorological Society 103(5), E1371–E1389 (2022). `https://doi.org/10.1175/BAMS-D-21-0044.1`

15. Germano, M., Piomelli, U., Moin, P., Cabot, W.H.: A dynamic subgrid-scale eddy viscosity model. Physics of Fluids A: Fluid Dynamics 3(7), 1760–1765 (1991). `https://doi.org/10.1063/1.857955`

16. Giorgetta, M.A., Brokopf, R., Crueger, T., *et al.*: ICON-A, the Atmosphere Component of the ICON Earth System Model: I. Model Description. Journal of Advances in Modeling Earth Systems 10(7), 1613–1637 (2018). `https://doi.org/10.1029/2017MS001242`

17. Gladskikh, D., Ostrovsky, L., Troitskaya, Y., *et al.*: Turbulent transport in a stratified shear flow. Journal of Marine Science and Engineering 11(1), 136 (2023). https://doi.org/10.3390/jmse11010136

18. Glazunov, A., Mortikov, E., Debolskiy, A., Pashkin, A.: Large eddy simulation in the urban environment with simplified and realistic surface morphology. Russian Meteorology and Hydrology 50, 491–506 (2025). https://doi.org/10.3103/S1068373925060056

19. Glazunov, A., Rannik, U., Stepanenko, V., *et al.*: Large-eddy simulation and stochastic modeling of Lagrangian particles for footprint determination in the stable boundary layer. Geoscientific Model Development 9(9), 2925–2949 (2016). https://doi.org/10.5194/gmd-9-2925-2016

20. Glazunov, A.V., Mortikov, E.V., Barskov, K.V., *et al.*: Layered structure of stably stratified turbulent shear flows. Izvestiya, Atmospheric and Oceanic Physics 55(4), 312–323 (2019). https://doi.org/10.1134/S0001433819040042

21. Heus, T., van Heerwaarden, C.C., Jonker, H.J.J., *et al.*: Formulation of the Dutch Atmospheric Large-Eddy Simulation (DALES) and overview of its applications. Geoscientific Model Development 3(2), 415–444 (2010). https://doi.org/10.5194/gmd-3-415-2010

22. Juliano, T., Tornow, F., Fridlind, A.: COMBLE Model-Observation Intercomparison Project Cookbook. https://arm-development.github.io/comble-mip/README.html (2023), accessed: 2025-11-04

23. Juliano, T.W., Tornow, Florian Fridlind, A.M., Ackerman, A.S., *et al.*: The Cold-Air Outbreaks in the Marine Boundary Layer Experiment model-observation intercomparison project (COMBLE-MIP), Part I: Model specification, observational constraints, and preliminary findings. Geoscientific Model Development (2025), (in review)

24. Kadantsev, E., Mortikov, E., Glazunov, A., *et al.*: On dissipation timescales of the basic second-order moments: the effect on the energy and flux budget (EFB) turbulence closure for stably stratified turbulence. Nonlinear Processes in Geophysics 31(3), 395–408 (2024). https://doi.org/10.5194/npg-31-395-2024

25. Kadantsev, E., Mortikov, E., Zilitinkevich, S.: The resistance law for stably stratified atmospheric planetary boundary layers. Quarterly Journal of the Royal Meteorological Society 147(737), 2233–2243 (2021). https://doi.org/10.1002/qj.4019

26. Khain, A.P., Beheng, K.D., Heymsfield, A., *et al.*: Representation of microphysical processes in cloud-resolving models: Spectral (bin) microphysics versus bulk parameterization. Reviews of Geophysics 53(2), 247–322 (2015). https://doi.org/10.1002/2014RG000468

27. Lilly, D.K.: A proposed modification of the Germano subgrid-scale closure method. Physics of Fluids A 4(3), 633–635 (1992). https://doi.org/10.1063/1.858280

28. Lund, T.S.: On the use of discrete filters for large eddy simulation. In: Annual Research Briefs, pp. 83–95. Center for Turbulence Research, Stanford University: Stanford (1997)

29. Maronga, B., Banzhaf, S., Burmeister, C., *et al.*: Overview of the PALM model system 6.0. Geoscientific Model Development 13(3), 1335–1372 (2020). https://doi.org/10.5194/gmd-13-1335-2020

30. Meneveau, C., Lund, T.S., Cabot, W.H.: A Lagrangian dynamic subgrid-scale model of turbulence. Journal of Fluid Mechanics 319, 353–385 (1996). https://doi.org/10.1017/S0022112096007379

31. Morinishi, Y., Lund, T., Vasilyev, O., Moin, P.: Fully conservative higher order finite difference schemes for incompressible flow. Journal of Computational Physics 143(1), 90–124 (1998). https://doi.org/10.1006/jcph.1998.5962

32. Mortikov, E.V.: Numerical simulation of the motion of an ice keel in a stratified flow. Izvestiya, Atmospheric and Oceanic Physics 52(1), 108–115 (2016), https://journal-vniispk.ru/0001-4338/article/view/148416

33. Mortikov, E.V., Debolskiy, A.V.: Direct numerical simulation of stratified turbulent flows and passive tracer transport on HPC systems: Comparison of CPU architectures. Supercomputing Frontiers and Innovations 8(4), 50–68 (2021). https://doi.org/10.14529/jsfi210405

34. Mortikov, E.V., Debolskiy, A.V., Glazunov, A.V., *et al.*: Planetary boundary layer scheme in the INMCM Earth system model. Russian Journal of Numerical Analysis and Mathematical Modelling 39(6), 343–352 (2024). https://doi.org/10.1515/rnam-2024-0029

35. Mortikov, E.V., Glazunov, A.V., Lykosov, V.N.: Numerical study of plane Couette flow: turbulence statistics and the structure of pressure-strain correlations. Russian Journal of Numerical Analysis and Mathematical Modelling 34(2), 119–132 (2019). https://doi.org/10.1515/rnam-2019-0010

36. Ovchinnikov, M., Ackerman, A.S., Avramov, A., *et al.*: Intercomparison of large-eddy simulations of Arctic mixed-phase clouds: Importance of ice size distribution assumptions. Journal of Advances in Modeling Earth Systems 6(1), 223–248 (2014). https://doi.org/10.1002/2013MS000282

37. Reisner, J., Rasmussen, R.M., Bruintjes, R.T.: Explicit forecasting of supercooled liquid water in winter storms using the MM5 mesoscale model. Quarterly Journal of the Royal Meteorological Society 124(548), 1071–1107 (1998). https://doi.org/10.1002/qj.49712454804

38. Sagaut, P.: Large eddy simulation for incompressible flows. Springer (2006). https://doi.org/10.1007/b137536

39. Satoh, M., Stevens, B., Judt, F., *et al.*: Global cloud-resolving models. Current Climate Change Reports 5, 172–184 (2019). https://doi.org/10.1007/s40641-019-00131-0

40. Schalkwijk, J., Jonker, H., Siebesma, A.P., Meijgaard, E.: Weather Forecasting Using GPU-Based Large-Eddy Simulations. Bulletin of the American Meteorological Society 96, 715–723 (2015). https://doi.org/10.1175/BAMS-D-14-00114.1

41. Seifert, A., Beheng, K.D.: A two-moment cloud microphysics parameterization for mixed-phase clouds. Part 2: Maritime vs. continental deep convective storms. Meteorology and Atmospheric Physics 92(1-2), 67–82 (2006). https://doi.org/10.1007/s00703-005-0113-3

42. Seifert, A., Heus, T.: Large-eddy simulation of organized precipitating trade wind cumulus clouds. Atmospheric Chemistry and Physics 13(11), 5631–5645 (2013). https://doi.org/10.5194/acp-13-5631-2013

43. Seifert, A., Beheng, K.D.: A double-moment parameterization for simulating autoconversion, accretion and selfcollection. Atmospheric Research 59-60, 265–281 (2001). `https://doi.org/10.1016/S0169-8095(01)00126-0`

44. Seifert, A., Stevens, B.: Microphysical scaling relations in a kinematic model of isolated shallow cumulus clouds. Journal of the Atmospheric Sciences 67, 1575–1590 (2010). `https://doi.org/10.1175/2009JAS3319.1`

45. Siebesma, A.P., Bretherton, C., Brown, A., *et al.*: A large eddy simulation intercomparison study of shallow cumulus convection. Journal of the Atmospheric Sciences 60, 1201–1219 (2003). `https://doi.org/10.1175/1520-0469(2003)60<1201:ALESIS>2.0.CO;2`

46. Smagorinsky, J.: Global atmospheric modeling and the numerical simulation of climate. In: Hess, W.N. (ed.) Weather and climate modification, pp. 633–686. John Wiley & Sons, Ltd: New York (1974)

47. Suiazova, V., Debolskiy, A., Mortikov, E.: Study of surface layer characteristics in the presence of suspended snow particles using observational data and Large-Eddy Simulation. Izv. Atmos. Ocean. Phys. 60(2), 183–195 (2024). `https://doi.org/10.1134/S000143382470021X`

48. Suiazova, V., Debolskiy, A., Mortikov, E.: Modeling turbulent flows over a heterogeneous surface using mesoscale and large eddy simulations. Russian Meteorology and Hydrology 50, 417–426 (2025). `https://doi.org/10.3103/S106837392505005X`

49. Tarasova, M., Debolskiy, A., Mortikov, E., *et al.*: On the parameterization of the mean wind profile for urban canopy models. Lobachevskii Journal of Mathematics 45, 3198–3210 (2024). `https://doi.org/10.1134/S1995080224603801`

50. Tkachenko, E., Debolskiy, A., Mortikov, E.: Intercomparison of subgrid scale models in large-eddy simulation of sunset atmospheric boundary layer turbulence: Computational aspects. Lobachevskii Journal of Mathematics 42, 1580–1595 (2021). `https://doi.org/10.1134/S1995080221070234`

51. Tomita, H.: New microphysical schemes with five and six categories by diagnostic generation of cloud ice. Journal of the Meteorological Society of Japan. Ser. II 86A, 121–142 (2008). `https://doi.org/10.2151/jmsj.86A.121`

52. vanZanten, M.C., Stevens, B., Nuijens, L., *et al.*: Controls on precipitation and cloudiness in simulations of trade-wind cumulus as observed during RICO. Journal of Advances in Modeling Earth Systems 3(2), M06001 (2011). `https://doi.org/10.1029/2011MS000056`

53. Varentsov, A.I., Mortikov, E.V., Glazunov, A.V., *et al.*: Large-eddy simulation of aerosol transport over different urban local climate zones. Geography, Environment, Sustainability 18(3), 68–79 (2025). `https://doi.org/10.24057/2071-9388-2025-3925`

54. Voevodin, V., Debolskiy, A., Mortikov, E.: Facilitating the Process of Performance Analysis of HPC Applications. Lobachevskii Journal of Mathematics 44, 3178–3190 (2023). `https://doi.org/10.1134/S1995080223080589`

55. Wisner, C., Orville, H.D., Myers, C.: A Numerical Model of a Hail-Bearing Cloud. Journal of the Atmospheric Sciences 29(6), 1160–1181 (1972). `https://doi.org/10.1175/1520-0469(1972)029<1160:ANMOAH>2.0.CO;2`

56. Wu, P., Ovchinnikov, M., Xiao, H., *et al.*: Effect of ice number concentration on the evolution of boundary layer clouds during Arctic marine cold-air outbreaks. Journal of Geophysical Research: Atmospheres 130(3), e2024JD041282 (2025). `https://doi.org/10.1029/2024JD041282`

57. Zasko, G.V., Glazunov, A.V., Mortikov, E.V., *et al.*: Optimal Energy Growth in Stably Stratified Turbulent Couette Flow. Boundary-Layer Meteorology 187(1), 395–421 (2023). `https://doi.org/10.1007/s10546-022-00744-3`

58. Zilitinkevich, S., Druzhinin, O., Glazunov, A., *et al.*: Dissipation rate of turbulent kinetic energy in stably stratified sheared flows. Atmospheric Chemistry and Physics 19(4), 2489–2496 (2019). `https://doi.org/10.5194/acp-19-2489-2019`

DOI: 10.14529/jsfi250406

Variational Data Assimilation in the Constructor of Dynamic Soil Carbon Models

Siumbel K. Shangareeva[1], *Victor M. Stepanenko*[1,2], *George M. Faykin*[1], *Alexander I. Medvedev*[1], *Irina M. Ryzhova*[3], *Vladimir A. Romanenkov*[3]

This work presents an automatic adjoint-model construction within the Carbon Cycle Model Constructor (CCMC) that enables variational data assimilation (VDA) for estimating the initial state of soil dynamic carbon models. The adjoint is generated once from the generic pool-flux representation used in CCMC, which allows efficient gradient evaluation and iterative optimization of the initial pool vector without constructing a model-specific adjoint. The proposed approach is tested with two soil carbon models: SOCS (Soil Organic Carbon Saturation) and RothC (Rothamsted model). Data assimilation experiments are performed using long-term field observations of soil carbon content. The entire VDA workflow, including the adjoint solver and optimization algorithm, is implemented in the same Fortran code base as CCMC. CCMC+VDA implementation is fully compatible with the MPI+OpenMP TerM land surface model and provides a reusable, scalable foundation for variational soil-carbon data assimilation on modern supercomputers.

Keywords: data assimilation, carbon dynamic models, adjoint model, automatic differentiation.

Introduction

Variational data assimilation (VDA) is a class of mathematical methods and numerical techniques used to reduce uncertainty in the external parameters of mathematical models by optimizing performance metrics (objective or cost function) with respect to observed data and prior estimates. VDA relies on the adjoint-equation framework to compute the gradient of the objective (cost) function. Among geophysical models, hydrodynamical models of the atmosphere and the ocean have most benefited from deep integration of VDA into research and operational systems over the last decades [18]. Land surface models have benefited less from VDA, even though they include many parameters and initial states that are not directly measurable. Specifically, this relates to terrestrial carbon-cycle models, where soil carbon pools are rarely measured in situ, and equation parameters are usually phenomenological, suggesting no method of field assessment. For the carbon cycle, VDA can potentially align model parameters and states with a variety of observations (e.g., soil carbon content, CO_2 fluxes, sensible and latent heat fluxes, etc.).

Over the past decade, the practical effectiveness of VDA-based modeling systems has been demonstrated at both regional and global levels [7, 8, 11, 17, 22, 23]. In [11], the authors propose a step-by-step data assimilation system that sequentially optimizes the parameters of the ORCHIDEE model to improve the model's estimation of terrestrial carbon uptake using three data streams: Moderate Resolution Imaging Spectroradiometer (MODIS)-Normalized Difference Vegetation Index (NDVI) satellite observations, net ecosystem exchange (NEE) and latent heat (LE) flux measurements from FLUXNET stations, and atmospheric CO_2 concentrations modeled using the general circulation model (GCM) of the Laboratoire de Météorologie Dy-

[1]Lomonosov Moscow State University, Research Computing Center, Moscow, Russian Federation
[2]Moscow Center of Fundamental and Applied Mathematics, Moscow, Russian Federation
[3]Lomonosov Moscow State University, Soil Science Faculty, Moscow, Russian Federation

namique (LMDz). In [17], the posterior parameter covariance obtained by applying a variational method to the BETHY land surface model is used to quantify forecast errors of CO_2 fluxes and atmospheric concentrations.

In these studies, adjoint model construction relies on source-to-source automatic differentiation (AD) to obtain adjoint code; the most widely used tool is TAF [7, 8, 11, 12, 16]. An alternative is to build the model within the YAO variational-assimilation platform, where the model is described as a modular graph for which an adjoint is generated automatically, as in [2]. These approaches are tightly coupled to specific model implementations. Using TAF requires restructuring and annotating the Fortran code to satisfy the constraints of the AD toolchain, YAO requires rewriting the model as a component graph within its own modeling language. As a result, adjoints must be rebuilt and revalidated for each individual model and model version.

In this work, we develop a module for automatic adjoint construction for models specified in the Carbon Cycle Model Constructor (CCMC) [5] to seek the initial conditions by VDA. The constructor is a general-purpose tool for building carbon-cycle models and is intended for integration into the INM RAS-MSU land active-layer model TerM (Terrestrial Model) [20]. By integrating adjoint generation into CCMC itself, we obtain a reusable, model-agnostic adjoint solver that automatically adapts to any CCMC-defined model without rewriting model code.

The paper is organized as follows. Section 1 briefly reviews the basic VDA framework to be then applied for carbon cycle models. Next, Section 2 presents the concept of carbon cycle model constructor (CCMC), which has been recently proposed and implemented by the authors; introduction of the VDA algorithm to CCMC code is described. Numerical experiments for testing the created CCMC+VDA code are performed with RothC and SOCS model, as detailed in Section 3; the code scalability towards larger scale problems and multicore computer systems and distributed-memory supercomputers is discussed. The Conclusions section summarizes the demonstrated properties of the developed CCMC+VDA code and draws prospects for future research.

1. Variational Data Assimilation

The variational data assimilation problem can be formulated as follows [9, 10, 14, 15]. Let us consider a model described by a system of differential equations:

$$\begin{cases} \frac{\mathrm{d}C(t)}{\mathrm{d}t} = F(C,t), \quad t \in (0,T), \\ C(0) = C_0, \end{cases} \tag{1}$$

where C is the state vector of the model (for carbon-cycle models, C typically represents a vector of carbon pools); F is the models dynamical operator; and C_0 is the unknown initial state to be determined. The optimal initial state C_0^{a} is found as the solution to the following minimization problem:

$$C_0^{\mathrm{a}} = \arg\min_{C_0} J(C_0), \tag{2}$$

$$J(C_0) = \frac{1}{2}\left(C_0 - C_0^b\right)^\top B^{-1}\left(C_0 - C_0^b\right) + \frac{1}{2}\int_0^T \left[HC(t) - y^{\mathrm{obs}}(t)\right]^\top R^{-1}\left[HC(t) - y^{\mathrm{obs}}(t)\right]\,\mathrm{d}t, \tag{3}$$

where C_0^b is the background (prior) initial state, B and R are the covariance matrices of background and observation errors, respectively, y^{obs} denotes the observation vector, and H is the corresponding observation operator, projecting the model state vector to the vector of observations. In the variational assimilation approach, the gradient of the cost function $J(C_0)$ is computed by solving the adjoint problem, yielding the optimality system (1), (4), (5), which can be derived, for example, via the method of Lagrange multipliers:

$$\begin{cases} \frac{\mathrm{d}C^*(t)}{\mathrm{d}t} = -\left[\nabla_C F(C,t)\right]^\top C^*(t) + H^\top R^{-1}\left(HC(t) - y^{\mathrm{obs}}(t)\right), \quad t \in (0,T), \\ C^*(T) = 0. \end{cases} \tag{4}$$

$$\nabla_{C_0} J = B^{-1}(C_0 - C_0^b) - C^*(0) = 0. \tag{5}$$

We then solve, in sequence, the forward model (1) and the adjoint model (4); their outputs are inserted into (5) to compute the cost-function gradient, which drives an update of C_0 using an appropriate gradient-based optimization method. Iterations proceed until the change between successive estimates of C_0 becomes sufficiently small or until a maximal number of iterations is exceeded.

2. Carbon Cycle Model Constructor and Data Assimilation

2.1. CCMC Concept and Implementation

The Carbon Cycle Model Constructor [5] enables the implementation of models representable as a system of differential equations that describe the dynamics of N_p carbon pools:

$$\frac{dC_i}{dt} = F_i = \sum_{j=1}^{N_p} \sum_{k=1}^{N_{i,j}^f} F_{i,j}^k, \quad F_{i,j}^k = \prod_{m=1}^{N_{i,j,k}^m} f_{i,j,k}^m\left(\psi_{i,j,k}^m\right), \quad i = 1, \dots, N_p, \tag{6}$$

where C_i is the carbon content in the i-th pool (a scalar variable); $N_{i,j}^f$ is the number of fluxes between pools i and j; $F_{i,j}^k$ is the k-th flux between pools i and j; $N_{i,j,k}^m$ is the number of multiplicative factors in the expression for the k-th flux between pools i and j; $f_{i,j,k}^m(\cdot)$ is the m-th single-argument function in the expression for the k-th flux between pools i and j; and $\psi_{i,j,k}^m$ is the argument of the m-th function, representing a biotic or abiotic driver of the process, which can be either one of the pools or an external given variable.

The set of possible forms of $f_{i,j,k}^m(\psi_{i,j,k}^m)$ in most carbon cycle models reduces to a number of standard functional dependencies (linear, exponential, Michaelis–Menten, etc.). This allows most models to be implemented within a single code in which the model structure is specified by a collection of standardized multiplicative factors $f = f(\psi)$. Accordingly, the constructor provides an interface for setting the number of pools, the graph of fluxes between pools, and the factors $f_{i,j,k}^m$ so that a solver for the general system (6) implements the model in a such specified configuration.

In order to implement a data assimilation system in CCMC for initial-state estimation for any model specified in CCMC, it is necessary to set the initial guess C_0^b, the error-covariance matrices B and R, and the observation operator H, as well as construct the adjoint model. As can be seen from (4), this requires computing partial derivatives of the models dynamics operator with respect to the carbon pools. The next section describes how this differentiation can be automated for models formulated within CCMC framework, i.e., those that can be specified by (6).

2.2. Automatic Construction of Adjoint Models in CCMC

CCMC solves the forward problem (1) numerically using a first-order explicit time-stepping scheme with a fixed time step Δt. The adjoint problem (4) for the general CCMC equation (6) takes the form:

$$\frac{\mathrm{d}C^*(t)}{\mathrm{d}t} = -\left[\frac{\partial F}{\partial C}(C(t),t)\right]^\top C^*(t) + H^\top R^{-1}\left[HC(t) - y^{\mathrm{obs}}(t)\right], \quad F = (F_1, \dots, F_{N_p}). \tag{7}$$

To construct the adjoint problem (7), it is necessary to evaluate $\frac{\partial F_i}{\partial C_l}$, where $i, l = 1, \dots, N_p$. It can be shown that

$$\frac{\partial F_i}{\partial C_l} = \sum_{j=1}^{N_p} \sum_{k=1}^{N^f_{i,j}} F^k_{i,j} \sum_{m=1}^{N^m_{i,j,k}} \frac{1}{f^m_{i,j,k}\left(\psi^m_{i,j,k}\right)} \frac{\partial f^m_{i,j,k}\left(\psi^m_{i,j,k}\right)}{\partial C_l}, \quad l = 1, \dots, N_p. \tag{8}$$

In the numerical implementation, to avoid division by zero, we replace $\frac{1}{f^m_{i,j,k}(\psi^m_{i,j,k})}$ with $\frac{1}{f^m_{i,j,k}(\psi^m_{i,j,k})+\varepsilon}$, where ε is a small regularization constant. In the current CCMC programming code, eight basic multiplicative factor functions are available: constant, linear, hyperbolic, Michaelis–Menthen, a step function, an exponential, and two piecewise-linear functions, which form the extendable library of functions. By specifying, for each function, the form of its derivative with respect to C_l (using a library of derivatives $df/d\psi$), i.e., $\frac{\partial f^m_{i,j,k}}{\partial C_l}$, one can compute and store $\frac{\partial F_i}{\partial C_l}$ alongside the evaluation of F_i. With $\frac{\partial F_i}{\partial C_l}$ available, the adjoint problem can be integrated in CCMC using the explicit solver used for the forward model.

2.3. Code Modifications to CCMC

To enable automatic adjoint solver construction within CCMC and to solve the data-assimilation system for restoring the initial-state vector, the constructor code was modified as follows:

1. For each base factor function $f^m_{i,j,k}$ implemented in the constructor, their derivatives with respect to C_i were added.
2. A function returning $\frac{\partial F_i}{\partial C_l}$ using formula (8) was added to the method that calculates the right-hand side F_i. The computed derivatives are stored over all time levels, since the adjoint solver requires access to their full temporal history when integrating backward in time.
3. A module was added in which, after solving the forward problem, the adjoint one is solved numerically using an explicit time scheme. The error-covariance matrices B, R, the observation operator H, and the first approximation for the optimized initial state C^b_0 are also specified in this module.
4. A function was added which computes the gradient of the cost function $J(C_0)$ via (5).
5. The iterative adaptive gradient method Adagrad [4] was implemented to update the initial condition C_0 values according to the following formulas:

$$\begin{aligned} g_k &= \nabla_{C_0} J(C_{0,k-1}), \\ G_k &= G_{k-1} + g_k \odot g_k, \\ C_{0,k} &= C_{0,k-1} - \alpha \frac{g_k}{\sqrt{G_k} + \varepsilon}. \end{aligned} \tag{9}$$

Here, $\odot$ denotes the elementwise product of vectors, and the operations of squaring, division, and taking square roots of vectors are all taken elementwise; k is the iteration number, α

is the learning rate, and ε is a small constant for numerical stability. Adagrad was chosen because the scale of the pools can vary greatly, and dynamically recalculating the learning step at each iteration for each pool allows this to be taken into account, thus achieving better convergence [4]. Figure 1 illustrates the schematic flow of CCMC with the embedded adjoint and optimization components.

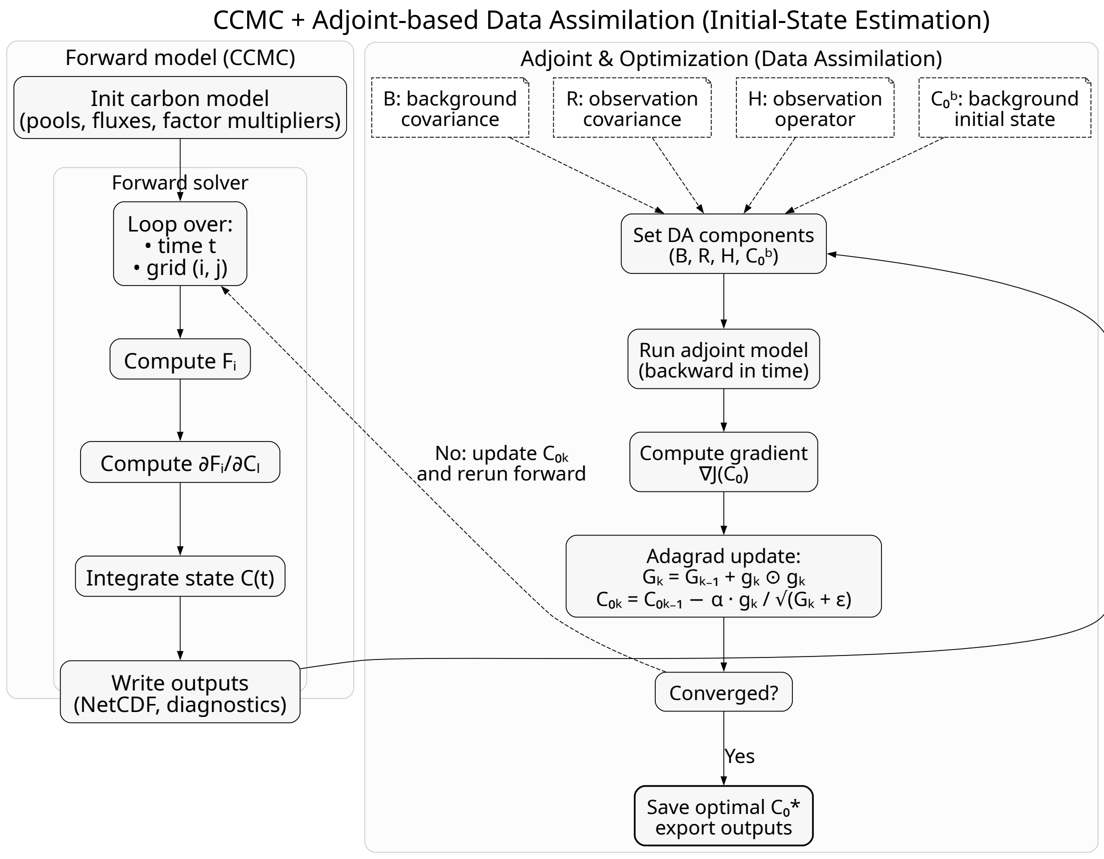

CCMC + Adjoint-based Data Assimilation (Initial-State Estimation)
Forward model (CCMC)
Init carbon model (pools, fluxes, factor multipliers)
Forward solver
Loop over: • time t • grid (i, j)
Compute F_i
Compute $\partial F_i/\partial C_i$
Integrate state C(t)
Write outputs (NetCDF, diagnostics)
Adjoint & Optimization (Data Assimilation)
B: background covariance
R: observation covariance
H: observation operator
C_0^b: background initial state
Set DA components (B, R, H, C_0^b)
Run adjoint model (backward in time)
Compute gradient $\nabla J(C_0)$
Adagrad update: $G_k = G_{k-1} + g_k \odot g_k$ $C_{0k} = C_{0k-1} - \alpha \cdot g_k / \sqrt{(G_k + \varepsilon)}$
Converged?
No: update C_{0k} and rerun forward
misfit $J(C_{0k})$
Yes
Save optimal C_0* export outputs

Figure 1. Flowchart of the variational data assimilation module in the carbon cycle model constructor

3. Numerical Experiments

The data-assimilation system for initial-state estimation within CCMC was tested for the two carbon-soil models implemented in CCMC: SOCS (Soil Organic Carbon Saturation) [13] and RothC (Rothamsted model) [3]. In both cases the optimization is performed in a low-dimensional state space: in a single site (zero-dimensional column) the control vector C_0 has dimension N_p, with $N_p = 2$ for SOCS and $N_p = 4$ for RothC.

3.1. SOCS Model

This model describes the dynamics of two soil-organic-matter (SOM) pools: a free (unprotected) pool and a protected pool formed through organo-mineral interactions and physical occlusion within microaggregates [13]. The model equations read:

$$\begin{aligned} \frac{\mathrm{d}C_1}{\mathrm{d}t} &= I - (1-r)\cdot k \cdot C_1 - rkC_1 \cdot \big(1 - \frac{C_2}{C_m}\big) + k_d C_2, \\ \frac{\mathrm{d}C_2}{\mathrm{d}t} &= r \cdot k \cdot C_1 \cdot (1 - \frac{C_2}{C_m}) - k_d \cdot C_2, \end{aligned} \tag{10}$$

where C_1 is carbon in the free (unprotected) SOM pool; C_2 is carbon in the protected SOM pool; C_m is the maximum amount of organic carbon that can be protected in soil (the soils protective capacity); I is the input of organic carbon to the soil; r is the fraction of carbon transferred to the protected pool during decomposition of C_1; $(1-r)$ are respiration losses; k is the decomposition-rate coefficient for C_1; and k_d is the rate coefficient for the transition of carbon from C_2 to C_1 due to desorption and aggregate breakdown.

3.2. RothC Model

In CCMC, the soil component of the RothC model is implemented, with the prescribed rate of plant litter input to the soil. The evolution of carbon stocks is described by the following differential equations [3]:

$$\begin{aligned} \frac{\mathrm{d}C_{\mathrm{DPM}}}{dt} &= f_{\mathrm{dpm}} \cdot F_{\mathrm{lit}} - R_{\mathrm{DPM}}, \\ \frac{\mathrm{d}C_{\mathrm{RPM}}}{dt} &= (1 - f_{\mathrm{dpm}}) \cdot F_{\mathrm{lit}} - R_{\mathrm{RPM}}, \\ \frac{\mathrm{d}C_{\mathrm{BIO}}}{dt} &= f_{\mathrm{bio}} \cdot \beta_R R_S - R_{\mathrm{BIO}}, \\ \frac{\mathrm{d}C_{\mathrm{HUM}}}{dt} &= f_{\mathrm{hum}} \cdot \beta_R R_S - R_{\mathrm{HUM}}, \end{aligned} \tag{11}$$

where β_R is the clay fraction (particles < 0.002 mm) in the soil; F_{lit} is the input of organic matter to the soil from vegetation, crop residues, and organic fertilizers, R_S is the total respiration rate over the four pools ($R_S = \sum_i R_i$, where $i \in \{\mathrm{DPM}, \mathrm{RPM}, \mathrm{BIO}, \mathrm{HUM}\}$); R_{BIO}, R_{HUM}, R_{RPM}, and R_{DPM} are the respiration rates of the microbial biomass (BIO), long lived humified (HUM), resistant plant material (RPM), and decomposable plant material (DPM) pools, respectively; f_{dpm} is the litter-quality function; and f_{bio} and f_{hum} are partitioning coefficients that allocate incoming organic matter to the BIO and HUM pools during mineralization.

The inert organic matter (IOM) pool is calculated from the total soil carbon at the initial time and kept constant during the simulation:

$$C_{\mathrm{IOM}} = a_{q1} C_{\mathrm{tot}}^{a_{q2}}, \tag{12}$$

where C_{tot} is the total soil carbon content (the sum of all pools), $a_{\mathrm{q1}} = 0.049$, $a_{\mathrm{q2}} = 1.139$ – empirical dimensionless constants.

The terms R_i are computed as

$$R_i = k_{si} F_T(T_{\mathrm{soil}}) F_s(s) F_v(v) C_i, \tag{13}$$

where $i \in \{\mathrm{DPM}, \mathrm{RPM}, \mathrm{BIO}, \mathrm{HUM}\}$, k_{si} is the respiration rate per unit mass of pool i under standard conditions [1/s]; $F_T(T_{\mathrm{soil}})$ is the soil-temperature factor; $F_s(s)$ is the soil-moisture factor; and $F_v(v)$ accounts for vegetation cover. The standard respiration rates k_{si} used in this study are

$$k_{s,\mathrm{DPM}} = 3.22, \qquad k_{s,\mathrm{RPM}} = 9.65, \qquad k_{s,\mathrm{BIO}} = 2.12, \qquad k_{s,\mathrm{HUM}} = 6.43.$$

The IOM pool has no respiration term ($k_{s,\mathrm{IOM}} = 0$).

The soil temperature function reflects temperature variations in the upper active soil layer (0–30 cm) and is defined as

$$F_T(T_{\mathrm{soil}}) = \frac{b_1}{1 + e^{b_2/(T_{\mathrm{soil}} - b_3)}}, \tag{14}$$

where T_{soil} is the mean monthly soil temperature [K]; $b_1 = 47.9$ (dimensionless), $b_2 = 106$ K, and $b_3 = 254.85$ K are empirical parameters.

The soil moisture factor is parameterized as

$$F_s(s) = \begin{cases} 1 - d_1 \cdot (s - s_o), & \text{for } s > s_o, \\ d_2 + d_1 \cdot \left(\frac{s - s_{\min}}{s_o - s_{\min}}\right), & \text{for } s_{\min} < s \leq s_o, \\ d_2, & \text{for } s \leq s_{\min}, \end{cases} \tag{15}$$

where s is the moisture of the upper (unfrozen) soil layer; s_o is the optimal soil moisture at which $F_s(s)$ attains its maximum (i.e., equals one); and $d_1 = 0.8$, $d_2 = 0.2$ are dimensionless empirical coefficients, chosen so that $d_1 + d_2 = 1$, a condition ensuring continuity of F_s. The wilting moisture is determined experimentally:

$$\begin{aligned} s_o &= \tfrac{1}{2} \cdot (1 + s_w), \\ s_{\min} &= c \cdot s_w, \end{aligned} \tag{16}$$

where s_w is the wilting moisture, and $c = 1.7$ is a dimensionless empirical coefficient.

The vegetation-cover factor is given by

$$F_v(v) = e_1 + e_2 \cdot (1 - v), \tag{17}$$

where $v \in [0, 1]$ indicates the presence of vegetation cover. The dimensionless empirical coefficients e_1 and e_2 are 0.6 and 0.4, respectively.

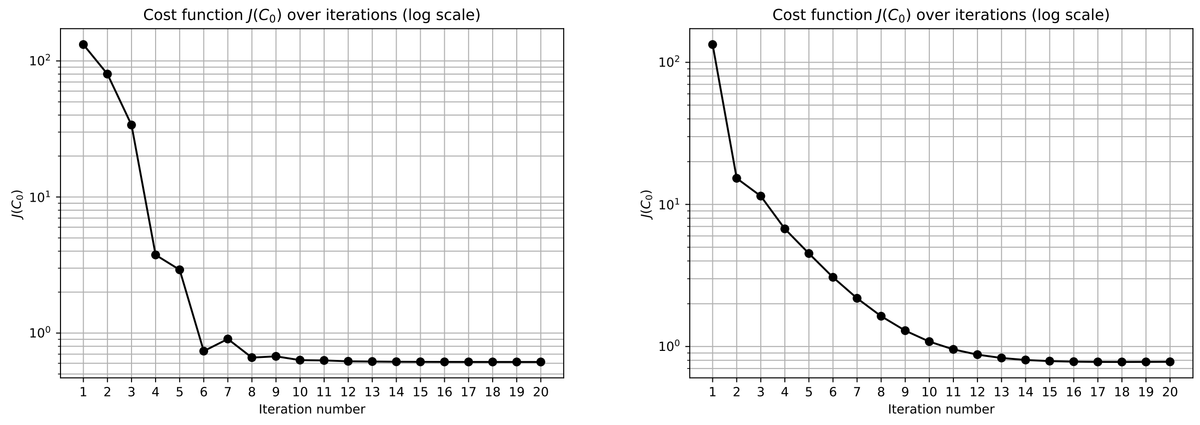
3.3. Results and Discussion

For validation of the variational data assimilation algorithm embedded into CCMC, we used experimental time series of soil carbon content from long-term fertilization field experiments [1] conducted at the Donskoy Federal Agrarian Research Center (Rostov Oblast) and at DAOS-3, the Dolgoprudny Agrochemical Experimental Station (Moscow Oblast). The external forcing time series used in the SOCS and RothC models were compiled from the above-mentioned field stations (soil temperature, soil moisture, organic carbon inputs, mean vegetation cover) together with ERA5 reanalysis data (clay content and wilting-point soil moisture). Both models were integrated with the time step of one month. Their parameter values were set as follows: $r = 0.45, C_m = 12, k_d = 0.007, k = 7.5$ for SOCS and $\beta_R = 0.2, s_w = 0.75, v = 0.8$ for RothC.

The observation operator is $H = (1, 1, \ldots, 1) \in \mathbb{R}^n$, since the measured soil carbon content is the sum of all model pools. The observation-error covariance matrix R is diagonal. Since the observations are of order 10 kg m^{-2}, we assume an absolute measurement uncertainty of 0.1 kg m^{-2}, and therefore set $R^{-1} = 1/0.1$. The background covariance matrix B is also diagonal.

Its diagonal entries are taken as 10% of the characteristic magnitude of each pool, assuming that pool errors are uncorrelated and that no detailed prior covariance information is available. The initial guess for the pool vector C_0 in each model was constructed using expert-based estimates of the typical distribution of soil carbon among pools for the corresponding soil type.

The optimization employs the Adagrad algorithm with learning rate $\alpha = 1$ which was chosen empirically as a compromise between convergence speed and stability. A total of 20 iterations were performed for both models. The convergence of the cost function $J(C_0)$ for the SOCS and RothC models is shown in Fig. 2. Figure 2 demonstrates that, for both SOCS and RothC, the VDA scheme systematically reduces the cost function over the course of the Adagrad iterations and approaches a nearly stationary value by the end of the optimization.



(a) SOCS (b) RothC

Figure 2. Cost function $J(C_0)$ over Adagrad iterations for the SOCS (a) and RothC (b) models

Figures 3 and 4 show the simulated soil carbon stock dynamics obtained with the prior initial conditions and with those estimated via data assimilation for the Rostov Oblast site. Blue dots indicate observations. Tables 1 and 2 compare the prior and optimal initial conditions and report the root-mean-square error (RMSE) of the simulated total soil carbon stocks with respect to the observed time series at the site. In both cases the reference values are the measurements, while the "prior" RMSE is computed from the forward model run started from the background initial state C_0^{b} and the "posterior" RMSE from the run started from the optimized state C_0^{a}. Quantitatively, the VDA-based initialization leads to a strong reduction of the model–data misfit. For SOCS, the RMSE decreases from 1.864 to 0.096 kg m^{-2} and for RothC, the RMSE decreases from 1.88 to 0.152 kg m^{-2}. In Figs. 3 and 4 the prior simulations systematically underestimate the measured soil carbon stocks, whereas the trajectories obtained with VDA-based initial conditions closely track the observed levels and reproduce the temporal evolution of the carbon stocks.

The established way of initializing the pools in carbon cycle models is to run the forward model under statistically stationary external forcing (litter input) for a sufficiently long time period until a quasi-steady-state of the pools is reached [21, 24]. This approach relies on long-term reconstructions of climate, carbon inputs, and land-use history, although such records are typically uncertain. It also presumes that soil carbon is close to equilibrium at the beginning of the simulation, which is often not the case [6, 24]. In our method, the initial pool vector C_0 is treated as an unknown control variable and is estimated directly from the observed soil carbon time series using VDA. This removes the need for uncertain multi-decadal forcing and does not

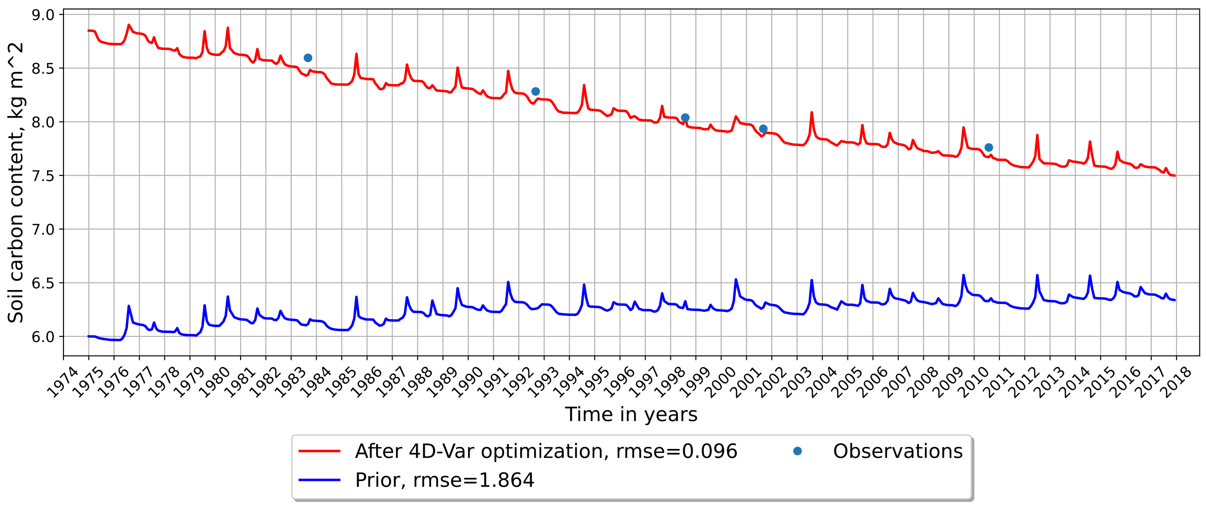


Figure 3. Soil carbon stocks at the Rostov Oblast site simulated by the SOCS model using the prior initial conditions (blue) and initial conditions estimated via data assimilation. Blue dots indicate observations

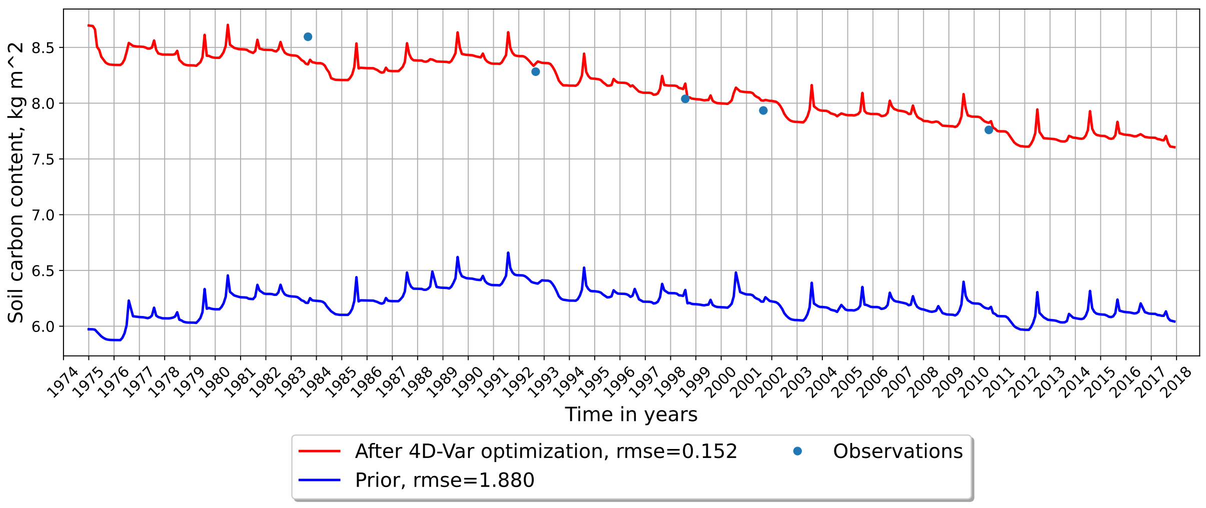


Figure 4. Same as Fig. 3, but simulations are performed with RothC model

Table 1. The prior and optimal initial conditions for SOCS model, and corresponding root-mean-square errors (RMSE)

Pool	Background (prior)	Analysis (posterior)
$C_1(0)$	0.	0.09
$C_2(0)$	6	8.76
RMSE	1.864	0.096

rely on the equilibrium assumption. A limitation of the presented VDA-based initialization is that it requires a time series of soil carbon observations, whereas spin-up can be applied when only a single measurement is available.

The developed CCMC+VDA programming code is intended for implementation into TerM land surface model. The VDA driver, adjoint solver, and Adagrad iterations are implemented in

Table 2. Same as Tab. 1, but simulations are performed with RothC model

Pool	Background (prior)	Analysis (posterior)
$C_{DPM}(0)$	0.01	0.2
$C_{RPM}(0)$	0.1	0.32
$C_{BIO}(0)$	0.05	0.24
$C_{HUM}(0)$	5.	7.13
RMSE	1.88	0.152

the same Fortran code base as CCMC and TerM. The parallel implementation of TerM uses MPI and OpenMP. The TerM computational domain is a rectangular latitude-longitude grid that is decomposed across MPI processes with nested OpenMP threads; each process/node computes all grid points in its subdomain independently of the others [19]. This decomposition is valid because TerM solves a system of one-dimensional vertical equations in the longitude-latitude cells with no horizontal coupling. CCMC+VDA module, including the Adagrad optimizer, follows the same domain decomposition and performs all optimization steps locally within each grid column, requiring no additional inter-process communication. As a result, the Adagrad-based VDA algorithm is fully compatible with the existing MPI+OpenMP parallelization of TerM and scales naturally from a single multicore node to multi-node supercomputers.

All numerical experiments reported in this paper were performed in a single-core mode on an M3 Pro processor (ARM64 architecture). The code was compiled using the GNU Fortran compiler version 15.2.0. For this configuration, a single forward model run required approximately 0.004 s for the SOCS model and 0.006 s for the RothC model. A complete variational data assimilation cycle, including 20 Adagrad iterations, required approximately 0.038 s for SOCS and 0.097 s for RothC.

Conclusion

In this work, we have implemented automatic construction of an adjoint model to the model specified in the carbon cycle model constructor (CCMC), and made the necessary modifications to the constructor code. Based on that, we have also built a solution to the system of variational data assimilation equations for restoring the initial conditions of the carbon model dynamics system. Numerical experiments with data assimilation using the SOCS and RothC models showed a substantial reduction in the model-observations misfit, confirming both the correctness of the data assimilation implementation and the practical value of the variational approach for state variables initialization.

The prospects of extending the variational assimilation method and its implementation presented in this paper include the following:

- moving from initial state estimation to *joint* optimization of initial state and model parameters; this requires adding derivatives of the base functions not only with respect to pools but also with respect to model parameters;
- adding quantification of uncertainty of the optimal solution via the posterior error-covariance matrix;
- further integrating CCMC into the land surface model TerM and optimizing the memory usage of variational data assimilation on modern supercomputers.

Thus, we have established a reproducible and scalable foundation for variational assimilation in CCMC – from automatic adjoint construction and gradient evaluation to practical validation on real soil data.

Acknowledgments

The mathematical method for data assimilation and its implementation in CCMC code was supported by the Ministry of Science and Higher Education of the Russian Federation as part of the program of the Moscow Center of Fundamental and Applied Mathematics under agreement No. 075-15-2025-345. Comparison of simulation data to observations is supported by Multidisciplinary research and educational school "The future of the planet" of the Lomonosov Moscow State University under agreement No. 23-Sh07-55.

References

1. Pryanishnikov institute of agrochemistry. `https://www.vniia-pr.ru/laboratorii/otdl-geoseti/lab-geogr-seti/` (2025), accessed: 2025-09-30

2. Benavides Pinjosovsky, H.S., Thiria, S., Ottlé, C., *et al.*: Variational assimilation of land surface temperature within the ORCHIDEE Land Surface Model Version 1.2.6. Geoscientific Model Development 10(1), 85–104 (2017). `https://doi.org/10.5194/gmd-2016-64`

3. Coleman, K., Jenkinson, D.S.: RothC-26.3-A Model for the turnover of carbon in soil. In: Evaluation of soil organic matter models: Using existing long-term datasets, pp. 237–246. Springer (1996)

4. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12, 2121–2159 (2011). `https://doi.org/10.5555/1953048.2021068`

5. Faykin, G.M., Stepanenko, V.M., Medvedev, A.I., *et al.*: Constructor of soil carbon dynamic models. Numerical Methods and Programming 26(3), 281–303 (2025). `https://doi.org/10.26089/NumMet.v26r320`

6. Herbst, M., Welp, G., Macdonald, A., *et al.*: Correspondence of measured soil carbon fractions and RothC pools for equilibrium and non-equilibrium states. Geoderma 314, 37–46 (2018). `https://doi.org/10.1016/j.geoderma.2017.10.047`

7. Kaminski, T., Knorr, W., Schürmann, G., *et al.*: The BETHY/JSBACH carbon cycle data assimilation system: Experiences and challenges. Journal of Geophysical Research: Biogeosciences 118(4), 1414–1426 (2013). `https://doi.org/10.1002/jgrg.20118`

8. Kuppel, S., Peylin, P., Chevallier, F., *et al.*: Constraining a global ecosystem model with multi-site eddy-covariance data. Biogeosciences 9(10), 3757–3776 (2012). `https://doi.org/10.5194/bg-9-3757-2012`

9. Lions, J.L.: Contrôle optimal de systèmes gouvernés par des équations aux dérivées partielles. Dunod Gauthier-Villars (1968)

10. Marchuk, G.I., Orlov, V.V.: On the theory of conjugate functions. In: Krupchinsky, P.A. (ed.) Neutron physics. Sat. articles. Atomizdat, Moscow. p. 30–45. (1961)

11. Peylin, P., Bacour, C., MacBean, N., *et al.*: A new stepwise carbon cycle data assimilation system using multiple data streams to constrain the simulated land surface carbon cycle. Geoscientific Model Development 9(9), 3321–3346 (2016). `https://doi.org/10.5194/gmd-9-3321-2016`

12. Raoult, N.M., Jupp, T.E., Cox, P.M., Luke, C.M.: Land-surface parameter optimisation using data assimilation techniques: the adJULES system V1.0. Geoscientific Model Development 9(8), 2833–2852 (2016). `https://doi.org/10.5194/gmd-9-2833-2016`

13. Ryzhova, I.M.: Analysis of soil organic matter dynamics based on minimal carbon-cycle models. Russia's Soils-Strategic Resource: Abstracts of the 8th Congress of the V. V. Dokuchaev Soil Science Society and the School of Young Scientists on Soil Morphology and Classification (Syktyvkar, 2020–2022) 2, 130–131 (2021)

14. Sasaki, Y.: An objective analysis based on the variational method. Meteor. Soc. Japan (1958)

15. Sasaki, Y.: Some basic formalisms in numerical variational analysis. Monthly Weather Review 98(12), 875–883 (1970). `https://doi.org/10.1175/1520-0493(1970)098<0875:SBFINV>2.3.CO;2`

16. Scholze, M., Kaminski, T., Knorr, W., *et al.*: Simultaneous assimilation of SMOS soil moisture and atmospheric CO2 in-situ observations to constrain the global terrestrial carbon cycle. Remote sensing of environment 180, 334–345 (2016). `https://doi.org/10.1016/j.rse.2016.02.058`

17. Scholze, M., Kaminski, T., Rayner, P., *et al.*: Propagating uncertainty through prognostic carbon cycle data assimilation system simulations. Journal of Geophysical Research: Atmospheres 112(D17) (2007). `https://doi.org/10.1029/2007JD008642`

18. Shutyaev, V.P.: Methods for observation data assimilation in problems of physics of atmosphere and ocean. Izvestiya, Atmospheric and Oceanic Physics 55(1), 17–34 (2019). `https://doi.org/10.1134/S0001433819010080`

19. Stepanenko, V.M.: River routing in the INM RAS-MSU land surface model: Numerical scheme and parallel implementation on hybrid supercomputers. Supercomputing Frontiers and Innovations 9(1), 32–48 (2022). `https://doi.org/10.14529/jsfi220103`

20. Stepanenko, V.M., Medvedev, A.I., Bogomolov, V.Y., *et al.*: Land surface scheme TerM: the model formulation, code architecture and applications. Russian Journal of Numerical Analysis and Mathematical Modelling 39(6), 363–377 (2024). `https://doi.org/10.1515/rnam-2024-0031`

21. Taghizadeh-Toosi, A., Cong, W.F., Eriksen, J., *et al.*: Visiting dark sides of model simulation of carbon stocks in European temperate agricultural soils: allometric function and

model initialization. Plant and Soil 450(1), 255–272 (2020). https://doi.org/10.1007/s11104-020-04500-9

22. Thum, T., Zaehle, S., Köhler, P., *et al.*: Modelling sun-induced fluorescence and photosynthesis with a land surface model at local and regional scales in northern Europe. Biogeosciences 14(7), 1969–1987 (2017). https://doi.org/10.5194/bg-14-1969-2017

23. Verbeeck, H., Peylin, P., Bacour, C., *et al.*: Seasonal patterns of CO2 fluxes in Amazon forests: Fusion of eddy covariance data and the ORCHIDEE model. Journal of Geophysical Research: Biogeosciences 116(G2) (2011). https://doi.org/10.1029/2010JG001544

24. Wutzler, T., Reichstein, M.: Soils apart from equilibrium–consequences for soil carbon balance modelling. Biogeosciences 4(1), 125–136 (2007). https://doi.org/10.5194/bg-4-125-2007

DOI: 10.14529/jsfi250407

Novel Oxygen-Deficient Centers and Other Intrinsic Defects in Amorphous SiO_2: Quantum Molecular Dynamics Simulations

Vladimir B. Sulimov[1], *Danil C. Kutov*[1], *Alexey V. Sulimov*[1], *Fedor V. Grigoriev*[1], *Alexander A. Tikhonravov*[1]

The formation of stoichiometric and oxygen-deficient amorphous states of a-SiO_2 from a corresponding crystal was simulated using the melting-quenching procedure. To simulate the entire process, quantum molecular dynamics implemented in the VASP program and supercells containing 64 Si atoms and 128 O atoms were used. This size allows modeling the formation of rings with a small number of Si–O–Si bridges. At given heating and cooling rates of 0.5 K/fs, the transformation of the crystal's atomic network into a disordered structure was studied depending on the melt stabilization temperature. It is shown that despite the strong change in the topology of the atomic network in a-SiO_2 compared to the crystal, the bulk of the atoms constitute a continuous network of SiO_4 tetrahedra linked by oxygen vertices. Local disturbances of this arrangement of atoms are intrinsic point defects of SiO_2, which are given special attention. It has been shown that most of the defects present in oxygen-deficient states are also present in stoichiometric states of a-SiO_2. Along with the known intrinsic defects of SiO_2, new defects were also identified, including those associated with oxygen deficiency. It is shown that for two generally accepted models of oxygen-deficient centers (ODCs) in a-SiO_2, the energy of formation of an oxygen vacancy is significantly lower than the energy of formation of a twofold coordinated silicon atom. The point defects of a-SiO_2 identified in this work create a foundation for the interpretation of experimental data on the radiation resistance of optical fibers, the effects of high-power laser radiation on optical coatings, and in microelectronics, where insulating layers based on a-SiO_2 are used.

Keywords: quantum molecular dynamics, amorphous SiO_2, point defects, oxygen deficiency, oxygen vacancy, ODC.

Introduction

Amorphous silica or a-SiO_2 and silica glass or quartz glass are names of disordered states of silicon dioxide. This non-crystalline material has unique properties and is widely used in microelectronics for insulating layers in most of semiconductor devices, in high-speed communication lines as a base component of optical fibers, in multilayer optical coating as low-refractive index thin films, and in many other applications. The structure of perfect a-SiO_2 at the atomic level is usually imagined as a disordered framework of SiO_4-tetrahedra bound to each other by common oxygen atoms in the tetrahedra vertexes. Many different silicon dioxide crystalline polymorphs (quartz, cristobalite, tridymite, coesite) have such structure with different packing of SiO_4-tetrahedra. Perfect silicon dioxide in its crystalline or amorphous form has one of the widest transparency windows, the band gap, about 9 eV. Disruptions to the ideal network of interatomic bonds in pure a-SiO_2 are called intrinsic point defects, and some of them are associated with electron states in the band gap. Existence of these states narrows the transparency window and results in absorption of the light irradiation with photon energies below 9 eV. This absorption leads to additional losses in optical fibers and decreases the laser damage threshold of optical coating. Band gap states of point defects play a role of hole or electron traps creating radiation-induced color centers in silica-based fibers [1] and different phenomena in semiconductor devices such as interface charge, leakage current, *etc.* [21].

[1]Research Computing Center of Lomonosov Moscow State University, Moscow, Russian Federation

Among silica intrinsic point defects, oxygen-deficient centers or ODCs are most widely discussed as precursors of various radiation-induced color and/or ESR-active centers, ESR – Electron Spin Resonance. ODCs also play a key role in the formation of the UV-written refractive index gratings in Ge-doped silica-based fibers [23]. Authors of various publications use two ODC models to interpret the phenomena observed in silicon dioxide: the oxygen vacancy, ODC(I), and the twofold coordinated silicon atom, ODC(II) or =Si, where the symbol "=" denotes two Si–O bonds that connect the silicon atom to the rest of the SiO_4-tetrahedra network. Note that the stoichiometry of the SiO_2 supercell with an oxygen vacancy is equal to the stoichiometry of the SiO_2 supercell with a =Si defect. The oxygen vacancy model refers to a common defect in crystals, but =Si originally relates to a defect at silica surface. Oxygen vacancies have been discussed as models for various defects in quartz glass, including ODC models, and corresponding calculations have been carried out for over 50 years, *e.g.* see [3, 25, 27] and references therein. The model of ODC in the form of a twofold coordinated silicon atom =Si was introduced much later [16, 19], but is currently being discussed for the interpretation of experimental data along with the oxygen vacancy [6, 14].

Despite the more or less general acceptance of these two models, the existence of oxygen vacancies and =Si atoms in a-SiO_2 is based only on indirect experimental evidences, since the atomistic structure of point defects can only be obtained by analyzing ESR spectra, but ODCs do not manifest themselves in ESR. Therefore, we decided to obtain direct evidence for the existence of these ODC models using atomistic computer simulations.

We recently performed quantum molecular dynamics simulations of amorphous silica using the melting-quenching procedure [22, 24] and found that in most cases, the resulting amorphous stoichiometric silicon dioxide does not contain oxygen vacancies or twofold coordinated silicon atoms. In these works, we used spin-unpolarized quantum MD calculations, which in general do not allow to describe adequately breaking and restoring interatomic bonds during the melting-quenching procedure.

In the present work, carried out with the aim of searching for possible intrinsic defects of a-SiO_2, the results of modeling stoichiometric and oxygen-deficient a-SiO_2 are described. Modeling was carried out using the same melting-quenching procedure as in [22, 24], but with spin-polarized calculations for a more adequate description of the breaking and restoring of Si–O bonds. We investigated more thoroughly than in works [22, 24] the transformation from the atomic lattice topology of the crystal to the more disordered atomic network topology of a-SiO_2. Oxygen deficiency was created in the initial crystal or a-SiO_2 by removing one or two oxygen atoms, creating single oxygen vacancies or divacancies. In this process, several dozen models of amorphous states of stoichiometric and oxygen-deficient a-SiO_2 were obtained, and the intrinsic point defects formed in them were identified, including 12 defects that can be attributed to oxygen deficiency. The obtained results allow us to take a different look at the models of oxygen-deficient centers in amorphous silicon dioxide and the mechanisms of their participation in various processes under ionizing or laser irradiation.

The article is organized as follows. Section 1 is devoted to the short description of the methods used for modeling amorphous states of silicon dioxide. In Section 2, we present results of modeling stoichiometric and oxygen-deficient amorphous states of SiO_2 and analysis of their point defects. Section 3 summarizes the results of the study, discusses the intrinsic point defects identified, particularly those related to oxygen deficiency, and highlightes some of the problems encountered. Conclusion contains a brief summary of the results of the work, final conclusions and points the directions for further work.

1. Models and Methods

The method for obtaining amorphous states of SiO_2 involves modeling the melting-quenching process of a crystal using quantum molecular dynamics, MD. The VASP program [10] was used for modeling. β-cristobalite was used as the starting crystal, and the simulations were performed with its supercell containing 192 atoms (64 Si atoms and 128 O atoms) with periodic boundary conditions. The system was heated and cooled at a rate of 0.5 K/fs; the melt was stabilized at a given T_{melt} within 6 ps in most cases, which was sufficient to achieve temperature stabilization at T_{melt}. In some cases, the stabilization time at T_{melt} was significantly increased to compare the properties of the obtained amorphous states. After cooling the system to 300 K, it was stabilized for 12 ps, after which the supercell energy was optimized. In all cases of melting-quenching, the time of 12 ps was sufficient to stabilize energy of the system at 300 K. Molecular dynamics simulations were performed in the NPT ensemble when the number of atoms N, pressure P and temperature T were constant. In this modeling, the volume of the system was not fixed and could change in accordance with strong temperature changes during the melting-quenching process, which made it possible to avoid strong internal stresses in the system. During MD simulation and during energy optimization of the supercell, its volume and shape, and the positions of all atoms could change. The modeling methodology is described in more detail in the works [22, 24].

2. Results

Our previous studies using spin-unpolarized quantum MD calculations have shown that there is a narrow range of melt stabilization temperature T_{melt} above which amorphization of silicon dioxide crystals occurs during the melting-quenching procedure [22, 24]. Amorphization here refers to a change in the topology of the atomic network of silicon dioxide from a regular crystalline topology to a less ordered amorphous topology. One of the characteristic features of amorphous topology is the presence of low-membered rings built from Si–O–Si bridges formed by SiO_4-tetrahedra connected to each other by their oxygen vertices. For example, the high temperature β-cristobalite crystal contains only 6-membered rings, α-quartz contains 6- and 8-membered rings, but in amorphous silica there are also 3-, 4- and 5-membered rings [13, 31]. In our spin-unpolarized simulation, amorphization of an initial stoichiometric β-cristobalite crystal happens above $T_{melt} = 4700$ K. Therefore, we investigated non-stoichiometric compositions for several values of T_{melt} near 4700 K, but first we briefly present new results on amorphization of stoichiometric SiO_2, modelled by the melting-quenching procedure using spin-polarized quantum MD calculations (keyword ISPIN = 2 in VASP version 5.4.4 compiled for GPU).

2.1. Stoichiometric Amorphous States

Amorphous states of SiO_2 were obtained from β-cristobalite crystal using the melting-quenching procedure at different melt stabilization temperatures. The results are presented in Tab. 1, where for a-SiO_2-4800, the E_g value given in brackets corresponds to the transition of an electron with the second spin direction, and the designations of point defects are explained in the text below. For brevity, we will further denote, for example, as a-SiO_2-4700 the amorphous state of silicon dioxide obtained by the melting-quenching procedure with stabilization of the melt at 4700 K for 6 ps.

Table 1. Stoichiometric SiO_2. Stabilization of the melt was 6 ps for all T_{melt}

Structure	Crystal	Amorphous SiO_2, T_{melt}, °K				
		4500	4600	4700	4800	4900
E_{opt}, eV	−1516.53	−1513.90	−1519.55	−1519.51	−1482.89	−1495.00
Defects	–	$2BO_3C$, ≡Si–O	–	–	=Si, OV, Si_5, 2BOC, $2BO_3C$, O_2	Si_5, O_3, 2BOC
R(Si–Si), Å	3.10	2.47	3.16	3.10	3.00	3.19
ρ, g/cm^3	1.92	2.17	2.119	2.126	2.056	2.256
E_g, eV	5.35	2.31	5.56	5.55	3.42 (2.07)	3.41

In Tab. 1, E_{opt} represents the energy of the stoichiometric supercell after energy optimization from the crystal geometry or from the disordered state at the end of the melting-quenching procedure, R(Si–Si) is the distance between Si atoms adjacent to the oxygen atom that will be eliminated when creating an oxygen vacancy (see Section 2.2), ρ is the mass density of the system, E_g = E(LUMO) – E(HOMO), where E(LUMO) and E(HOMO) are energies of the lowest unoccupied and highest occupied molecular orbitals, respectively. The value of E_g estimates the lowest energy of electron excitations, for example, this is the width of the forbidden energy gap of the oxide crystal. However, if a supercell, crystalline or amorphous, contains several point defects with energies of electron states in the forbidden energy gap of the defect-free system, then the value of E_g, strictly speaking, cannot provide an estimate of the lowest excitation energy of the system, since the HOMO and LUMO states can belong to different point defects, widely separated in space, and the oscillator strength of the HOMO-to-LUMO transition in this case will be negligibly small. These considerations should be kept in mind when reading this article, in which, for the sake of completeness, we provide the values of E_g for all the cases considered.

Visual analysis of the position of atoms in the obtained amorphous supercells shows that at all melt stabilization temperatures T_{melt} up to and including 4400 K, the topology of the arrangement of atoms is the same as in the initial crystal with relatively small deviations of Si–O–Si angles from their values in the crystal. At T_{melt} = 4500 K, the first point defects appear. These are three following point defects. Two of them, are painted green and gray in Fig. 1. These defects are a 2-Bridging Oxygen Center (2BOC), which is two adjacent SiO_4-tetrahedra linked to each other by two oxygen vertices [22, 24], and associated with it a threefold coordinated oxygen atom O_3, respectively. The third defect is designated as ≡Si–O, where the symbol "≡" stands for three ordinary Si–O bonds with the remaining SiO_2 atomic network, and the oxygen atom has only one Si–O bond. This is a non-bridging oxygen atom.

One of the two oxygen atoms of 2BOC forms a third bond to a Si atom from another SiO_4-tetrahedron; hereinafter such a combination of 2BOC and O_3 is designated as $2BO_3C$, where the index "3" denotes the presence of a threefold coordinated oxygen atom O_3. In Tab. 1, for a-SiO_2-4500, the value of R(Si-Si) corresponds to the distance between two Si atoms that make up the $2BO_3C$ defect. The topology of the atomic network of other supercell regions of a-SiO_2-4500 is close to the crystal structure with some variations of Si–O–Si angles. Obviously,

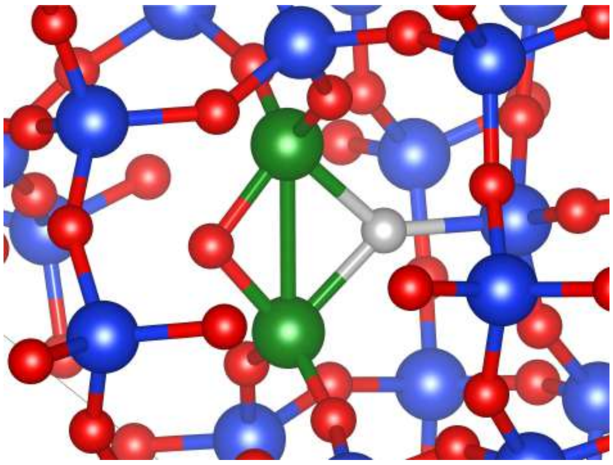


Figure 1. The $2BO_3C$ defect in a-SiO_2-4500; green balls are silicon atoms of the 2BOC defect, a grey ball is a threefold coordinated oxygen atom O_3

the 2BOC defect does not violate stoichiometry of the SiO_2 system. Also, two defects, $2BO_3C$ and ≡Si–O, do not violate stoichiometry of a-SiO_2-4500, which is expected, since the number of atoms in the supercell is constant in the MD simulations because we use the NPT ensemble, see Section 1. Indeed, the $2BO_3C$ center can be represented as a combination of the 2BOC center and a threefold coordinated silicon atom ≡Si. The latter in a combination with the ≡Si–O defect restore stoichiometry of SiO_2. The ≡Si–O defect has long been considered as a model for non-bridging oxygen hole center, the so-called NBOHC, which explains some optical absorption and luminescence bands in SiO_2 [15, 17, 18].

When the melt is stabilized at 4600 or 4700 K, then in the corresponding amorphous states, a-SiO_2-4600 and a-SiO_2-4700, the topology of the arrangement of atoms is again as in a defect-free crystal with small deviations of Si–O–Si angles from their crystalline values.

An increase in the duration of melt stabilization at T = 4700 K from 6 ps to 19 ps leads to a strong rearrangement of the topology of the atomic network, an increase in the energy of corresponding a-SiO_2-4700_19ps by more than 20 eV and the appearance of point defects: 2BOC, fivefold coordinated Si_5 and threefold coordinated ≡Si silicon atoms, which results in a sharp decrease of E_g to 2.02 eV due to the electron states of defects in the forbidden energy gap of SiO_2. The subscript “5” shows that the silicon atom has five Si–O bonds with five oxygen neighbors. The atomic network contains several four- and five-membered rings built from Si–O–Si bridges, which are absent in the crystal network. The ≡Si defect is a well known model of one of the most studied ESR active E′-centers in pure SiO_2 [5, 28].

The five Si–O bonds of Si_5 atoms are noticeably elongated to 1.68–1.85 $\mathring{A}$ compared to the length of the Si–O bonds of regular fourfold coordinated Si atoms 1.62–1.63 $\mathring{A}$ in the defect-free regions of the supercell; three adjacent oxygen atoms are located in the vertices of a triangle, in the center of which is the Si atom, and the other two adjacent oxygen atoms are at opposite ends of a straight line passing through the silicon atom perpendicular to the plane of the triangle. Note that from the stoichiometry point of view, the Si_5 defect is equivalent to one regular fourfold coordinated silicon atom forming a SiO_4-tetrahedron, and a ≡Si–O defect with a non-bridging oxygen atom.

At higher melt stabilization temperatures, a greater number of 2BOC defects and small rings built from three, four or five Si–O–Si bridges appear. Besides, some additional point defects can be identified in a-SiO_2-4900 and a-SiO_2-4800. In a-SiO_2-4900, Si_5 and O_3 defects are found. Three Si–O bonds of length 1.77–1.89 $\AA$ of the O_3 defect are somewhat elongated comparing to a regular Si–O bond of 1.62 $\AA$. Note that from the stoichiometry point of view, the threefold coordinated oxygen atom O_3 is equivalent to the threefold coordinated silicon atom ≡Si, therefore, the O_3 defect can be classified as an oxygen deficient center (ODC) in the same way as the ≡Si. Also, a pair of defects Si_5 and O_3 corresponds to the stoichiometry of defect-free SiO_2.

The a-SiO_2-4800 supercell has two unpaired electrons from the triplet state of the interstitial oxygen molecule O_2, which appears during the melting-quenching process, and this amorphous state is significantly higher in energy than the other amorphous states we obtained, which do not have unpaired electrons. Also, there are a twofold coordinated silicon atom =Si, an oxygen vacancy in the form of ≡Si–Si≡ bond between two adjacent to the oxygen vacant site Si atoms with the Si–Si bond length 2.30 $\AA$, two Si_5 defects and two $2BO_3C$ defects. The presence of all these defects does not disturb the correct stoichiometry of SiO_2. Indeed, from the O_2 molecule, one oxygen atom binds to an oxygen vacancy and restores a regular oxygen bridge. Two Si_5 defects together with the =Si defect restore the stoichiometry of SiO_2, and two $2BO_3C$ centers give two ≡Si defects, which, together with the remaining oxygen atom, also restore the stoichiometry.

The Si–O bonds of the twofold coordinated silicon atom =Si are equal to 1.66 and 1.69 $\AA$, they are only slightly larger than the length of the regular Si–O bond in defect free regions. The angle O–Si–O is equal to 106.4°. An oxygen vacancy has the form of a ≡Si–Si≡ bond, which for brevity we will simply call the Si–Si bond.

An independent run of the melting-quenching procedure to obtain a-SiO_2-4800 again leads to a high energy state with two unpaired electrons. This amorphous state has an energy 3.86 eV lower and its mass density $\rho = 2.141$ g/cm^3 is slightly higher than the corresponding values of the a-SiO_2-4800 state presented in Tab. 1. However, in the new a-SiO_2-4800 state, there are no the =Si defect, oxygen vacancy, $2BO_3C$ and Si_5 defects, as well as the interstitial O_2 molecule, that were present in the previous a-SiO_2-4800 state. Instead, there are several 2BOC defects, several 3-, 4- and 5-membered rings, the ≡Si–O defect, and three threefold coordinated Si atoms: two Si atoms of the type ≡Si and one Si atom of the type =Si–O, the latter being a threefold coordinated Si atom, the third Si–O bond of which is linked to a non-bridging oxygen atom. One ≡Si atom has a pyramidal conformation with somewhat elongated Si–O bonds (1.63–1.68 $\AA$) and the other ≡Si atom has a planar conformation, in which the Si atom is in the plane of three adjacent oxygen atoms, with slightly shortened bond lengths (1.56–1.58 $\AA$). One unpaired electron is on the ≡Si atom with a pyramidal conformation and the second unpaired electron is on the non-bridging oxygen atom of the defect =Si–O. The Si–O bond of the non-bridging oxygen atom is noticeably shorter, 1.52 $\AA$, than other two bonds of this Si atom, the lengths of which are close to the length of regular Si–O bonds in silicon dioxide 1.62 $\AA$. In the latter a-SiO_2-4800 state, $E_g = 1.94$ and 1.51 eV for the excitation of electrons with different spins.

As can be seen from Tab. 1, the density of most of the obtained stoichiometric amorphous states, with the exception of the a-SiO_2-4900 state, is in the range of 2.10–2.17 g/cm^3, which corresponds to the results of our previously conducted spin-unpolarized calculations [22, 24]. The obtained values of the density of amorphous states are slightly lower than the experimental density of quartz glass 2.20 g/cm^3, and this small difference is within the errors of the PBE functional [8] and the PAW method [11].

As we can see in Tab. 1, the energy gap E_g of the crystal and of the defect free amorphous states a-SiO_2-4600 and a-SiO_2-4700 is about 5.4–5.6 eV. These values are much less than experimentally measured band gap of silicon dioxide, which is about 9 eV [2, 30]. This is a well know drawback of the PBE [8] functional and these values can be improved using PBE0 functional [12]. The formation of point defects leads to the appearance of electron states of defects in the SiO_2 forbidden gap and, as a consequence, to a significant decrease in the E_g values (see Tab. 1 for a-SiO_2-4800 and a-SiO_2-4900).

The conducted spin-polarized MD simulations resulted in practically the same amorphous states and their point defects as those obtained by spin-unpolarized quantum MD simulations [22, 24] using the melting-quenching procedure with the same parameters: melting and quenching rate, melt stabilization time, and stabilization time of the final amorphous state at 300 K. In addition, a new point defect =Si–O was discovered, which was not detected in the spin-unpolarized simulation. States with unpaired electrons, i.e., ESR-active states, were also obtained, which, however, are significantly higher in energy than the ESR-inactive states. With the same melting-quenching parameters, the use of spin-polarized calculations leads to amorphous states with a richer set of intrinsic defects than similar spin-unpolarized calculations, the results of which are presented in [22, 24].

Properties of amorphous states obtained from quartz crystal are close to the properties of amorphous states obtained from cristobalite crystal. In contrast to this, amorphization from stishovite crystal occurs at much lower temperatures due to instability of stishovite at normal pressure. Using the same melting-quenching procedure, we found that radical restructuring of the stishovite atomic network topology occurred at $T_{melt} = 3000$ K, when sixfold coordinated Si atoms and threefold coordinated oxygen atoms in the stishovite atomic network are transformed into a quartz glass network with fourfold and twofold coordinated Si and O atoms, respectively. Topology of atomic network of amorphous states obtained with $T_{melt} \geq 3000$ K from stishovite is, in general, the same as one of amorphous states obtained from cristobalite with the same set of possible point defects.

In the following research, we will carry out modeling of oxygen-deficient centers based on amorphous structures obtained from the cristobalite crystal. Three types of oxygen deficient centers ODCs are considered: a single oxygen vacancy OV, two spatially separated oxygen vacancies OV+OV, and an oxygen divacancy. Three methods of modeling are used: (**A**) ODC is created in a cristobalite crystal or in one of the above-obtained models of stoichiometric a-SiO_2, and then the energy of the corresponding supercell is optimized by changing its shape and volume, as well as the position of all its atoms; (**B**) ODC is created in the cristobalite crystal and then the melting-quenching procedure is carried out for different melt stabilization temperatures T_{melt}; (**C**) a supercell of a-SiO_2 created by the (**A**) approach with the corresponding ODC is subjected to the melting-quenching procedure with the same melt stabilization temperature that has been used in the preparation of the initial amorphous stoichiometric state.

2.2. Single Oxygen Vacancy, OV

A. Oxygen vacancy is created in existing stoichiometric states. The oxygen vacancy (OV) is created in the corresponding structure, cristobalite or amorphous (see Tab. 1), and then the energy of the supercell is optimized, and as a result of this, two adjacent to the vacant site silicon atoms relax to each other forming the Si–Si bond. Characteristics of the supercell with the oxygen vacancy are presented in Tab. 2. In a-SiO_2-4500, two types of oxygen vacancy were created by removing one of the two oxygen atoms included in the $2BO_3C$ defect (Fig. 1). In Tab. 2, the 4500_2BOC column corresponds to the removal of the threefold coordinated oxygen atom, and the 4500_2BOC-1 column corresponds to the removal of the twofold coordinated oxygen atom from the $2BO_3C$ defect. In the table E_{opt}(OV) is the energy of the supercell with an oxygen vacancy after optimization, R(Si–Si) is the distance between Si atoms adjacent to the vacant oxygen site, E_g = E(LUMO) – E(HOMO), E_{form}(OV) is the formation energy of the oxygen vacancy calculated as follows: $E_{form}(OV) = E_{opt}(OV) + E(O) - E_{opt}$, where E_{opt} is the optimized energy of the supercell without oxygen vacancy (see Tab. 1), and $E(O) = -1.897$ eV is the energy of a single O atom obtained using spin-polarized calculations with the VASP program, ISPIN = 2, with the same PBE functionals and the same value of the ENCUT parameter that were used for the crystal and amorphous calculations presented in this report. If the oxygen vacancy is created in the defect free region of a-SiO_2-4500, then its properties are practically the same as those in defect free states a-SiO_2-4600 and a SiO_2-4700. Thus, for a-SiO_2-4500, an oxygen vacancy was created by removing an oxygen atom from the $2BO_3C$ defect that exists in the stoichiometric system.

Table 2. Main characteristics of oxygen vacancy created in crystalline or amorphous silicon dioxide supercell

Structure	Oxygen vacancy						
	Crystal	Amorphous SiO_2, T_{melt}, °K					
		4500_2BOC	4500_2BOC-1	4600	4700	4800	4900
E_{opt}(OV), eV	−1506.42	−1502.73	−1504.07	−1509.52	−1509.44	−1476.94	−1484.59
R(Si–Si), $\mathring{A}$	2.48	2.49	2.21	2.42	2.42	2.38	2.52
ρ, g/cm^3	1.92	2.18	2.16	2.12	2.12	2.14	2.27
E_{form}(OV), eV	8.22	9.27	7.93	8.13	8.18	7.92	8.51
E_g, eV	5.34	0.46	2.29	5.27	5.32	2.00	3.39

In cristobalite and all amorphous states, except those indicated in Tab. 2 as 4500_2BOC and 4500_2BOC-1, oxygen vacancies have the form of the Si–Si bond. The distances between silicon atoms adjacent to vacant oxygen sites vary in the range from 2.38 to 2.52 $\mathring{A}$ and are significantly smaller than distances between silicon atoms in regular Si–O–Si bridges of the defect-free region of silicon dioxide, indicating the formation of a strong Si–Si covalent bond. The variations of lengths of Si–Si bonds are apparently associated with some scattering of the positions of atoms in the immediate vicinity of the vacancies, caused by the structure of the amorphous state. The formation energy of the oxygen vacancy in the form of the Si–Si bond is in the range from 7.92 to 8.51 eV, and this scatter is also apparently due to some difference in the positions of the atoms in the vacancy environment. The formation energy of an oxygen vacancy created by removing an oxygen atom from the $2BO_3C$ defect is close to the above values for other amorphous states and the crystal, when a twofold coordinated oxygen atom is removed, but is slightly higher (9.27 eV) in the case of removing a threefold coordinated oxygen atom due to the breaking of three Si–O bonds.

The energy E_g for the cristobalite crystal and amorphous defect-free states a-SiO_2-4600 and a-SiO_2-4700 in the presence of an oxygen vacancy is only slightly less than the corresponding energy gap of defect-free supercells of stoichiometric compositions (see Tab. 1). This qualitatively corresponds to the excitation energy of the oxygen vacancy in quartz crystal 7.6 eV, calculated by an *ab initio* method [27]; this energy is in the vacuum UV region and is slightly less than the silicon dioxide forbidden energy gap [2, 30].

The structure of amorphous a-SiO_2-4700_19ps (the melt stabilized during 19 ps) contains a hand-created oxygen vacancy and the same defects that existed in this amorphous state before the creation of the oxygen vacancy. Comparing the values of E_g = 5.32 eV for a-SiO_2-4700 in Tab. 2 (here, apart from the vacancy, there are no other defects) and E_g = 2.06 eV for a-SiO_2-4700_19ps, it is clear that such a strong decrease in E_g in the latter can be attributed to the presence of the electron states of the 2BOC, Si_5 and ≡Si defects in the SiO_2 energy gap, since four-membered rings create only electron states near the top of the valence band, which leads to only a very small decrease in E_g [26]. In the a-SiO_2-4800 and a-SiO_2-4900 structures with an oxygen vacancy, the E_g values are also determined by the presence of corresponding point defects with electron states in the band gap of silicon dioxide in these amorphous structures even before the creation of an oxygen vacancy.

The conformations of two types of oxygen vacancies created by removing one of two oxygen atoms from the $2BO_3C$ defect existing in the stoichiometric a-SiO_2-4500 state are different from the simple Si–Si bond. When the threefold coordinated oxygen atom is removed from $2BO_3C$ (this state is denoted as 4500_2BOC in Tab. 2), the ≡Si defect is formed in a pyramidal conformation near the vacancy, and the 2BOC defect retains its original geometry but without one removed bridging oxygen atom; in particular, the R(Si–Si) distance of 2.49 $\AA$ is almost the same as in the original stoichiometric a-SiO_2-4500 in the $2BO_3C$ defect. The resulting oxygen-deficient center can be interpreted either as two adjacent threefold coordinated silicon atoms with a common oxygen atom, or as two twofold coordinated silicon atoms linked to each other by an oxygen bridge =Si–O–Si=, where symbol "=" designates two Si–O bonds with a regular part of SiO_2 atomic network. When the twofold coordinated oxygen atom is removed from $2BO_3C$ (this state is denoted as 4500_2BOC-1 in Tab. 2), after energy optimization, two Si atoms of the $2BO_3C$ defect remain to be linked by the threefold coordinated oxygen atom, the distance between these silicon atoms decreases to 2.21 $\AA$ indicating strengthening of Si–Si bond between these atoms. This oxygen-deficient center can be interpreted as two twofold coordinated silicon atoms linked to each other by a threefold coordinated oxygen atom.

B. Oxygen vacancy in initial cristobalite + the melting-quenching procedure. An oxygen vacancy is created near the center of the supercell of the initial cristobalite crystal, followed by the melting-quenching procedure to obtain amorphous states at different melt stabilization temperatures T_{melt}. Main characteristics of these amorphous supercells with the initially created oxygen vacancy are presented in Tab. 3.

In a-SiO_2-4600, the atomic network topology is almost the same as in the initial crystal with some deviations of Si–O–Si angles from their crystalline values. However, besides the oxygen vacancy there are three five-membered and one four-membered strained ring. The presence of these rings increases the supercell energy by 4.8 eV (compare E_{opt}(OV) in Tabs. 2 and 3, columns 4600), since stoichiometric a-SiO_2-4600 has only six-membered rings as in the initial crystal. The oxygen vacancy created in the initial crystal keeps its form as a Si–Si bond with R(Si–Si) = 2.32 $\AA$ after the melting-quenching procedure, but its two Si neighbors have changed

Table 3. Characteristics of the amorphous supercell with an oxygen vacancy after the melting-quenching procedure from the initial crystalline supercell

Structure	Oxygen vacancy				
	Amorphous SiO_2, T_{melt}, °K				
	4600	4600-1	4700	4800	4900
E_{opt}(OV), eV	−1504.73	−1488.54	−1506.81	−1473.52	−1486.28
Defects	OV	2BOC, $2BO_3C$, Si_5, ≡Si, –Si–O	OV, 2BOC	2BOC, =Si, ≡Si–O, ≡Si, Si_5	2BOC, =Si–O, ≡Si
R(Si–Si), $\mathring{A}$	2.32	–	2.37	–	–
ρ, g/cm^3	2.196	2.172	2.147	2.012	2.058
E_g, eV	4.98	1.99 3.24	5.08	1.87	1.91

their numbers – each atom in the supercell has its own unique number in the list of all atoms in the supercell, which is preserved during MD simulations. This means that the oxygen vacancy changes its position in the supercell during the melting-quenching procedure due to the hopping diffusion of oxygen atoms through the nodes of the SiO_4-network. Two Si atoms adjacent to the vacant site form Si–O bonds with six neighboring oxygen atoms, which are somewhat elongated to 1.64–1.65 $\mathring{A}$ compared to the Si–O bonds of 1.62–1.63 $\mathring{A}$ in defect-free regions.

An independent run of the melting-quenching procedure gives a new configuration of the amorphous state a-SiO_2-4600-1 with two unpaired electrons. The energy of this state is 16 eV higher than the energy of the a-SiO_2-4600 state obtained during the first run of melting-quenching and having no unpaired electrons, see the 4600 column in Tab. 3. The topology of the atomic network in the amorphous state of a-SiO_2-4600-1 differs significantly from the topology of the weakly disordered arrangement of atoms in a-SiO_2-4600, which is close to topology of cristobalite. In the a-SiO_2-4600-1 state, there are two 2BOC centers, one $2BO_3C$, two fivefold coordinated silicon atoms Si_5, two threefold coordinated silicon atoms ≡Si, and also a twofold coordinated silicon atom, in which one of the neighboring oxygen atoms is non-bridging, and this defect can be designated for brevity as –Si–O. The Si–O bond of the non-bridging atom (1.54 $\mathring{A}$) is much shorter than the Si–O bond of a regular bridging oxygen atom (1.62 $\mathring{A}$). In the a-SiO_2-4600-1 state, there are also 3-, 4- and 5-membered rings. It is easy to make sure that stoichiometry of this amorphous state is equal to one oxygen vacancy in the supercell. Really, the 2BOC defects do not violate the stoichiometry of SiO_2, two Si_5 defects being combined with two ≡Si defects restore stoichiometry of SiO_2, the $2BO_3C$ is transformed to 2BOC and ≡Si, and the latter being combined with the –Si–O defect gives the twofold coordinated silicon atom =Si, which corresponds to the stoichiometry of one oxygen vacancy in the supercell.

One of the two fivefold coordinated silicon atoms is included into one of the two 2BOC defects. One of the two threefold coordinated silicon atoms ≡Si has a pyramidal conformation. This silicon atom is linked by two Si–O bonds to regular fourfold coordinated silicon atoms, and the oxygen atom of the third Si–O bond is included in the 2BOC defect. The second threefold coordinated atom ≡Si has a planar conformation; it is linked by two Si–O bonds to regular fourfold coordinated silicon atoms and the third Si–O bond forms a bridge with the fivefold coordinated atom Si, which is a part of a 2BOC defect.

Two unpaired electrons are located at two separated defects: at ≡Si with pyramidal conformation and at non-bridging oxygen atom of the –Si–O defect. The former is the so-called paramagnetic E′-center and the latter is a new type of modified Non-Bridging Oxygen Hole Center (NBOHC), both of them are in the list of most common radiation-induced centers in silica glass and in silica-based fibers [5, 15, 17, 18, 28].

In a-SiO_2-4600-1, the values of E_g for excitation of electron with opposite directions of spins are 1.99 and 3.24 eV. The mass density of the a-SiO_2-4600-1 state is 0.024 g/cm^3 lower than density of a-SiO_2-4600.

In a-SiO_2-4700, the a-SiO_2 structure contains one oxygen vacancy, one 2BOC defect, three five-membered and one three-membered strained rings. There are no unpaired electrons in this state. Otherwise, this structure has not lost the ordered crystal topology with some angular disorder. The stoichiometric a-SiO_2-4700 has the SiO_4-network topology as in the initial crystal without the 2BOC defect and 5- and 3-membered rings, with only six-membered rings. The oxygen vacancy keeps its form as a Si–Si bond, but its two Si neighbors have changed again – a pair of other Si atoms have become adjacent to the vacant oxygen site. The distance between Si atoms neighboring to the vacant oxygen site R(Si-Si) is equal to 2.37 $\mathring{A}$.

An increase in the duration of melt stabilization at T_{melt} = 4700 K from 6 ps to 14 ps leads to a strong rearrangement of the topology of the atomic network, appearance of 3-, 4-, and 5-membered rings, an increase in the energy of corresponding a-SiO_2 by more than 26 eV, and the appearance of point defects: five 2BOC defects and two defects =Si–O. The oxygen vacancy has a form of Si–Si bond with R(Si-Si) = 2.31 $\mathring{A}$. The silicon atoms that are neighbors of the oxygen vacancy have changed again – a pair of other Si atoms now appear next to the oxygen vacancy. This is a different pair of atoms comparing with neighbors of the vacancy in the initial crystal and in the amorphous state obtained from the melt stabilized during 6 ps. Values of R(Si-Si) for the oxygen vacancies in a-SiO_2 obtained for T_{melt} = 4600 and 4700 K correspond well to the R(Si-Si) = 2.36 $\mathring{A}$ for the oxygen vacancy in the quartz crystal calculated with a full long-range atomic relaxation by the *ab initio* method [27]. The significantly larger distance R(Si-Si) = 2.48 $\mathring{A}$ obtained for the oxygen vacancy in the initial β-cristobalite crystal after its energy optimization (see Tab. 2) demonstrates the importance of long-range atomic relaxation and disorder for the properties of this defect.

In the =Si–O defect, Si atom is in the plane of its three oxygen neighbors, the lengths of two Si–O bonds are close to those in the defect-free region, and the Si–O bond with the non-bridging oxygen atom is noticeably shorter and is 1.53 $\mathring{A}$. Since there are no unpaired electrons in the system, it can be assumed that one of the valence electrons of the Si atom in such a defect passes to the non-bridging oxygen atom. Due to this, the defect acquires a noticeable dipole moment, and the additional Coulomb attraction of these opposite charges leads to an additional strengthening of the Si–O bond between the silicon atom and the non-bridging oxygen atom and to its shortening. In the obtained amorphous supercell, two =Si–O defects are located far from each other: the distance between corresponding Si atoms equals 6.63 $\mathring{A}$. The pair of =Si–O defects does not violate stoichiometry. Each defect in this pair complements the other, and they both form one 2BOC. Therefore, this pair of =Si–O defects does not relate to oxygen deficiency of the sample, as well as 2BOCs, and the oxygen vacancy is the only ODC in this sample.

At higher melt stabilization temperatures T_{melt} = 4800 and 4900 K, the SiO_4-network with the initial oxygen vacancy loses the ordered topology of the initial crystal, and the oxygen vacancy in the form of a Si–Si bond disappears, possibly due to the mobility of Si atoms in the melt. In both cases, there are no unpaired electrons, and many small-membered rings, 2BOC defects and pores appear. Besides, several additional point defects can be observed.

In a-SiO_2-4800, there is no oxygen vacancy, and instead the following defects are found: one twofold coordinated silicon atom =Si, two threefold coordinated silicon atoms ≡Si, a fivefold coordinated silicon atom Si_5 and the ≡Si–O defect with a non-bridging oxygen atom. The Si_5 defect is simultaneously a part of the 2BOC center and a part of the strained three-membered ring. One of the threefold coordinated Si atoms is also a part of another 2BOC defect. So, in this amorphous state, only the twofold coordinated silicon atom =Si can be definitely attributed to the oxygen deficiency, because remaining pairs of defects, Si_5 and ≡Si, as well as ≡Si–O and ≡Si, are combined to the perfect stoichiometry of silicon dioxide. In fact, stoichiometry of =Si is the same as an oxygen vacancy. The energy of this supercell is more than 33 eV higher than the energy of the supercell obtained from the melt stabilized at 4700 K for 6 ps (see Tab. 3).

In the a-SiO_2-4900 diamagnetic state, the =Si–O defect with a non-bridging oxygen atom and two threefold coordinated silicon atoms ≡Si can be identified. The ≡Si defects have a pyramidal structure, are separated from each other by a distance of 12.6 $\mathring{A}$ and are apparently associated with oxygen deficiency, since they represent two parts of an oxygen vacancy. Note that the presence of defect =Si–O does not violate the stoichiometry of this supercell, since the addition of this defect through its non-bridging oxygen atom to defect ≡Si again yields a defect of the same type, that is, the ≡Si defect. The supercell energy is about 20 eV higher than the energy of the supercell obtained from the melt stabilized at 4700 K for 6 ps.

C. Melting-quenching procedure applied to amorphous supercells with an oxygen vacancy. The optimized oxygen vacancy created in stoichiometric amorphous SiO_2 (see Tab. 2) is subjected to the melting-quenching procedure at the same melt stabilization temperature T_{melt}, that was used in the preparation of the original amorphous stoichiometric state.

All obtained states of amorphous supercells are diamagnetic – there are no unpaired electrons. After the melting-quenching procedure, oxygen vacancies with exotic conformations, created in the $2BO_3C$ defect in the amorphous state a-SiO_2-4500, are transformed into the usual type of oxygen vacancies in the form of a Si–Si bond. In addition to oxygen vacancies, either 2BOC or =Si–O are formed. In the 4500_2BOC-1 state, 5- and 3-membered rings also appear.

In the case of a-SiO_2-4600, there are one 2BOC defect, an oxygen vacancy in the form of Si–Si bond, and 3- and 5-membered rings. In the case of a-SiO_2-4700, there are three 2BOC defect, an oxygen vacancy in the form of Si–Si bond, and three 3- and one 5-membered rings.

After applying the melting-quenching procedure with melt stabilization for 6 ps to the amorphous state of a-SiO_2-4700_19ps obtained by stabilizing the melt at 4700 K for 19 ps with an oxygen vacancy, an amorphous state is obtained containing $2BO_3C$, Si_5 and =Si defects in addition to 2BOC centers and a large number of 3- and 4-membered rings, but without an oxygen vacancy. Here, only twofold coordinated silicon atom =Si can be considered as an oxygen deficient center (ODC). Indeed, the combination of $2BO_3C$ and Si_5 defects can obviously be converted into stoichiometry of defect-free SiO_2: $2BO_3C + Si_5 \rightarrow 2BOC + {\equiv}Si + Si_5 \rightarrow SiO_2 + {\equiv}Si{-}O + {\equiv}Si$, where a combination of ≡Si and ≡Si–O gives again stoichiometry of defect-free SiO_2.

The main results obtained for the oxygen vacancy are as follows:

1. Despite the dramatic change in some cases of the topology of the atomic network after the melting-quenching procedure, the bulk of silicon dioxide atoms form a network of SiO_4-

tetrahedra connected to each other by their vertices – one common vertex for two adjacent tetrahedra, as in silica glass.

2. An oxygen vacancy is a relatively stable point defect of silicon dioxide. When created in a cristobalite crystal, it retains the Si–Si bond form up to the melt stabilization temperature of 4700 K (for 6 and for 14 ps of melt stabilization). The formation energy of such a vacancy is about 8 eV.
3. At higher melt stabilization temperatures, the vacancy loses its form as a Si–Si bond and is transformed into two threefold coordinated Si atoms located far from each other. This may be due to the diffusion of silicon atoms.
4. The presence of an oxygen vacancy in the original crystal can even lead to a strong rearrangement of the topology of the atomic network after the melting-quenching procedure even at such a low melt stabilization temperature as 4600 K when a corresponding stoichiometric supercell retains the atomic network topology of the initial crystal.
5. The presence of an oxygen vacancy in the original crystal promotes formation of other defects as a result of obtaining an amorphous state using the melting-quenching procedure compared to the amorphous state of the stoichiometric supercell. These are the same defects that are presented in stoichiometric samples of amorphous silicon dioxide: 2BOC defects, threefold and fivefold coordinated silicon atoms, ≡Si and Si_5, =Si–O defects with a non-bridging oxygen atom, threefold coordinated oxygen atom O_3, as well as 3-, 4- and 5-membered strained rings.
6. Due to stochastic nature of the MD simulation, independent runs of the melting-quenching procedure can lead to amorphous states with different atomic network topology and different set of point defects.
7. ESR-active states with unpaired electrons have significantly higher energies than diamagnetic states in the studied amorphous states of SiO_2.

2.3. Two Separated in Space Oxygen Vacancies, OV+OV

A. Two separated in space oxygen vacancies are created in existing stoichiometric states. By creating two such oxygen vacancies in either crystalline or amorphous supercells and then optimizing the supercell energy, it is easy to show that in all cases the formation energy of two vacancies is approximately twice the formation energy 8 eV of one vacancy found above. The lowest excitation energy of electrons for supercells without other defects corresponds to the excitation energy of one oxygen vacancy obtained above. By creating two oxygen vacancies by removing two oxygen atoms from the $2BO_3C$ center present in a-SiO_2-4500, the following new defect is obtained. This defect consists of two adjacent oxygen vacancies sharing a silicon atom, which is linked to the rest of the atomic network by two Si–O bonds. Of the two other silicon atoms adjacent to the vacancies, one is also linked to the rest of the atomic network by two Si–O bonds and the second one is linked to the atomic network by three Si–O bonds and corresponding distances R(Si–Si) are equal to 2.36 and 2.46 $\AA$, respectively. A characteristic feature of this defect is the very low excitation energy of electrons, 0.22 eV.

B. Two separated oxygen vacancies in initial cristobalite + the melting-quenching procedure. The separated oxygen vacancies OV+OV are created near the center of the supercell of the initial cristobalite crystal, followed by the melting-quenching procedure to obtain amorphous states at different melt stabilization temperatures T_{melt} from 4600 to 4900 K. At T_{melt} = 4600 K, in the oxygen-deficient amorphous state, two separate oxygen vacancies of the Si–Si bond type are retained. At higher T_{melt}, higher energy amorphous states of the supercell (more than 16 eV) are obtained, in which there are either no oxygen vacancies at all, or there is only one oxygen vacancy, but there is a twofold coordinated silicon atom =Si and other defects that we have already considered. For example, the energy of the amorphous supercell obtained at T_{melt} = 4700 K, which contains one =Si defect, two ≡Si defects, and one 2BOC center, is 16 eV higher than the energy of the amorphous supercell obtained at T_{melt} = 4600 K, which contains only two oxygen vacancies. Comparing the energies of these amorphous supercells, we can conclude that the energy of formation of the =Si defect is significantly greater (by 10–12 eV) than the energy of formation of an oxygen vacancy.

It should be noted that independent runs of the melting-quenching procedure with the same parameters can result in amorphous states that differ greatly in energy and in the composition of point defects. For example, with one run of melting-quenching at T_{melt} = 4700 K of the initial crystal with two oxygen vacancies, a-SiO_2-4700 is obtained with the following point defects: one =Si, two ≡Si and one 2BOC. When re-starting the melting-quenching with the same T_{melt}, we obtain the amorphous state of a-SiO_2-4700-1 with a supercell energy exceeding the supercell energy of the state of a-SiO_2-4700 by 9.3 eV and containing a different set of defects: three ≡Si, one Si_5, one 2BOC, one $2BO_3C$ and one oxygen vacancy of a special type OV′, and in both amorphous states there are no unpaired electrons. The oxygen vacancy OV′ of the Si–Si bond type present here has a peculiarity: one of the silicon atoms adjacent to the vacant site has three bonds with neighboring oxygen atoms, and the second silicon atom has four Si–O bonds. Despite this feature, the Si–Si bond length in this vacancy is 2.44 $\mathring{A}$, which is close to the values of ordinary oxygen vacancies with two threefold coordinated neighboring silicon atoms. It is easy to verify that in the a-SiO_2-4700-1 state, the oxygen deficiency of two initial oxygen vacancies is maintained. Indeed, $2BO_3C \rightarrow 2BOC + \equiv Si$, where the 2BOC center does not violate the stoichiometry of SiO_2, and the ≡Si defect together with Si_5 also does not violate the stoichiometry of SiO_2; two ≡Si defects correspond to the stoichiometry of one oxygen vacancy, and the oxygen vacancy OV′ can be decomposed into an ordinary oxygen vacancy OV and a ≡Si–O defect, which together with the remaining third ≡Si defect gives the stoichiometry of defect-free SiO_2.

In both amorphous states of a-SiO_2-4700 and a-SiO_2-4700-1, 3-, 4- and 5-membered rings are present, which, like the 2-membered rings of 2BOC, do not violate the stoichiometry of SiO_2.

At T_{melt} = 4800 and 4900 K, even higher-energy amorphous oxygen-deficient states are obtained, in which various sets of the SiO_2 defects already described above are present.

C. Melting-quenching procedure applied to amorphous supercells with two separated oxygen vacancies. Let us see, what happens if supercells of some amorphous states (see Tab. 1) with a created two oxygen vacancies are again subjected to the melting-quenching procedure. For this purpose, four states with two oxygen vacancies were chosen: three of them are a-SiO_2-4600, a-SiO_2-4700 and a-SiO_2-4800 with two oxygen vacancies separated in space, and the fourth state is a-SiO_2-4500 with two vacancies created by removing two oxygen atoms from the $2BO_3C$ defect, which we designated as a-SiO_2-4500_2BOC. Here, in the melting-quenching procedure, all melts were stabilized within 6 ps. The energies of the supercells of all the obtained amorphous states are significantly higher than the energies of most of the amorphous states with a deficit of two oxygen atoms that we have previously considered, and basically they contain only the defects that we have already considered, but in two cases new defects were encountered.

After melting-quenching, the a-SiO_2-4500_2BOC system with two oxygen vacancies initially created in the $2BO_3C$ defect has no unpaired electrons, and it contains two twofold coordinated silicon atoms =Si, one =Si–O defect with a non-bridging oxygen atom, one Si_5, several 2BOC centers, 3-, 4- and 5-membered rings, and a center that we designate as Di-$2BO_3C$ – a double $2BO_3C$ center (Fig. 2).

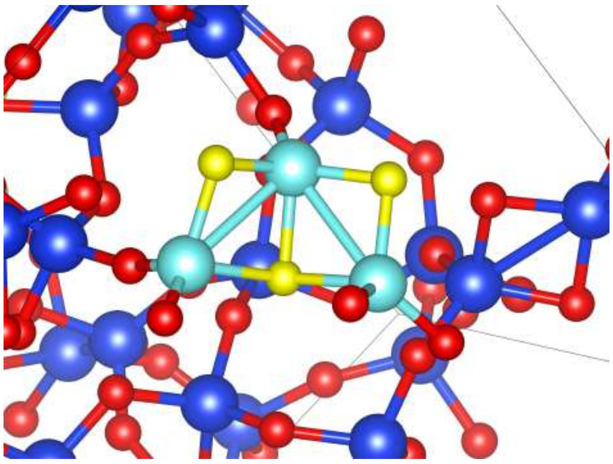


Figure 2. Defect Di-$2BO_3C$ found in a-SiO_2-4500; light blue and yellow balls are Si and O atoms, respectively, which make up the Di-$2BO_3C$ defect

The Di-$2BO_3C$ defect involves three silicon atoms and three oxygen atoms connecting them, and one of these three oxygen atoms, common to two $2BO_3C$ defects, forms three Si–O bonds with the neighboring Si atoms included in this defect. Note that the stoichiometry of the Di-$2BO_3C$ defect is equivalent to the stoichiometry of the threefold coordinated silicon atom ≡Si, which together with Si_5 does not violate the stoichiometry of SiO_2. The defect =Si–O does not violate the stoichiometry of defect-free SiO_2, since it can be broken down into an oxygen atom and a twofold coordinated silicon atom =Si, the stoichiometry of which is equivalent to the stoichiometry of an oxygen vacancy, adding to which the remaining oxygen atom, we obtain the stoichiometry of defect-free SiO_2. Thus, the stoichiometry of the considered amorphous state with two oxygen vacancies and after melting-quenching with a melt stabilization temperature of $T_{melt} = 4500$ K corresponds to two oxygen vacancies in the supercell, since the stoichiometry of two defects =Si is equivalent to the stoichiometry of two oxygen vacancies.

An independent run of melting-quenching at $T_{melt} = 4500$ K led to a new, slightly lower in energy (by 3.3 eV), amorphous state with a different set of intrinsic defects, among which there is again the Di-$2BO_3C$ defect, shown in Fig. 2.

Another previously unseen defect, designated by us as the 2BOC′ center, was obtained by applying the melting-quenching procedure to a-SiO_2-4800, in which two oxygen vacancies were created, separated apart in space. The 2BOC′ center is a 2BOC center in which one of the silicon atoms has not two Si–O bonds with the rest of the atomic network, but only one, i.e., it is a threefold coordinated silicon atom atom embedded in a 2BOC center. In this relatively high-energy amorphous state with two unpaired electrons, in addition to the two 2BOC′ centers, the following defects already discovered are also present: one oxygen vacancy OV, a =Si–O defect with a non-bridging oxygen atom, two Si_5 defects, two $2BO_3C$ defects, several 2BOC defects, and 3-, 4-, and 5-membered rings.

2.4. Oxygen di-Vacancy, OdV

The OdV defect can be created in a regular SiO_2-network, crystal or amorphous, by eliminating two oxygen atoms bound to the same Si atom. The oxygen deficiency is the same in both cases, OdV and two separated oxygen vacancies VO+VO: each supercell is missing two oxygen atoms.

A. The OdV defect is created in existing stoichiometric states. OdV is created in the cristobalite crystal and in a-SiO_2 prepared from this crystal using several melt stabilization temperatures T_{melt} (4600–4900 K): two oxygen atoms bonded to the same silicon atom are removed, and the energy of the supercell is optimized by varying positions of all atoms, form and volume of the supercell. E_{form}(OdV) is the formation energy of the OdV defect calculated as follows: $E_{form}(\text{OdV}) = E_{opt}(\text{OdV}) + 2\times E(\text{O}) - E_{opt}$, where E_{opt} is the optimized energy of the stoichiometric supercell (see Tab. 1), and $E(\text{O}) = -1.897$ eV is determined above just before Tab. 2.

In cristobalite, the OdV defect is in the configuration of two oxygen vacancies in the form of Si–Si bonds of the length 2.61 $\AA$ with a common Si atom, which is linked to the rest of the atomic network by only two Si–O bonds (Fig. 3).

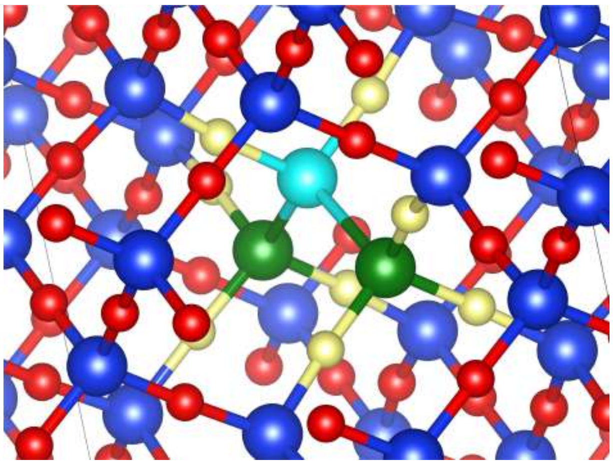


Figure 3. The OdV defect in cristobalite after the energy optimization

In Fig. 3, the light blue ball is a Si atom with two Si-O bonds; green balls are Si atoms with three Si-O bonds each, yellow balls are oxygen atoms bonded to blue and green Si atoms.

In addition to the restrictions imposed on the relaxation of the atoms of this defect by the atoms of its crystalline environment, each of the two neighboring threefold coordinated silicon atoms ≡Si restricts the relaxation of the third Si atom to the second neighboring threefold coordinated silicon atom. This is the reason why Si–Si distances in these two vacancies are noticeably larger than the R(Si–Si) = 2.48 $\AA$ in the single oxygen vacancy in cristobalite (see Tab. 2). In amorphous states, the Si–Si distances in the two adjacent oxygen vacancies are in the range 2.4–2.6 $\AA$. The formation energy of the OdV defect is close to twice the formation energy of a single oxygen vacancy (see Tab. 2).

Having created OdV in cristobalite and in defect-free a-SiO_2-4600 and a-SiO_2-4700 amorphous states, we determine the lowest energy of electronic excitations for these states E_g, equal to 4.20, 4.46 and 4.47 eV, respectively. These values are considerably lower E_g of a single oxygen vacancy and of defect-free amorphous systems, see Tab. 2 and Tab. 1, respectively. This indicates that the optical excitation energy of the OdV defect is significantly less than the excitation energy of a single oxygen vacancy. Taking into account the known underestimation of the E_g values in the PBE calculations [8], we can assume that our results qualitatively correspond to the assignment of the experimentally observed absorption bands in quartz glass at 7.6 eV and 5.0 eV to a single oxygen vacancy and to an oxygen divacancy OdV.

B. The OdV defect in initial cristobalite + the melting-quenching procedure. Let us consider what happens to the oxygen divacancy created in cristobalite after the melting-quenching amorphization procedure at different melt stabilization temperatures T_{melt}.

The initially created in the crystal oxygen divacancy OdV is not preserved as a result of the melting-quenching procedure at T_{melt} = 4600 K, and other defects arise in its place. Two fivefold coordinated silicon atoms Si_5 and a new diamagnetic defect OdV′ shown in Fig. 4 are formed.

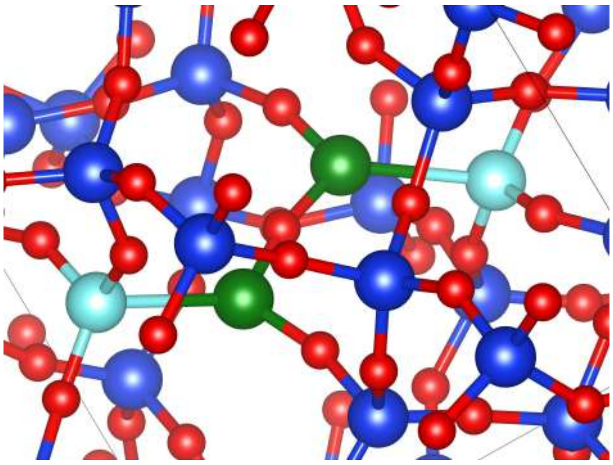


Figure 4. The OdV′ center appeared in a-SiO_2-4600 after the melting-quenching procedure from the initial β-cristobalite with OdV

In Fig. 4, the green balls are silicon atoms neighbors of oxygen vacancies which bound the vacancies by a Si–O–Si bridge; the light blue balls are silicon atoms that link oxygen vacancies to the rest of a-SiO_2 network; blue and red balls are fourfold and twofold coordinated Si and O atoms, respectively, forming the a-SiO_2 network.

The OdV′ defect consists of two adjacent oxygen vacancies linked to each other by a Si–O–Si bridge, in which each silicon atom, a part of Si–Si bond of OV, participates in only two Si–O bonds. For brevity, we denote this defect as modified oxygen divacancy OdV′. Topology of the atomic network has changed somewhat compared to the crystal – two three-membered rings have appeared.

When melting-quenching is repeated at T_{melt} = 4600 K, a state with two unpaired electrons is obtained, and its energy is higher than the previous state of a-SiO_2-4600 by approximately 15 eV. In this case, instead of two Si_5 defects and an OdV′ defect, there is an oxygen vacancy with R(Si-Si) = 2.39 $\AA$, two ≡Si defects with a pyramidal conformation, a =Si–O defect with a non-bridging oxygen atom, and two 2BOC centers. Of course, there are also 3-, 4- and 5-link rings. It is easy to verify that these defects do not change the degree of oxygen deficiency, which remains as it was before melting-quenching – one oxygen divacancy OdV per supercell. Indeed, by breaking two Si–Si bonds, the OdV′ defect can be represented as two twofold coordinated silicon atoms =Si and two threefold coordinated silicon atoms ≡Si; the combination of two ≡Si with two Si_5 gives a defect-free stoichiometry of SiO_2, so that two =Si defects remain, corresponding to an oxygen deficiency of two oxygen atoms per supercell.

In other amorphous states obtained at T_{melt} = 4700, 4800 and 4900 K, different sets of point defects described above are formed, but energies of these states are more than 13 eV greater than the energy of a-SiO_2-4600 with two Si_5 and OdV′. There are neither OdV, nor OdV′ in these high energy amorphous states, but other defects are presented. Most of these defects were described above, but a new defect =Si–O–Si= is found in a-SiO_2-4800, which is two threefold coordinated silicon atoms linked to each other by a common oxygen atom (Fig. 5). In this defect, silicon atoms are located so close to each other that a strong Si–Si bond is formed between them, the length of which is 2.19 $\mathring{A}$, and due to this approaching, the Si–O–Si angle of the oxygen bridge between these atoms is close to a right angle.

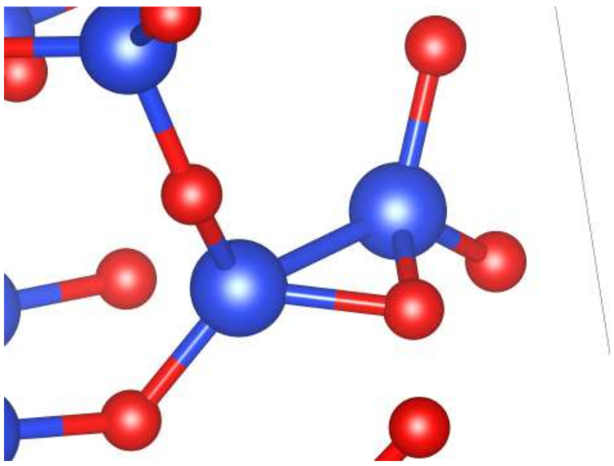


Figure 5. Defect =Si–O–Si= found in a-SiO_2-4800 with the oxygen deficiency of oxygen divacancy

This new defect can be interpreted as a 2BOC center without one oxygen bridge, but with the geometry of the remaining atoms almost like in the 2BOC center. In addition to the =Si–O–Si= defect, there are also OV, =Si–O, several 2BOC and 3-, 4- and 5-membered rings in such a-SiO_2-4800. The =Si–O–Si= defect can be obviously named as Oxygen Deficient Center (ODC) corresponding to the stoichiometry of an oxygen vacancy.

C. Melting-quenching procedure applied to amorphous supercells with OdV. When supercells of some amorphous states with a hand created OdV defect are again subjected to the melting-quenching procedure with T_{melt} = 4600, 4700 and 4800 K, new amorphous states are obtained with defects described above: =Si, OV, =Si–O, ≡Si–O, $2BO_3C$, Si_5, ≡Si, and OdV.

The main results obtained for the oxygen divacancy OdV are as follows. The oxygen divacancy OdV is unstable with respect to the melting-quenching procedure at all studied melt stabilization temperatures and decomposes into other defects. The lowest-energy of these defects is the OdV′ center, formed by two adjacent oxygen vacancies linked by a Si–O–Si bridge, in which each Si atom participates in only two Si–O bonds (Fig. 4). This defect is formed at T_{melt} = 4600 K. At higher melt stabilization temperatures, spatially separated oxygen vacancies, threefold coordinated silicon atoms ≡Si, and twofold coordinated silicon atoms =Si appear.

3. Discussion

Based on the conducted studies of numerous (more than 35) amorphous states of silicon dioxide obtained by the melting-quenching procedure from a crystal, the following main results can be summarized and discussed. The density of the obtained a-SiO_2 samples is close to the experimentally measured value of 2.2 g/cm^3 and is in the range 1.91–2.26 g/cm^3. Among all possible intrinsic point defects, the first to appear at the lowest melt stabilization temperatures T_{melt} are the 2BOC or 2BO_3C centers. This means that the formation energy of such defects is the lowest compared to other intrinsic defects in silica. The 2BOC defect is two SiO_4-tetrahedra linked by two oxygen vertices, and the 2BO_3C defect is a modification of it. This was first shown in stoichiometric a-SiO_2 using spin-unpolarized quantum molecular dynamics simulations in [22, 24] and confirmed in stoichiometric and oxygen-deficient a-SiO_2 using spin-polarized simulations in the present work. The number of such defects increases with increasing T_{melt}, but does not exceed several defects per supercell containing 192 atoms. The 2BOC centers do not violate stoichiometry of SiO_2.

For cristobalite and quartz crystals, the point defects and amorphization occur at T_{melt} between 4500 and 4700 K with the parameters of the melting-quenching procedure used: heating and cooling rates of 0.5 K/fs and melt stabilization of several picoseconds. The term “amorphization” refers to a change in the topology of the atomic network of crystalline SiO_2 with fourfold coordinated Si atoms and twofold coordinated oxygen atoms, which results in the formation of rings consisting of a small number (3, 4, 5) of Si–O–Si bridges. For a stishovite crystal with sixfold coordinated Si atoms and threefold coordinated oxygen atoms, amorphization occurs at significantly lower melt stabilization temperatures T_{melt} of about 3000 K, with the same parameters of the melting-quenching procedure. In general, the topology of atomic network of amorphous silica obtained from stishovite is similar to the topology of a-SiO_2 obtained from cristobalite or quartz.

With increasing time of the melting-quenching procedure, for example, with increasing time of melt stabilization, the critical value T_{melt}, above which amorphization occurs, decreases. The presence of vacancies or divacancies in the original crystal promotes the formation of point defects in the amorphous silicon dioxide obtained by melting-quenching and causes a change in the topology of the atomic network compared to the crystal at lower melt stabilization temperatures.

Despite the radical change in the topology of the atomic network during amorphization using melting-quenching, the bulk of the silicon dioxide atoms form a network of SiO_4-tetrahedra, connected to each other by their vertices - one common vertex for two adjacent tetrahedra, as in quartz glass. Most of the point defects - the local disruption of this network - found in the stoichiometric a-SiO_2 models are also present in oxygen-deficient a-SiO_2. The use of spin-polarized calculations in the melting-quenching procedure leads to the discovery of new point defects that are not obtained in spin-unpolarized calculations, in particular, ESR-active point defects with unpaired electrons. The energy of the diamagnetic states of a-SiO_2 is significantly lower than the energy of ESR-active states containing defects with unpaired electrons.

The following intrinsic point defects are found in stoichiometric a-SiO_2: 2BOC, 2BO_3C in which one oxygen atom of the 2BOC defect has a third Si–O bond linking it to the rest of the atomic network, the $\equiv$Si–O and =Si–O defects with non-bridging oxygen atoms, a threefold coordinated silicon atom $\equiv$Si, a fivefold coordinated silicon atom Si_5, a twofold coordinated silicon atom =Si, an oxygen vacancy OV and a threefold coordinated oxygen atom O_3. All these defects are also found in oxygen-deficient a-SiO_2.

In the present work, oxygen deficient a-SiO_2 was prepared by removing one or two oxygen atoms from a supercell containing 64 Si and 128 O atoms, combined with a melting-quenching procedure. In oxygen-deficient a-SiO_2, apart from defects of stoichiometric a-SiO_2, the following additional point defects were obtained: a 2BOC′ defect, in which one silicon atom has only three Si–O bonds, a Di-$2BO_3C$ defect, which is two combined 2BOC defects with one common silicon atom and one common oxygen atom, the latter forms three Si–O bonds with three silicon atoms that make up this defect, a =Si–O–Si= defect, which is two threefold coordinated silicon atoms linked to each other by a common oxygen atom, a modified oxygen vacancy OV′, in which one of the Si atoms adjacent to the vacant site has three bonds with neighboring oxygen atoms, and the second Si atom has four Si–O bonds, an oxygen divacancy OdV, a modified oxygen divacancy OdV′, and a –Si–O defect, which is a silicon atom with two Si–O bonds, one bond links this atom to the rest of atomic network and another bond links this silicon atom with a non-bridging oxygen atom. For clarity, we have presented all the defects we discovered in Tab. 4.

Table 4. Intrinsic defects discovered in various amorphous states of SiO_2 in our simulation

Silicon dioxide amorphous states	Point defects
Stoichiometric a-SiO_2	$2BO_3C$, ≡Si–O, Si_5, ≡Si, O_3, 2BOC, =Si, OV, =Si–O
Oxygen-deficient a-SiO_2, Single oxygen vacancy	OV, Si_5, ≡Si, O_3, 2BOC, $2BO_3C$, =Si, =Si–O, –Si–O
Oxygen-deficient a-SiO_2, Two spatially separated oxygen vacancies	OV, Si_5, ≡Si, 2BOC, $2BO_3C$, =Si, =Si–O, O_3, Di-$2BO_3C$, 2BOC′, OV′
Oxygen-deficient a-SiO_2, Single oxygen divacancy	OV, OdV, Si_5, ≡Si, 2BOC, $2BO_3C$, =Si, =Si–O, ≡Si–O, OdV′, =Si–O–Si=

Comparing the sets of defects in different rows of Tab. 4, we can conclude that the following 16 point defects are found in oxygen-deficient a-SiO_2: nine defects 2BOC, $2BO_3C$, ≡Si–O, =Si–O, ≡Si, =Si, Si_5, OV, O_3, which are also found in stoichiometric a-SiO_2, and seven defects –Si–O, Di-$2BO_3C$, 2BOC′, =Si–O–Si=, OV′, OdV, and OdV′, which are found only in oxygen-deficient a-SiO_2. Not all of these sixteen defects are associated with local oxygen deficiency. The Si_5, 2BOC, and ≡Si–O defects present in our models of stoichiometric and oxygen-deficient a-SiO_2 are obviously not oxygen-deficient centers. The =Si–O defect is also not an oxygen-deficient center. Indeed, the =Si–O defect can be decomposed into the twofold coordinated silicon atom =Si and an oxygen atom, the stoichiometry of =Si is equal to the stoichiometry of an oxygen vacancy, which in turn combines with an oxygen atom, resulting in the formation of stoichiometric silicon dioxide. Consequently, we can conclude that we have discovered the following 12 intrinsic point defects in a-SiO_2 associated with oxygen deficiency: OV, OV′, =Si, ≡Si, OdV, OdV′, =Si–O–Si=, $2BO_3C$, Di-$2BO_3C$, 2BOC′, O_3 and –Si–O. Note that the latter, together with the ≡Si defect, produces a twofold coordinated silicon atom =Si, which also represents an oxygen-deficient center, corresponding in stoichiometry to the oxygen vacancy OV. Also, the stoichiometry of the O_3 defect coincides with the stoichiometry of the ≡Si defect, and therefore the O_3 defect can be considered to be associated with oxygen deficiency, just like the ≡Si defect. To avoid confusion, we emphasize that in existing publications, oxygen-deficient centers (ODCs) are non-paramagnetic defects with specific optical characteristics, and until now, either an oxygen vacancy OV or a twofold coordinated silicon atom =Si have been considered as models of these centers. In this paper, we take a broader approach and refer to any defects caused by local oxygen deficiency as oxygen-deficient centers.

In this study, we did not detect the so-called peroxy bridge (linkage) in any amorphous state. This defect has long been discussed in the scientific literature as a precursor to the radiation-induced paramagnetic peroxy radical [4, 7, 9, 20]. In [24], this defect was discovered in the amorphous state of a-SiO_2-5000, obtained by simulating the melting-quenching of an alpha-quartz crystal using spin-unpolarized calculations. Clarification of the reasons for its absence in the amorphous states obtained in this work requires further research.

An estimate of the formation energy of a twofold coordinated silicon atom =Si is given, according to which the formation energy of this defect is significantly greater, by 10–12 eV, than the formation energy of an oxygen vacancy, equal to 8 eV. Moreover, a comparison of the supercell energies of all the amorphous states we studied showed that twofold coordinated silicon atoms =Si are present only in high-energy states along with other defects.

The total CPU time for the entire melting-quenching procedure at $T_{melt} = 4700$ K is 11.22 days on the Lomonosov-2 supercomputer. In total, more than 35 melting-quenching procedures were carried out during the present research, which took more than 1 year CPU time on Lomonosov-2.

Conclusion

The conducted modeling revealed the configurations of possible intrinsic point defects in amorphous silicon dioxide, and along with the known ones, oxygen vacancies, twofold coordinated silicon atoms =Si, E′-centers and NBOHC, new defects were also identified, including defects associated with oxygen deficiency. A total of twelve defects associated with local oxygen deficiency were identified. Many of the defects found have electronic states in the band gap of silicon dioxide, which can lead to absorption bands in the transparency window of defect-free SiO_2, including the visible and near-IR ranges. Such defects can significantly affect the laser damage threshold of optical coatings using amorphous silicon dioxide layers, as well as radiation-induced absorption in silica-based optical fibers and leakage currents in various semiconductor devices using silicon dioxide as insulating layers. The defects found can become a broad basis for interpreting experimental data from optical and ESR measurements. Many new defects described in the present work for the first time are found due to using quantum molecular dynamics through the whole melting-quenching procedure in the spin-polarized approach. A more accurate description of the optical characteristics of the detected defects requires separate studies, including other more accurate methods for calculating electron excitations. The simulation technique used in this work can also be applied to model a wide range of impurity defects in silicon dioxide, which play an important role in both fiber optics and multilayer optical coatings.

Acknowledgments

The study was conducted under the state assignment of Lomonosov Moscow State University.

The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University, including the Lomonosov-2 supercomputer [29].

References

1. Dianov, E.M., Kornienko, L.S., Nikitin, E.P., *et al.*: Radiation-optical properties of quartz glass fiber-optic waveguides (review). Soviet Journal of Quantum Electronics 13(3), 274 (1983). https://doi.org/10.1070/QE1983v013n03ABEH004145

2. DiStefano, T.H., Eastman, D.E.: The band edge of amorphous SiO_2 by photoinjection and photoconductivity measurements. Solid State Communications 9(24), 2259–2261 (1971). https://doi.org/10.1016/0038-1098(71)90643-0

3. Feigl, F.J., Fowler, W., Yip, K.L.: Oxygen vacancy model for the $E_1{}'$ center in SiO_2. Solid State Communications 14(3), 225–229 (1974). https://doi.org/10.1016/0038-1098(74)90840-0

4. Friebele, E.J., Griscom, D.L., Stapelbroek, M., Weeks, R.A.: Fundamental defect centers in glass: The peroxy radical in irradiated, high-purity, fused silica. Physical Review Letters 42, 1346–1349 (1979). https://doi.org/10.1103/PhysRevLett.42.1346

5. Griscom, D.L.: Defect structure of glasses: Some outstanding questions in regard to vitreous silica. Journal of Non-Crystalline Solids 73(1), 51–77 (1985). https://doi.org/10.1016/0022-3093(85)90337-0

6. Griscom, D.L.: A minireview of the natures of radiation-induced point defects in pure and doped silica glasses and their visible/near-IR absorption bands, with emphasis on self-trapped holes and how they can be controlled. Physics Research International 2013(1), 379041 (2013). https://doi.org/10.1155/2013/379041

7. Griscom, D.L., Mizuguchi, M.: Determination of the visible range optical absorption spectrum of peroxy radicals in gamma-irradiated fused silica. Journal of Non-Crystalline Solids 239(1), 66–77 (1998). https://doi.org/10.1016/S0022-3093(98)00721-2

8. Hafner, J.: *Ab-initio* simulations of materials using VASP: Density-functional theory and beyond. Journal of Computational Chemistry 29(13), 2044–2078 (2008). https://doi.org/10.1002/jcc.21057

9. Kitagawa, I., Maruizumi, T.: New intrinsic pair defects in silicon dioxide interface. Applied Surface Science 216(1), 264–269 (2003). https://doi.org/10.1016/S0169-4332(03)00379-9

10. Kresse, G., Furthmüller, J.: Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. Physical Review B 54, 11169–11186 (1996). https://doi.org/10.1103/PhysRevB.54.11169

11. Kresse, G., Joubert, D.: From ultrasoft pseudopotentials to the projector augmented-wave method. Physical Review B 59, 1758–1775 (1999). https://doi.org/10.1103/PhysRevB.59.1758

12. Marsman, M., Paier, J., Stroppa, A., Kresse, G.: Hybrid functionals applied to extended systems. Journal of Physics: Condensed Matter 20(6), 064201 (2008). https://doi.org/10.1088/0953-8984/20/6/064201

13. Shiga, M., Hirata, A., Onodera, Y., Masai, H.: Ring-originated anisotropy of local structural ordering in amorphous and crystalline silicon dioxide. Communications Materials 4(1), 91 (2023). https://doi.org/10.1038/s43246-023-00416-w

14. Skuja, L.: Optical properties of defects in silica. In: Pacchioni, G., Skuja, L., Griscom, D.L. (eds.) Defects in SiO_2 and Related Dielectrics: Science and Technology. pp. 73–116. Springer Netherlands, Dordrecht (2000). https://doi.org/10.1007/978-94-010-0944-7_3

15. Skuja, L., Tanimura, K., Itoh, N.: Correlation between the radiation-induced intrinsic 4.8 eV optical absorption and 1.9 eV photoluminescence bands in glassy SiO_2. Journal of Applied Physics 80(6), 3518–3525 (1996). https://doi.org/10.1063/1.363224

16. Skuja, L.: Optically active oxygen-deficiency-related centers in amorphous silicon dioxide. Journal of Non-Crystalline Solids 239(1), 16–48 (1998). https://doi.org/10.1016/S0022-3093(98)00720-0

17. Skuja, L., Kajihara, K., Grube, J., Hosono, H.: Luminescence of non-bridging oxygen hole centers in crystalline SiO_2. AIP Conference Proceedings 1624(1), 130–134 (2014). https://doi.org/10.1063/1.4900468

18. Skuja, L., Kajihara, K., Hirano, M., Hosono, H.: Visible to vacuum-UV range optical absorption of oxygen dangling bonds in amorphous SiO_2. Physical Review B 84, 205206 (2011). https://doi.org/10.1103/PhysRevB.84.205206

19. Skuja, L., Streletsky, A., Pakovich, A.: A new intrinsic defect in amorphous SiO_2: Twofold coordinated silicon. Solid State Communications 50(12), 1069–1072 (1984). https://doi.org/10.1016/0038-1098(84)90290-4

20. Sokolov, V.O., Sulimov, V.B.: Semi-empirical calculation of peroxy linkage in vitreous silicon dioxide. Physica Status Solidi (B) 142(1), K7–K12 (1987). https://doi.org/10.1002/pssb.2221420134

21. Strand, J., Kaviani, M., Gao, D., *et al.*: Intrinsic charge trapping in amorphous oxide films: status and challenges. Journal of Physics: Condensed Matter 30(23), 233001 (2018). https://doi.org/10.1088/1361-648X/aac005

22. Sulimov, A.V., Kutov, D.C., Grigoriev, F.V., *et al.*: Generation of amorphous silicon dioxide structures via melting-quenching density functional modeling. Lobachevskii Journal of Mathematics 41(8), 1581–1590 (2020). https://doi.org/10.1134/S1995080220080193

23. Sulimov, V.B., Sokolov, V.O., Dianov, E.M., Poumellec, B.: Photoinduced structural transformations in silica glass: the role of oxygen vacancies in the mechanism of formation of refractive-index gratings by UV irradiation of optical fibres. Quantum Electronics 26(11), 988 (1996). https://doi.org/10.1070/QE1996v026n11ABEH000857

24. Sulimov, V., Kutov, D., Sulimov, A., *et al.*: Density functional modeling of structural and electronic properties of amorphous high temperature oxides. Journal of Non-Crystalline Solids 578, 121170 (2022). `https://doi.org/10.1016/j.jnoncrysol.2021.121170`

25. Sulimov, V., Sokolov, V.: Cluster modeling of the neutral oxygen vacancy in pure silicon dioxide. Journal of Non-Crystalline Solids 191(3), 260–280 (1995). `https://doi.org/10.1016/0022-3093(95)00293-6`

26. Sulimov, V., Kutov, D., Sulimov, A., *et al.*: Study of the anomalous behavior of the a-HFO_2 refractive index with increasing Si doping by quantum molecular dynamics simulation. Journal of the Optical Society of America B 40(10), 2643–2649 (2023). `https://doi.org/10.1364/JOSAB.500520`

27. Sulimov, V.B., Sushko, P.V., Edwards, A.H., *et al.*: Asymmetry and long-range character of lattice deformation by neutral oxygen vacancy in α-quartz. Physical Review B 66, 024108 (2002). `https://doi.org/10.1103/PhysRevB.66.024108`

28. Sushko, P.V., Mukhopadhyay, S., Mysovsky, A.S., *et al.*: Structure and properties of defects in amorphous silica: new insights from embedded cluster calculations. Journal of Physics: Condensed Matter 17(21), S2115 (2005). `https://doi.org/10.1088/0953-8984/17/21/007`

29. Voevodin, V.V., Antonov, A.S., Nikitenko, D.A., *et al.*: Supercomputer Lomonosov-2: Large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). `https://doi.org/10.14529/jsfi190201`

30. Weinberg, Z.A., Rubloff, G.W., Bassous, E.: Transmission, photoconductivity, and the experimental band gap of thermally grown SiO_2 films. Physical Review B 19, 3107–3117 (1979). `https://doi.org/10.1103/PhysRevB.19.3107`

31. Yuan, X., Cormack, A.N.: Efficient algorithm for primitive ring statistics in topological networks. Computational Materials Science 24(3), 343–360 (2002). `https://doi.org/10.1016/S0927-0256(01)00256-7`

DOI: 10.14529/jsfi250408

Characterization of the Role of Amino Acid Residues in the 2-Hydroxybiphenyl 3-Monooxygenase Catalysis Based on Bioinformatic Analysis of the Flavin-dependent Monooxygenases and Supercomputer Modeling of the Structure of Mobile Fragments Applying Variational Autoencoders

Kirill E. Kopylov[1,2], *Maxim A. Shchepetov*[2,3], *Vytas K. Švedas*[1,2,3]

By modeling of predominant conformations of mobile loops in previously unresolved regions of 2-hydroxybiphenyl 3-monooxygenase structure (PDB ID: 5BRT) using GPU-accelerated metadynamics simulations integrated with artificial intelligence and high-performance computing the full-length protein model was built. Combined with bioinformatic analysis of the flavin-dependent monooxygenases it allowed to propose the functional role of amino acid residues in the 2-hydroxybiphenyl 3-monooxygenase catalysis. Three subfamily-specific residues Glu359, Lys339, Arg360 and the Asp332 residue, conservative throughout the entire family of flavin-dependent monooxygenases, form salt bridges Glu359-Lys339 and Arg360-Asp332, which stabilize alpha helices preserving the integrity of the Rossmann fold of the FAD-binding domain; subfamily-specific residues Trp338 and Glu359 provide the correct positioning of alpha-helices by interacting with two conservative residues Asp557 and Arg555 from the hydroxylase domain.

NAD binding pocket is formed by a number of subfamily-specific residues Trp38, Ser40, Ser42, Arg46, Ser47, Ala180, Asn205, Ser291, Trp293 located in an elongated pocket adjacent to the FAD binding site. The Asp313 residue, conservative in the entire family of flavin-dependent monooxygenases, directly interacts with FAD through hydrogen bonding with 2'-OH-ribitol, contributing to the binding and orientation of the cofactor. The Arg46, Ser47, Gly202, Ser203, Asn205, Arg242, Val253, Trp293, Met321, and Pro320, conservative for the entire family, play a crucial role forming the substrate binding site. The binding of cofactors and substrate in a quaternary complex and their orientation due to interactions with subfamily-specific positions Arg46, Ala180, His181 and Trp293 allows to perform the hydride transfer to the substrate stereospecifically. The triple stacking interaction between the FAD isoalloxazine ring, NADH nicotinamide ring and the subfamily-specific residue Trp293 leads to the formation of a highly stable charge-transfer complex and preferential Pro-S position in 2-hydroxybiphenyl 3-monooxygenase catalysis.

Keywords: flavin-dependent monooxygenases, 2-hydroxybiphenyl 3-monooxygenase from Pseudomonas azelaica, mobile loop structure prediction, full-length protein modeling, bioinformatics analysis, functional amino acid residues.

Introduction

Establishing the relationship between the structure and function of proteins, and enzymes in particular, is one of the most important tasks of modern biology. Despite the development of experimental techniques in structural biology (e.g., X-ray crystallography, NMR and cryo-EM), currently the PDB database [`https://rcsb.org`, `https://wwpdb.org` [3]] contains information on experimentally determined structures of 246,005 proteins (1,068,557 structures are presented in a separate section of Calculated Structure Models), although Uniprot [`https://uniprot.org` [1]] contains 199,006,239 amino acid sequences (mainly with unknown function, whereas

[1]Lomonosov Moscow State University, Belozersky Institute of Physicochemical Biology, Moscow, Russia
[2]Lomonosov Moscow State University, Research Computing Center, Moscow, Russia
[3]Lomonosov Moscow State University, Faculty of Bioengineering and Bioinformatics, Moscow, Russia

annotated are 573,661), and effective approaches need to be developed to bridge the gap between these information arrays.

Expectations of a breakthrough in this area are associated with the use of computer modeling and artificial intelligence. Indeed, the development of methods for predicting protein structure based on a known amino acid sequence has allowed to propose multiple spatial models (more than 200 million structures in the AlphaFold Protein Structure Database [`https://alphafold.ebi.ac.uk/`, [46]]), but the success of prediction is largely due to the presence of homologues of the protein under study and similar structures in training sets. The complexity of the problem also lies in the fact that many experimentally determined protein structures contain unresolved regions. Therefore, one of the not yet solved difficult problems is the prediction of the state of mobile elements, including loops, in the protein structure. The presence of such "blank spots" in the available information greatly complicates the search for the relationship between a full-length structure and a function. The mobile elements of the structure play an important role in the functioning of the protein/enzyme, ensuring its conformational plasticity and controlling access of the regulatory ligand or substrate to the complementary binding site, on the one hand, as well as their retention in a bound state, isolation of the formed complex from the environment and ensuring the specific microenvironment in the complex.

Since among the experimentally determined structures accumulated in the Protein Data Bank (PDB) over half contain unresolved segments (that often represent loops [38, 48]), accurate modeling of their structure is of high importance for preparing full-length protein models. Even in the post-AlphaFold era loop positioning remains one of the most challenging yet indispensable tasks in full-length protein modeling. Loops are often located on the protein surface, exhibit dynamic movement and adopt multiple distinct conformations that are not represented in PDB structures or adequately predicted [2, 30]. Accounting for loop dynamics has been recognized to be a key issue studying protein-ligand as well as protein-protein or antigen-antibody interaction and allosteric regulation, protein design, etc. [14, 16, 24, 29, 32]. Loop dynamics in enzyme structure has been recognized as an extremely important factor for catalytic activity, specificity as well as stability, evolution and design of improved enzyme variants [5, 7, 8, 25–28, 31, 49]. Flexible loop modeling using computational approaches in combination with artificial intelligence can provide an essential progress studying loop structural organization and their flexibility (appear to be open direction for exploration). Separate publications on the use of machine learning methods on this topic began to appear starting in the mid-80s, but a noticeable increase in research activity has been observed over the past 20 years (Fig. 1). These approaches allow to solve a number of problems and indicate the promise of supercomputer molecular modeling methods combined with the use of artificial intelligence approaches [21, 45].

Flavin-dependent monooxygenases are involved in a wide range of biological processes often playing a key role in the catabolism of natural and anthropogenic compounds or participating in the biosynthesis of numerous physiologically active compounds like hormones, vitamins and antibiotics [33]. Enzymes of this superfamily can be used as catalysts for large scale practical applications starting from lignin degradation or detoxification of xenobiotic compounds to the biocatalytic synthesis of key intermediates in pharmaceutical industry [9, 15]. The detailed mechanistic and structural studies of the FMO family enzymes are therefore of fundamental and practical interest. In this work the role of key amino acid residues in the catalytic mechanism of 2-hydroxybiphenyl 3-monooxygenase, a flavoprotein from *Pseudomonas azelaica* was analyzed

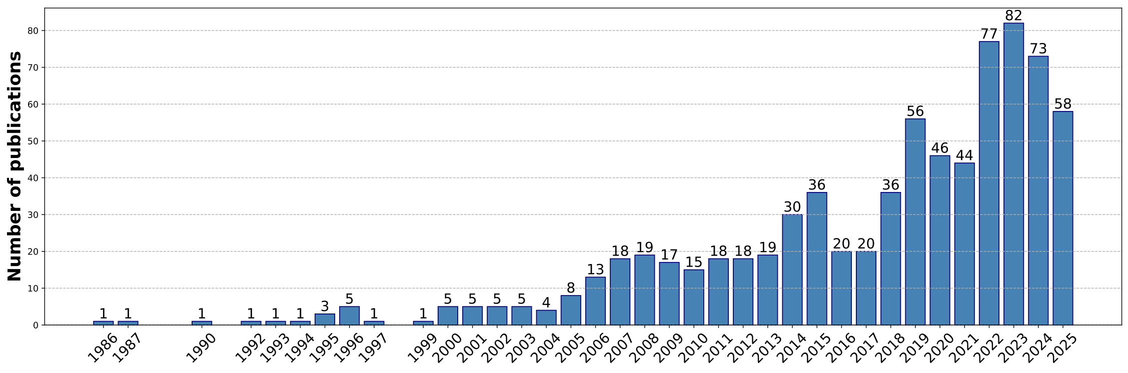


Figure 1. Publications on the topics of machine learning / artificial intelligence / neural networks and protein unresolved/non resolved/unstructured/non structured region modelling (PubMed)

based on the bioinformatics analysis of flavin-dependent monooxygenase family enzymes as well as supercomputer molecular modeling applying methods of artificial intelligence.

1. Materials and Methods

1.1. Bioinformatic Analysis

Bioinformatic analysis of a large representative set of flavin-dependent monooxygenases was used to study their structural and sequence variability by implementing a combination of protein sequence and structure comparison algorithms to account for structural and functional variability within this superfamily. The multiple structure-guided sequence alignment was constructed using the *Mustguseal* web-server (`https://biokinet.belozersky.msu.ru/mustguseal`) [42]. The server sequentially utilizes several bioinformatics algorithms: structural similarity search is used to find evolutionarily distant related proteins that have acquired functional/regulatory diversity due to significant changes in amino acid sequences and structures during natural selection (step 1); structural alignment to superpose the identified evolutionarily distant proteins that represent different families within a large superfamily (step 2); amino acid sequence similarity search to find evolutionarily close proteins, representatives of the selected families, and subsequent amino acid sequence alignment to superpose the sequences of evolutionarily close proteins (representatives of the same family, step 3). In the final step, the alignment of available structures of evolutionarily distant homologs is used as a framework for aligning the amino acid sequences of all collected representatives of the superfamily (step 4). Thus the selected PDB structures and their corresponding complete amino acid sequences were used to perform a structural alignment. The obtained structural core alignment was further used by the Mustguseal web-server to construct a larger alignment by incorporating all available sequences of related proteins. Each representative protein was used as a query for a sequence similarity search in Swiss-Prot and TrEMBL databases to collect its evolutionary close relatives. The obtained sequence sets were further filtered using the default parameters to remove redundant entries (at the 95% pairwise sequence identity threshold) within each family and superimposed using the core structural alignment of the representative proteins as a guide. The final structure-guided sequence alignment contained 4417 sequences and structures of monooxygenases with high structural, but low sequence similarity to 2-hydroxybiphenyl

3-monooxygenase from *Pseudomonas azelaica*. This representative set of homologs was automatically clustered into groups by maximizing sequence conservation within the groups and sequence variability between the groups using the *Zebra2* web-server (`https://biokinet.belozersky.msu.ru/zebra2`) [40]. The predicted clusters corresponded to different subfamilies of flavin-dependent monooxygenases that can be further analyzed according to the available functional annotation of members in each predicted group. Web-servers *pocketZebra* (`http://biokinet.cmm.msu.ru/index.php/pocketzebra`) and *visualCMAT* (`http://biokinet.cmm.msu.ru/visualcmat`) were used to find the NADH binding site and evaluate the function of residues in this site [39, 42].

1.2. Molecular Modeling

Neural network models with the architecture of hyperspherical variational autoencoders (S-VAE) were used as a component in modelling of the position of mobile loops. A critical choice in VAE design is the structure of the latent space, which significantly affects the model's capacity to represent underlying data properties [19]. For systems described by dihedral or torsional angles, such as molecular structures, a hyperspherical latent space offers an elegant solution [10]. The autoencoder was trained on the set of dihedral angles of unresolved loops, obtained by molecular dynamics. Dihedral spaces represent angular relationships between atoms in a molecular system. These angles, ranging from $[0, 2\pi)$, are inherently periodic. The periodicity implies that a dihedral space is better represented as a flat torus. The mathematical relationship between flat tori and hyperspheres further substantiates this choice. Embedding a flat torus in a hypersphere allows the model to naturally preserve properties such as smoothness and periodicity, ensuring that adjacent angular values in the data space (e.g., $2\pi - \epsilon$ and $0 + \epsilon$) remain adjacent in the latent representation [20]. The hypersphere inherently respects the periodic boundary conditions of these angles. This feature contrasts with Euclidean latent spaces, which are not naturally suited to handle periodicity or modular arithmetic, often leading to discontinuities or distortions in the latent representation.

The autoencoder is based on multiple narrowing convolutional layers and uses reparameterization of the three-dimensional von Mises-Fisher distribution. A weighted sum of the cumulative data reconstruction function (cosine distance) and Kullback–Leibler divergence is used as a metric for the error function. KL divergence assumes the hyperspherical nature of the latent space being calculated between the determined three-dimensional von Mises–Fisher distribution and a two-dimensional uniform hyperspherical distribution. The coefficient for accounting for KL divergence in the error function is 1.2, and the learning rate is $1 \cdot 10^{-2}$. The AdamW optimizer was used to train the model during 500 epochs. To find the optimal structure of protein sites, GPU accelerated metadynamics simulations were performed, using the Amber 20 molecular dynamics package [35] in conjunction with the Plumed 2.9 metadynamics package [12, 43]. The collective variable was defined using the `PYTORCH_MODEL` function [4], which consisted of compiled JIT PyTorch model (encoding part of the S-VAE) and the corresponding set of dihedral angles. Metadynamics was performed on three outputs of the model using a "well-tempered" scheme. The initial height of the Gaussian was set to 6.0 kJ/mol, the width to 0.06 for each measurement. A new potential was added every 400 steps. The bias factor was set to 10.0 and the simulation temperature to 300 K. For one computational experiment, 10 tasks ("walkers") were run simultaneously with a common metadynamics potential. All classical parameters for molecular dynamics modeling were as described above.

1.3. Mobile Loop Structure Prediction

The AlphaFold 2 model [17] was used to predict the structure of missing sites. An enzyme sequence from the UniProt database with the identifier O06647 was provided to the model. The model was run with standard settings in the "monomer" mode, producing 5 predicted structures using 5 different models from AlphaFold.

Rosetta [37] and MODELLER [13] were used as well for modeling of the missing sections in the 5BRT [18] structure. Using the MODELLER software, a Gln195-Ser201 segment from the 6EM0 [6] crystal structure was inserted into the original structure. Using Rosetta Remodel, a fully atomic model with the starting positions of the loops from the monomer of the protonated original structure was created. All protein atoms that were not included in the modeled regions were fixed in their positions, and the method of preserving the initial protein's protonation state was used. In the next step, 100 models of the 2-hydroxybiphenyl 3-monooxygenase monomer with alternative loop positions were generated using the Rosetta LoopModel. The coordinates of FAD and 2-hydroxybiphenyl were added to the model as user-defined residues. Next Generation KIC method was used to model 10000 loop conformations. The top 10 structures were selected based on Rosetta energy score and conformational reasonableness.

2. Results and Discussion

2.1. Bioinformatic Analysis

A search for homologous enzymes was performed using the CATH superfamily database [11]. The 2-hydroxybiphenyl 3-monooxygenase belongs to the CATH 3.50.50.60 superfamily. Since the CATH database is updated with a delay, a structural similarity search in the PDB database was additionally used to search for new homologues of the 2-hydroxybiphenyl 3-monooxygenase enzyme using the superpose algorithm (PDBeFOLD service) [23]. For comparative analysis of specific proteins, pairwise structural alignments were constructed using the Combinatorial Extension method [36]. The PDB structure 5BRT of 2-hydroxybiphenyl 3-monooxygenase, a flavoprotein from *Pseudomonas azelaica*, was used as a query for a structure similarity search in the PDB database to collect a nonredundant set of 16 proteins which shared 20% pairwise sequence similarity with 5BRT and represented different families within the superfamily (8FHJ,2R0G,3IHG,4K2X,2DKH,6UI5,4ICY,7YJ0,1PN0,6AIN,8JQP,3GMB,5EOW,6SW2,5TUI). Bioinformatic analysis revealed six distinct subfamilies of enzymes within the flavin-dependent monooxygenase family. The number of identified subfamilies corresponds well to the classification of external flavoprotein monooxygenases developed two decades ago on the basis of amino acid sequence, tertiary structure and cofactor preference when the 309 annotated bacterial genomes available at that time in the NCBI (`http://www.ncbi.nlm.nih.gov/BLAST/`) were screened for the presence of monooxygenase homologs using the prototype protein sequence and the BLAST tool. Review by Van Berkel W.J., Kamerbeek N.M. and Fraaije M.W. provided an inventory of known flavoprotein monooxygenases belonging to these different subclasses and highlighted the biocatalytic potential of this family of enzymes [44].

Currently, thanks to the accumulation of vast amounts of structural data, including even unannotated protein sequences, it has become possible to investigate distant evolutionary relationships within protein superfamilies and identify key amino acid residues critical to the entire family, as well as variable residues responsible for functional divergence and being specific to subfamilies of homologous enzymes. Thus, using bioinformatics analysis of protein families, it is possible to identify amino acid residues conserved across the entire family or only within individual subfamilies (subfamily-specific positions in multiple structure-based sequence alignments) [40]. As a result, 21 amino acid residues were identified as conservative ones for the entire family (eight Gly residues in positions 13, 15, 18, 62, 176, 179, 312, 325, three Leu residues in positions 25, 336, 340, three Asp residues in positions 178, 313, 332, two Ala185 and Ala314, one residue of Ser182, Arg307, His316, Pro320 and Tyr356) (Fig. 2). These residues are located in the most structurally conservative FAD/NAD binding domain D1.

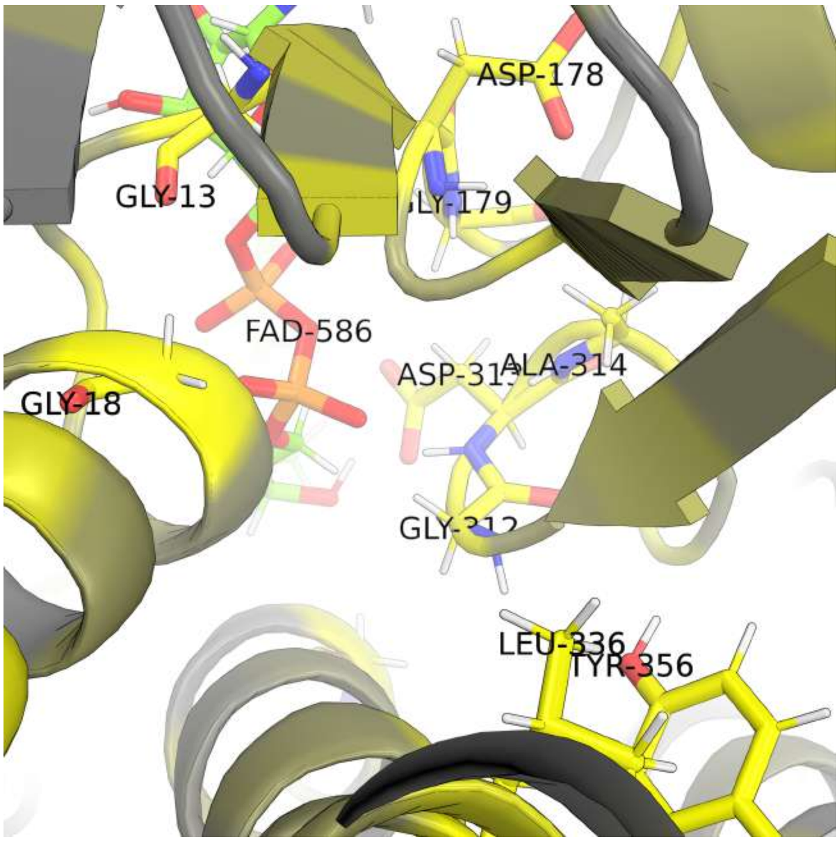


Figure 2. Conservative positions in the FAD-binding domain

2.2. Functional Role of Amino Acid Residues Identified by Bioinformatics Analysis. Binding of Substrates and Cofactors

Since the focus of this article is on the role of amino acid residues in the structure and catalytic mechanism of 2-hydroxybiphenyl 3-monooxygenase from *Pseudomonas azelaica*, we will discuss the information obtained from bioinformatic analysis in relation to this protein (Fig. 3).

Bioinformatic analysis takes into account the evolutionary relationships in protein families, and amino acid residues conservative in subfamilies may indicate functionally important, but variable positions in the protein structure.

The most typical for the entire family is a FAD binding subsite. On the one side, this area is bounded by two alpha-helices. Three subfamily-specific positions (Glu359, Lys339, Arg360) and the Asp332 residue, conservative throughout the entire family, form two structurally important salt bridges Glu359-Lys339 and Arg360-Asp332, which stabilize alpha helices in the dynamic

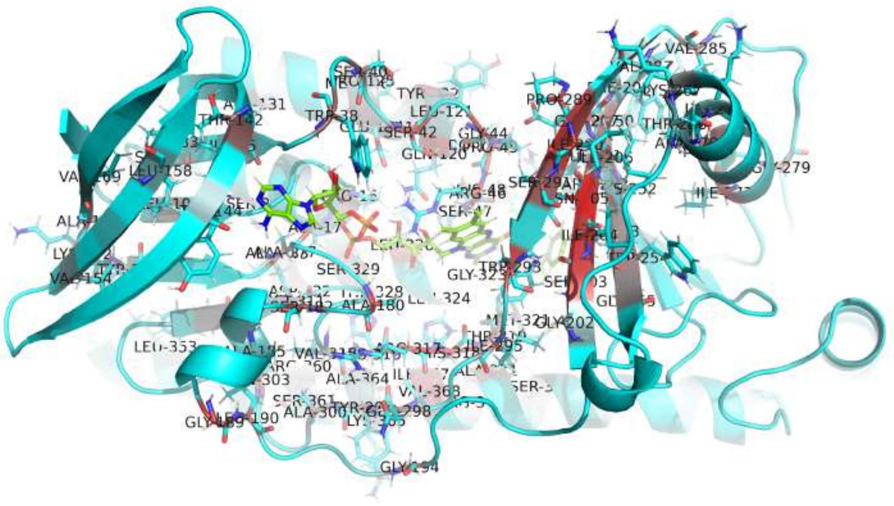


Figure 3. The subfamily-specific positions shown in 2-hydroxybiphenyl 3-monooxygenase from *Pseudomonas azelaica*

protein structure, preserving the integrity of the Rossmann fold of the FAD-binding domain D1 (Fig. 4). Such salt bridges are present in most subfamilies, however, in some cases Arg360 is replaced by Cys or Met, Lys339 by Cys, Ala or Glu, and Glu359 by Asn or Thr. One more subfamily-specific position, Trp338, participates in the correct positioning of these alpha-helices by interacting with the Asp557 along with the additional interaction of Glu359 with the Arg555 residue, i.e., two conservative residues from the hydroxylase domain.

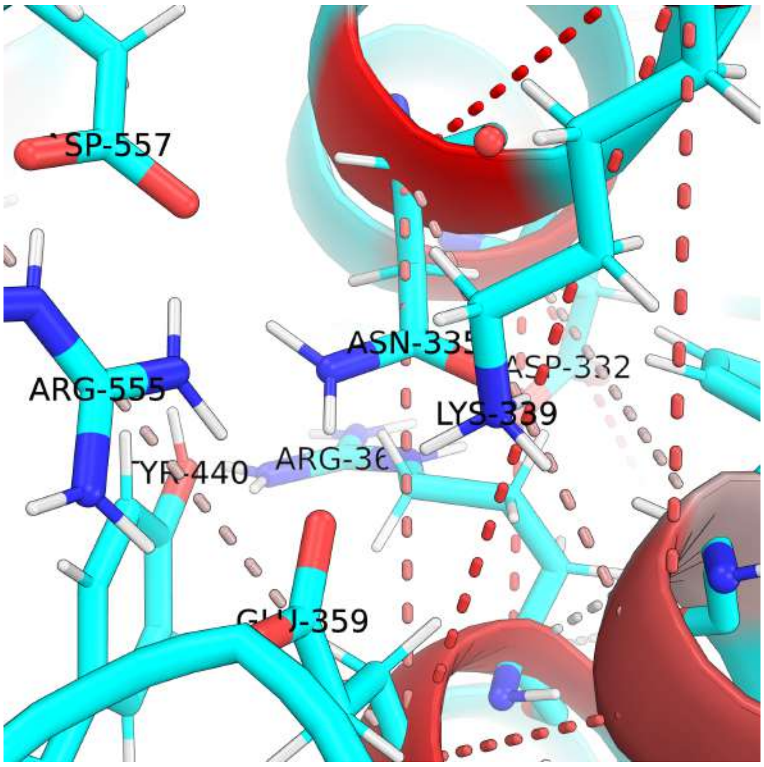


Figure 4. Salt bridges formed by subfamily-specific and correlating amino acid residues in 2-hydroxybiphenyl 3-monooxygenase from *Pseudomonas azelaica*

On the other side, in the conformationally flexible region limiting the FAD binding site, there are two glycine residues: Gly325, a conservative position for the entire family, and Gly323, a specific position for the 2-hydroxybiphenyl-3-monooxygenase subfamily. Molecular dynamics simulations indicate that their role is to provide conformational flexibility for the transfer of the isoalloxazine ring of FAD occurring in the catalytic mechanism. Only the Asp313 residue, which is conservative in the entire family, is directly interacting with the FAD molecule through hydrogen bonding with 2'-OH-ribitol, thereby contributing to the binding and orientation of the cofactor both in the static structure and in molecular dynamics simulations.

Earlier identified NAD binding pocket located next to the FAD binding subsite [22] is also formed by a number of subfamily-specific residues (Trp38, Ser40, Ser42, Arg46, Ser47, Ala180, Asn205, Ser291, Trp293). These residues are located in an elongated pocket adjacent to the FAD binding site, some of them are also involved in the formation of the substrate delivery tunnel (Fig. 5). Subfamily-specific positions Arg46, Ser47, Gly202, Ser203, Asn205, Arg242, Val253, Trp293, Met321, and Pro320, conservative for the entire family, play a crucial role forming the substrate binding site in 2-hydroxybiphenyl 3-monooxygenase structure. Functional role of some of them (Arg46, Arg242, Trp293) was earlier described [18, 22], the others (Ser201, Met223, Trp225, Val236, Val238, Ala240, Trp 254, Leu 375, Leu428) have not been observed in the course of bioinformatic analysis.

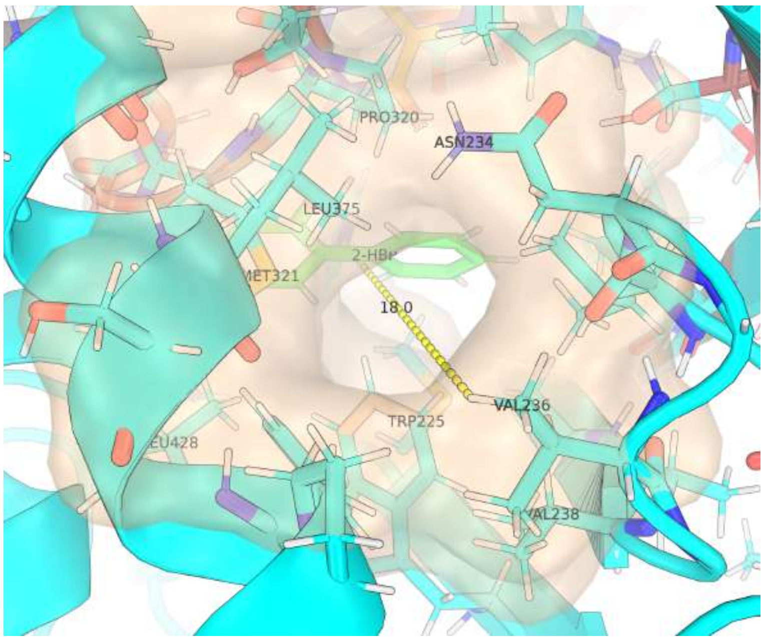


Figure 5. A tunnel for the entrance of the substrate to the active site of 2-hydroxybiphenyl 3-monooxygenase from *Pseudomonas azelaica*. The sticks highlight functionally important residues. The green color shows the substrate 2-hydroxybiphenyl

2.3. Subfamily-specific Amino Acid Residues in a Catalytic Mechanism

2-Hydroxybiphenyl 3-monooxygenase is capable to carry out stereospecific transformations because two hydrogen atoms at the C4 carbon of the dihydropyridine ring of NADH or NADPH as well as the sides of FAD isoalloxazine ring are not equivalent; the binding of cofactors and substrate in a quaternary complex and their orientation due to interactions with subfamily-specific positions Arg46, Ala180, His181 and Trp293 allows to perform the hydride transfer to the substrate stereospecifically. In particular, the triple stacking interaction between the FAD isoalloxazine ring, NADH nicotinamide ring and the subfamily-specific residue Trp293 leads to the formation of a highly stable charge-transfer complex and preferential Pro-S position in 2-hydroxybiphenyl 3-monooxygenase catalysis. As in the course of catalytic conversion the Trp293 residue is pushed aside, forming a cavity for the NADH nicotinamide ring, it is important to control dynamic changes in the active site. The efficiency of the stacking interaction is regulated by the Arg46 and Trp293 residues, and their relationship with the catalytic His48 residue located in the substrate binding site controls the interaction between the centers of the oxidative and reduction stages of the reaction mechanism [22].

2.4. Flexible Elements of the Structure

Along with the structure of the binding sites of cofactors and substrates, mobile structural elements play an important role in the catalytic mechanism, ensuring enzymes conformational plasticity thus controlling entrance to the channels and delivery of substrates and cofactors to the active site, as well as their retention in a bound state, isolation from the environment and ensuring the unique microheterogenicity of the medium for the effective catalytic transformations. In most cases, when constructing a full-scale structural model of an enzyme, it is necessary to determine the state of the mobile loops so that molecular modeling of the catalytic stages of the enzymatic reaction can be carried out adequately. There are four mobile regions in the structure of the substrate binding domain of 2-hydroxybiphenyl 3-monooxygenase formed by the polypeptide chain fragments Gln195-Ser201, Arg219, Tyr256-Ile265, and Arg228-Val236. Three loops, Gln195-Ser201, Tyr256-Ile265, and Arg228-Val236, are located in close proximity to the active site (Fig. 6).

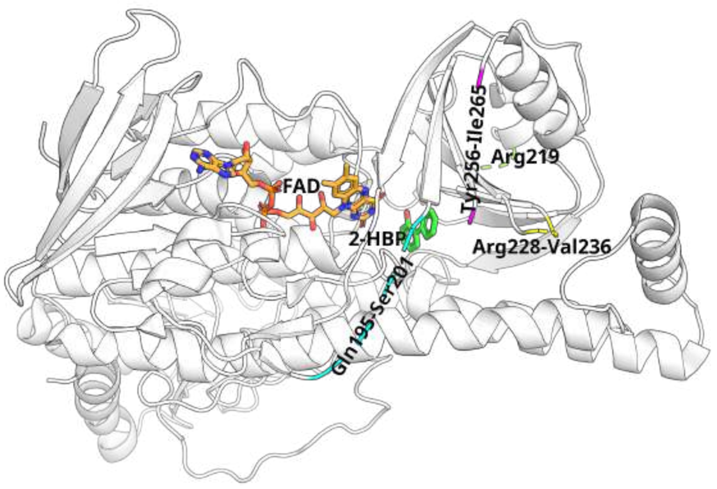


Figure 6. Unresolved structural fragments in the structure (PDB ID 5BRT) of 2-hydroxybiphenyl 3-monooxygenase from *Pseudomonas azelaica*: the blue color shows the *Gln195-Ser201* region, green – *Arg219*, pink – *Tyr256-Ile265*, yellow – *Arg228-Val236*. The FAD cofactor is shown in orange, the 2-hydroxybiphenyl substrate is shown in green

Full-length 2-hydroxybiphenyl 3-monooxygenase models with different positions of the loops predicted using different methods: LoopModel module of the Rosetta package, Modeller software, Alphafold v2.3.0 [13, 17, 34] are presented below (Fig. 7). Obviously, there is a remarkable discrepancy in the predicted positions of flexible loops. Subfamily-specific residues are not present in these regions, however Gly194 and Gly202 as well as Gly255 and Thr266 are located next to flexible loops and can make a remarkable impact on conformational dynamics of 2-hydroxybiphenyl 3-monooxygenase structure. We suggested to apply an integrated approach combining metadynamics simulations in latent space using variational autoencoders on supercomputers to explore initial approximations of these flexible region structures derived from modeling tools such as AlphaFold, RosettaFold, Modeller, SwissModel, etc. [21]. The predominant conformations of previously unresolved mobile regions in the active site of flavin-dependent 2-hydroxybiphenyl 3-monooxygenase identified using this approach are shown in Fig. 7.

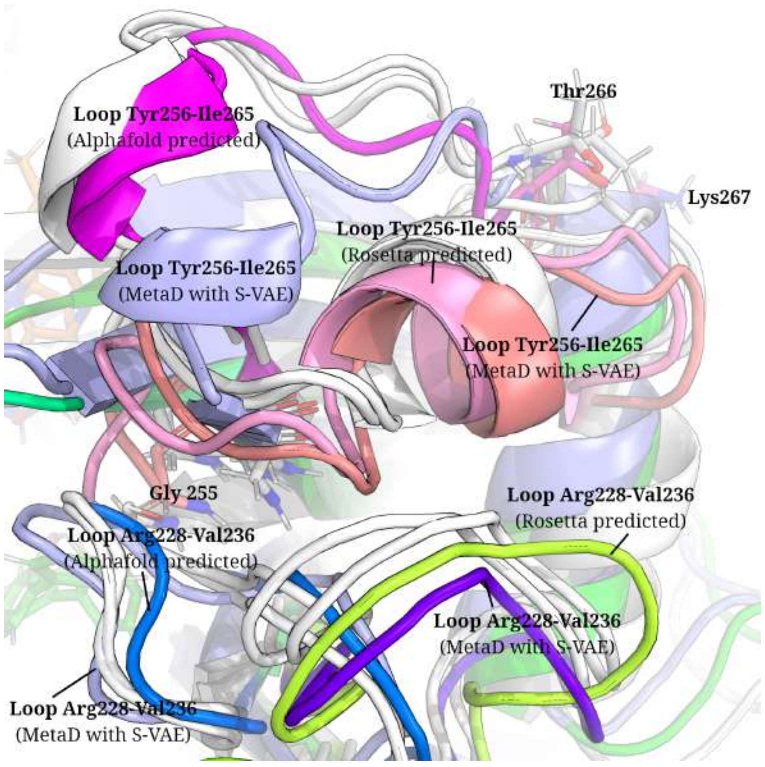


Figure 7. The positions of the loops in the active site of 2-hydroxybiphenyl 3-monooxygenase from *Pseudomonas azelaica*, predicted by Alphafold and Rosetta and optimized using molecular dynamics with hyperspherical variational autoencoder (S-VAE)

The coevolutionary relationship between the subfamily-specific residues Arg46 and Trp293, which regulate the efficiency of the stacking interaction between FAD and NADH was established using the *visualCMAT* server [41], as well as their relationship with the catalytic His48 residue located in the substrate binding site, which controls the interaction between the centers of the oxidation and reduction stages of the reaction mechanism. The important role of amino acid residues of the active site in the stabilization of the transition state should be noted: Trp293 forms a stacking interaction with the NADH nicotinamide ring, and His318 in the protonated form participates in stabilization due to the hydrogen bond formed between the backbone carbonyl oxygen and the NADH amide group. However, the catalytic mechanism of 2-hydroxybiphenyl 3-monooxygenase has not been fully understood yet. From two parts of the catalytic cycle, the reduction and oxidation half-reaction, only the first one was characterized on a molecular level: the reduction of FAD with NADH leading to the formation of $FADH^-$ anion, i.e., the step when hydrogen atoms are transferred between the coenzymes and the substrate. Stabilization of the transition state in 2-hydroxybiphenyl 3-monooxygenase catalysis occurs due to the induced fit formation of a specific pocket for binding the nicotinamide ring of NADH, which is connected to the active site by interacting subfamily-specific residues Arg46 and Trp293 [22]. Thus, bioinformatic analysis has helped to elucidate or suggest previously unknown roles for some amino acid residues, as well as to identify a list of residues that appear to play an important role in the evolution of this enzyme family, although the influence of these positions on the structural and functional properties is still unknown and should be investigated.

Conclusion

The bioinformatic analysis of the flavin-dependent monooxygenases used together with the modeling of predominant conformations of mobile loops in previously unresolved regions of 2-hydroxybiphenyl 3-monooxygenase structure using GPU-accelerated metadynamics simulations integrated with artificial intelligence and high-performance computing allowed to elucidate previously unknown functional roles for some amino acid residues in the reduction half-reaction. Three subfamily-specific residues Glu359, Lys339, Arg360 and the Asp332 residue, conservative throughout the entire enzyme family, were shown to form structurally important salt bridges Glu359-Lys339 and Arg360-Asp332, which stabilize alpha helices preserving the integrity of the Rossmann fold of the FAD-binding domain; subfamily-specific residues Trp338 and Glu359 were shown to provide the correct positioning of alpha-helices by interacting with two conservative residues Asp557 and Arg555 from the hydroxylase domain. Subfamily-specific residues Trp38, Ser40, Ser42, Arg46, Ser47,Ala180, Asn205, Ser291, Trp293 form the elongated NAD binding pocket adjacent to the FAD binding site. The conservative Asp313 residue makes important contribution to the binding and orientation of the cofactor through hydrogen bonding with 2'-OH-ribitol of FAD. The Arg46, Ser47, Gly202, Ser203, Asn205, Arg242, Val253, Trp293, Met321, and conservative Pro320 residue play a crucial role forming substrate binding site. The binding of cofactors and substrate in a quaternary complex and their orientation due to interactions with subfamily-specific residues Arg46, Ala180, His181 and Trp293 allows to perform the hydride transfer to the substrate stereospecifically. The triple stacking interaction between the FAD isoalloxazine ring, NADH nicotinamide ring and the subfamily-specific residue Trp293 leads to the formation of a highly stable charge-transfer complex and preferential Pro-S position in 2-hydroxybiphenyl 3-monooxygenase catalysis. A primary challenge for the future remains the role of amino acid residues in the mechanism of the oxidation half-reaction of the complete catalytic cycle of 2-hydroxybiphenyl 3-monooxygenase.

Acknowledgements

The study was conducted under the state assignments of Lomonosov Moscow State University for Belozersky Institute of Physicochemical Biology, topic 119042590056-2, and for Research Computing Center, topic 121011690003-6. The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University [47].

References

1. Apweiler, R., Bairoch, A., Wu, C.H., *et al.*: UniProt: the Universal Protein knowledgebase. Nucleic Acids Res 32(Database issue), D115–D119 (Jan 2004). `https://doi.org/10.1093/nar/gkh131`

2. Barozet, A., Bianciotto, M., Vaisset, M., *et al.*: Protein loops with multiple meta-stable conformations: A challenge for sampling and scoring methods. Proteins 89(2), 218–231 (Feb 2021). `https://doi.org/10.1002/prot.26008`

3. Berman, H.M., Westbrook, J., Feng, Z., *et al.*: The Protein Data Bank. Nucleic Acids Research 28(1), 235–242 (Jan 2000). `https://doi.org/10.1093/nar/28.1.235`

4. Bonati, L., Rizzi, V., Parrinello, M.: Data-driven collective variables for enhanced sampling. The Journal of Physical Chemistry Letters 11(8), 2998–3004 (2020). `https://doi.org/10.1021/acs.jpclett.0c00535`

5. Borges, P.T., Brissos, V., Hernandez, G., *et al.*: Methionine-Rich Loop of Multicopper Oxidase McoA Follows Open-to-Close Transitions with a Role in Enzyme Catalysis. ACS Catal. 10(13), 7162–7176 (Jul 2020). `https://doi.org/10.1021/acscatal.0c01623`

6. Bregman-Cohen, A., Deri, B., Maimon, S., *et al.*: Altering 2-Hydroxybiphenyl 3-Monooxygenase Regioselectivity by Protein Engineering for the Production of a New Antioxidant. ChemBioChem 19(6), 583–590 (2018). `https://doi.org/10.1002/cbic.201700648`

7. Corbella, M., Pinto, G.P., Kamerlin, S.C.L.: Loop dynamics and the evolution of enzyme activity. Nat Rev Chem 7(8), 536–547 (Aug 2023). `https://doi.org/10.1038/s41570-023-00495-w`

8. Crean, R.M., Biler, M., van der Kamp, M.W., *et al.*: Loop Dynamics and Enzyme Catalysis in Protein Tyrosine Phosphatases. J. Am. Chem. Soc. 143(10), 3830–3845 (Mar 2021). `https://doi.org/10.1021/jacs.0c11806`

9. Crozier-Reabe, K., Moran, G.R.: Form follows function: structural and catalytic variation in the class A flavoprotein monooxygenases. International Journal of Molecular Sciences 13(12), 15601–15639 (2012). `https://doi.org/10.3390/ijms131215601`

10. Davidson, T.R., Falorsi, L., De Cao, N., *et al.*: Hyperspherical variational auto-encoders. arXiv preprint arXiv:1804.00891 (2018). `https://doi.org/10.48550/arXiv.1804.00891`

11. Dawson, N.L., Lewis, T.E., Das, S., *et al.*: CATH: an expanded resource to predict protein function through structure and sequence. Nucleic Acids Research 45(D1), D289–D295 (2017). `https://doi.org/10.1093/nar/gkw1098`

12. Drobot, V.V., Kirilin, E.M., Kopylov, K.E., Švedas, V.K.: PLUMED plugin integration into high performance pmemd program for enhanced molecular dynamics simulations.

Supercomputing Frontiers and Innovations 8(4), 94–99 (2021). https://doi.org/10.14529/jsfi210408

13. Eswar, N., Webb, B., Marti-Renom, M.A., *et al.*: Comparative protein structure modeling using MODELLER. Current Protocols in Protein Science 50(1), 2–9 (2007). https://doi.org/10.1002/0471250953.bi0506s15
14. Fiser, A., Do, R.K.G., Šali, A.: Modeling of loops in protein structures. Protein Science 9(9), 1753–1773 (Jan 2000). https://doi.org/10.1110/ps.9.9.1753
15. Huijbers, M.M.E., Montersino, S., Westphal, A.H., *et al.*: Flavin dependent monooxygenases. Arch Biochem Biophys 544, 2–17 (Feb 2014). https://doi.org/10.1016/j.abb.2013.12.005
16. Jiang, H., Jude, K.M., Wu, K., *et al.*: De novo design of buttressed loops for sculpting protein functions. Nat Chem Biol 20(8), 974–980 (Aug 2024). https://doi.org/10.1038/s41589-024-01632-2
17. Jumper, J., Evans, R., Pritzel, A., *et al.*: Highly accurate protein structure prediction with AlphaFold. Nature 596(7873), 583–589 (2021). https://doi.org/10.1038/s41586-021-03819-2
18. Kanteev, M., Bregman-Cohen, A., Deri, B., *et al.*: A crystal structure of 2-hydroxybiphenyl 3-monooxygenase with bound substrate provides insights into the enzymatic mechanism. Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics 1854(12), 1906–1913 (2015). https://doi.org/10.1016/j.bbapap.2015.08.002
19. Kingma, D.P., Welling, M.: An Introduction to Variational Autoencoders. Foundations and Trends in Machine Learning 12(4), 307–392 (Nov 2019). https://doi.org/10.1561/2200000056
20. Kirilin, E.M., Švedas, V.K.: Study of the Conformational Variety of the Oligosaccharide Substrates of Neuraminidases from Pathogens using Molecular Modeling. Moscow Univ. Chem. Bull. 73(1), 39–45 (Jan 2018). https://doi.org/10.3103/S0027131418020050
21. Kopylov, K., Kirilin, E., Voevodin, V., Švedas, V.: Characterization of conformational flexibility in protein structures by applying artificial intelligence to molecular modeling. Journal of Structural Biology 217(2), 108204 (Jun 2025). https://doi.org/10.1016/j.jsb.2025.108204
22. Kopylov, K., Kirilin, E., Švedas, V.: Conformational transitions induced by NADH binding promote reduction half-reaction in 2-hydroxybiphenyl-3-monooxygenase catalytic cycle. Biochemical and Biophysical Research Communications 639, 77–83 (2023). https://doi.org/10.1016/j.bbrc.2022.11.066
23. Krissinel, E., Henrick, K.: Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. Acta Crystallographica Section D: Biological Crystallography 60(12), 2256–2268 (2004). https://doi.org/10.1107/S0907444904026460
24. Kundert, K., Kortemme, T.: Computational design of structured loops for new protein functions. Biol Chem 400(3), 275–288 (Feb 2019). https://doi.org/10.1515/hsz-2018-0348

25. Li, Z., Meng, S., Nie, K., *et al.*: Flexibility Regulation of Loops Surrounding the Tunnel Entrance in Cytochrome P450 Enhanced Substrate Access Substantially. ACS Catal. 12(20), 12800–12808 (Oct 2022). https://doi.org/10.1021/acscatal.2c02258

26. Li, Z., Xie, D., Song, C., *et al.*: The open-closed transitions within dynamic conformational changes of enzyme loops. Systems Microbiology and Biomanufacturing 6(1), 2 (Nov 2025). https://doi.org/10.1007/s43393-025-00396-7

27. Liao, Q., Kulkarni, Y., Sengupta, U., *et al.*: Loop Motion in Triosephosphate Isomerase Is Not a Simple Open and Shut Case. J. Am. Chem. Soc. 140(46), 15889–15903 (Nov 2018). https://doi.org/10.1021/jacs.8b09378

28. Malabanan, M.M., Amyes, T.L., Richard, J.P.: A Role for Flexible Loops in Enzyme Catalysis. Curr Opin Struct Biol 20(6), 702–710 (Dec 2010). https://doi.org/10.1016/j.sbi.2010.09.005

29. Marks, C., Deane, C.M.: Antibody H3 Structure Prediction. Computational and Structural Biotechnology Journal 15, 222–231 (Jan 2017). https://doi.org/10.1016/j.csbj.2017.01.010

30. Marks, C., Shi, J., Deane, C.M.: Predicting loop conformational ensembles. Bioinformatics 34(6), 949–956 (Mar 2018). https://doi.org/10.1093/bioinformatics/btx718

31. Nestl, B.M., Hauer, B.: Engineering of Flexible Loops in Enzymes. ACS Catal. 4(9), 3201–3211 (Sep 2014). https://doi.org/10.1021/cs500325p

32. Papaleo, E., Saladino, G., Lambrughi, M., *et al.*: The Role of Protein Loops and Linkers in Conformational Dynamics and Allostery. Chem Rev 116(11), 6391–6423 (Jun 2016). https://doi.org/10.1021/acs.chemrev.5b00623

33. Reis, R.A.G., Li, H., Johnson, M., Sobrado, P.: New frontiers in flavin-dependent monooxygenases. Arch Biochem Biophys 699, 108765 (Mar 2021). https://doi.org/10.1016/j.abb.2021.108765

34. Rohl, C.A., Strauss, C.E., Misura, K.M., Baker, D.: Protein structure prediction using Rosetta. Methods in Enzymology 383, 66–93 (2004). https://doi.org/10.1016/S0076-6879(04)83004-0

35. Salomon-Ferrer, R., Götz, A.W., Poole, D., *et al.*: Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. Explicit solvent particle mesh Ewald. Journal of Chemical Theory and Computation 9(9), 3878–3888 (2013). https://doi.org/10.1021/ct400314y

36. Shindyalov, I.N., Bourne, P.E.: Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. Protein Engineering 11(9), 739–747 (1998). https://doi.org/10.1093/protein/11.9.739

37. Stein, A., Kortemme, T.: Improvements to Robotics-Inspired Conformational Sampling in Rosetta. PLoS One 8(5), e63090 (2013). https://doi.org/10.1371/journal.pone.0063090

38. Stevens, A.O., He, Y.: Benchmarking the Accuracy of AlphaFold 2 in Loop Structure Prediction. Biomolecules 12(7), 985 (Jul 2022). https://doi.org/10.3390/biom12070985

39. Suplatov, D., Kirilin, E., Arbatsky, M., *et al.*: pocketZebra: a web-server for automated selection and classification of subfamily-specific binding sites by bioinformatic analysis of diverse protein families. Nucleic Acids Research 42(W1), W344–W349 (2014). https://doi.org/10.1093/nar/gku448

40. Suplatov, D., Sharapova, Y., Geraseva, E., Švedas, V.: Zebra2: advanced and easy-to-use web-server for bioinformatic analysis of subfamily-specific and conserved positions in diverse protein superfamilies. Nucleic Acids Res 48(W1), W65–W71 (Jul 2020). https://doi.org/10.1093/nar/gkaa276

41. Suplatov, D., Sharapova, Y., Timonina, D., *et al.*: The visualCMAT: A web-server to select and interpret correlated mutations/co-evolving residues in protein families. Journal of Bioinformatics and Computational Biology 16(02), 1840005 (2018). https://doi.org/10.1142/S021972001840005X

42. Suplatov, D.A., Kopylov, K.E., Popova, N.N., *et al.*: Mustguseal: a server for multiple structure-guided sequence alignment of protein families. Bioinformatics 34(9), 1583–1585 (2018). https://doi.org/10.1093/bioinformatics/btx831

43. Tribello, G.A., Bonomi, M., Branduardi, D., *et al.*: PLUMED 2: New feathers for an old bird. Computer Physics Communications 185(2), 604–613 (2014). https://doi.org/10.1016/j.cpc.2013.09.018

44. Van Berkel, W.J.H., Kamerbeek, N.M., Fraaije, M.W.: Flavoprotein monooxygenases, a diverse class of oxidative biocatalysts. Journal of Biotechnology 124(4), 670–689 (2006). https://doi.org/10.1016/j.jbiotec.2006.03.044

45. Vander Meersche, Y., Cretin, G., Gheeraert, A., *et al.*: ATLAS: protein flexibility description from atomistic molecular dynamics simulations. Nucleic Acids Res 52(D1), D384–D392 (Jan 2024). https://doi.org/10.1093/nar/gkad1084

46. Varadi, M., Anyango, S., Deshpande, M., *et al.*: AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. Nucleic Acids Research 50(D1), D439–D444 (Jan 2022). https://doi.org/10.1093/nar/gkab1061

47. Voevodin, V.V., Antonov, A.S., Nikitenko, D.A., *et al.*: Supercomputer Lomonosov-2: large scale, deep monitoring and fine analytics for the user community. Supercomputing Frontiers and Innovations 6(2), 4–11 (2019). https://doi.org/10.14529/jsfi190201

48. Wang, T., Wang, L., Zhang, X., *et al.*: Comprehensive assessment of protein loop modeling programs on large-scale datasets: prediction accuracy and efficiency. Briefings in Bioinformatics 25(1), bbad486 (Jan 2024). https://doi.org/10.1093/bib/bbad486

49. Zinovjev, K., Guénon, P., Ramos-Guzmán, C.A., *et al.*: Activation and friction in enzymatic loop opening and closing dynamics. Nat Commun 15(1), 2490 (Mar 2024). https://doi.org/10.1038/s41467-024-46723-9