

Supercomputing Frontiers and Innovations

2022, Vol. 9, No. 3

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

Editorial Board

Editors-in-Chief

- **Jack Dongarra**, University of Tennessee, Knoxville, USA
- **Vladimir Voevodin**, Moscow State University, Russia

Editorial Director

- **Leonid Sokolinsky**, South Ural State University, Chelyabinsk, Russia

Associate Editors

- **Pete Beckman**, Argonne National Laboratory, USA
- **Arndt Bode**, Leibniz Supercomputing Centre, Germany
- **Boris Chetverushkin**, Keldysh Institute of Applied Mathematics, RAS, Russia
- **Alok Choudhary**, Northwestern University, Evanston, USA
- **Alexei Khokhlov**, Moscow State University, Russia
- **Thomas Lippert**, Jülich Supercomputing Center, Germany

- **Satoshi Matsuoka**, Tokyo Institute of Technology, Japan
- **Mark Parsons**, EPCC, United Kingdom
- **Thomas Sterling**, CREST, Indiana University, USA
- **Mateo Valero**, Barcelona Supercomputing Center, Spain

Subject Area Editors

- **Artur Andrzejak**, Heidelberg University, Germany
- **Rosa M. Badia**, Barcelona Supercomputing Center, Spain
- **Franck Cappello**, Argonne National Laboratory, USA
- **Barbara Chapman**, University of Houston, USA
- **Yuefan Deng**, Stony Brook University, USA
- **Ian Foster**, Argonne National Laboratory and University of Chicago, USA
- **Geoffrey Fox**, Indiana University, USA
- **William Gropp**, University of Illinois at Urbana-Champaign, USA
- **Erik Hagersten**, Uppsala University, Sweden
- **Michael Heroux**, Sandia National Laboratories, USA
- **Torsten Hoeferl**, Swiss Federal Institute of Technology, Switzerland
- **Yutaka Ishikawa**, AICS RIKEN, Japan
- **David Keyes**, King Abdullah University of Science and Technology, Saudi Arabia
- **William Kramer**, University of Illinois at Urbana-Champaign, USA
- **Jesus Labarta**, Barcelona Supercomputing Center, Spain
- **Alexey Lastovetsky**, University College Dublin, Ireland
- **Yutong Lu**, National University of Defense Technology, China
- **Bob Lucas**, University of Southern California, USA
- **Thomas Ludwig**, German Climate Computing Center, Germany
- **Daniel Mallmann**, Jülich Supercomputing Centre, Germany
- **Bernd Mohr**, Jülich Supercomputing Centre, Germany
- **Onur Mutlu**, Carnegie Mellon University, USA
- **Wolfgang Nagel**, TU Dresden ZIH, Germany
- **Alexander Nemukhin**, Moscow State University, Russia
- **Edward Seidel**, National Center for Supercomputing Applications, USA
- **John Shalf**, Lawrence Berkeley National Laboratory, USA
- **Rick Stevens**, Argonne National Laboratory, USA
- **Vladimir Sulimov**, Moscow State University, Russia
- **William Tang**, Princeton University, USA
- **Michela Taufer**, University of Delaware, USA
- **Andrei Tchernykh**, CICESE Research Center, Mexico
- **Alexander Tikhonravov**, Moscow State University, Russia
- **Eugene Tyryshnikov**, Institute of Numerical Mathematics, RAS, Russia
- **Roman Wyrzykowski**, Czestochowa University of Technology, Poland
- **Mikhail Yakobovskiy**, Keldysh Institute of Applied Mathematics, RAS, Russia

Technical Editors

- **Yana Kraeva**, South Ural State University, Chelyabinsk, Russia
- **Mikhail Zymbler**, South Ural State University, Chelyabinsk, Russia
- **Dmitry Nikitenko**, Moscow State University, Moscow, Russia

Contents

How to Assess the Quality of Supercomputer Resource Usage Vad.V. Voevodin, D.I. Shaikhislamov, D.A. Nikitenko	4
A General Neural-Networks-Based Method for Identification of Partial Differential Equations, Implemented on a Novel AI Accelerator M.A. Krinitkiy, V.M. Stepanenko, A.O. Malkhanov, M.E. Smorkalov	19
Scalability and Performance of a Program that Uses Domain Decomposition for Monte Carlo Simulation of Molecular Liquids A.V. Teplukhin	51
Calculation of Electrostatic Potential Field of Coronavirus S Proteins for Brownian Dynamics Simulations E.P. Vasyuchenko, V.A. Fedorov, E.G. Kholina, S.S. Khruschev, I.B. Kovalenko, M.G. Strakhovskaya	65
Computational Alchemy Using Accelerated GPU Calculations: Fine Structural Tuning Inhibitors of L,D-transpeptidase 2 from <i>Mycobacterium tuberculosis</i> S.M. Baldin, V.O. Shegolev, V.K. Švedas	72



This issue is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

How to Assess the Quality of Supercomputer Resource Usage

Vadim V. Voevodin¹ , Denis I. Shaikhislamov¹ , Dmitry A. Nikitenko¹ 

© The Authors 2022. This paper is published with open access at SuperFri.org

Supercomputer is an exceptionally valuable computational resource and it must be used as efficiently as possible. However, in practice, the efficiency of its usage leaves much to be desired. There are various reasons for this. One of the main ones is the low performance of user applications, but users themselves are often not aware of the presence of performance issues in their programs. Therefore, it is necessary for administrators of a supercomputer to be able to constantly monitor the performance and behavior of all running jobs. However, the problem is that the commonly used metrics for assessing the quality of resource consumption (such as CPU or GPU load, the amount of bytes transferred over the MPI network, etc.) are often far from being convenient and accurate. This paper describes the implementation and evaluation of the previously proposed assessment system, which, in our opinion, makes it possible to significantly ease the task of properly evaluating the quality of the supercomputer resource usage. We also touch upon another topic related to the assessment of the quality of using HPC resources — organization of HPC resource provisioning.

Keywords: *supercomputing, high-performance computing, performance analysis, monitoring, workload analysis, resource utilization, resource provisioning.*

Introduction

Many modern supercomputers are inefficient. If we study, in detail, the use of the computational resources of a supercomputer, it often turns out that a significant part of these resources is either idle, or poorly utilized, or wasted (for example, in the case of an incorrect program launch). Over time, the situation does not get better: the constant complication of supercomputer architecture leads to an increase in its overall performance, but efficiency often drops as it becomes more and more difficult to properly utilize all the available hardware capabilities.

This is exacerbated by the fact that many supercomputer users initially come from not HPC-related areas (chemistry, physics, medicine, etc.) and therefore do not have sufficient theoretical knowledge and practical skills in writing highly efficient parallel applications [7]. Moreover, HPC area is growing quite fast [2], so more and more new specialists for various scientific fields start using supercomputers. With that, if a user does not pay for the consumed supercomputer node-hours, then s/he is not always motivated to care much about the performance of the applications. For example, if a user runs a job at night, from his/her point of view it may not make much difference whether it will run for 4 or 6 hours.

In such a situation, it is important for supercomputer administrators to control the efficiency of the operation themselves. And for this, it is necessary to constantly monitor and analyze the quality of its functioning, including the entire flow of running applications. And, at this moment, in practice, an unpleasant situation often arises: administrators are usually able to collect and store a wide variety of data on the behavior and performance of supercomputing applications, but it is not clear what to do with this data and how to extract useful information from it? In particular, how to understand which jobs have performance issues? Common metrics usually used for that (like CPU load, load average, frequency of LLC cache misses, or amount of bytes sent over MPI network) in many cases are not so insightful and do not help much in promptly detecting performance issues.

¹Lomonosov Moscow State University, Moscow, Russian Federation

For these purposes, we are developing an assessment system that is designed to quickly and accurately analyze the quality of the supercomputer resources usage. These assessments are needed for the initial analysis, which allows understanding, in general, which jobs have low efficiency and therefore need to be paid attention to. It is assumed that subsequent detailed analysis of the selected application, needed to determine the root causes of performance degradation and ways to eliminate them, should be performed using existing analysis tools, such as profilers, debuggers, etc.

We also draw attention to the problems of efficient HPC resource provisioning as a necessary part of the whole computing workflow. We have interviewed five large HPC centers in Russia and a few European ones to investigate the most concerning questions of HPC centers management and resource provisioning. In this paper, we give a short summary of these surveys regarding the observed problem.

The main contribution of this paper is the description of the methods for implementing previously proposed assessments in practice, as well as the demonstration of the applicability of these assessments using real-life collected statistics and interesting examples. Another contribution is a ranked list of the most important questions regarding HPC resource provisioning, built according to the survey of the largest HPC centers in Russia and a few European ones.

The rest of the paper is organized as follows. Section 1 describes our background, briefly presenting our previously proposed assessment system. Section 2 is devoted to the implementation of methods for collecting needed data and computing assessments. In Section 3, the analysis of some real-life statistics and specific examples collected on Lomonosov-2 supercomputer is performed. Section 4 describes machine learning techniques planned to be used for expanding the applicability of the proposed solution. Section 5 is aimed at questions of efficient resource provisioning and its assessment. Conclusions are described in the last section.

1. Previously Proposed Assessment System

In the previous paper [17], a description of the proposed assessments is given. It was decided that for each supercomputer job assessments should evaluate how “inefficiently” or “poorly” this job is using the given computational resource. In our case, we decided to assess how much working with the selected type of resource interferes with useful computations (which can be performed by CPU or GPU). If such interference is high, this means that a processor is far from being fully utilized, waiting for the execution of operations with the specified resource.

Assessments were developed for 6 types of resources — CPU, memory, MPI network, I/O, GPU and GPU memory, and specific formulas for their calculation were proposed (only general ideas were proposed for two GPU-related assessments). These assessments are briefly presented in Tab. 1. Hereinafter, assessment for a certain type of resources will be denoted as $\text{score}_{\text{type}}$ (for example, $\text{score}_{\text{cpu}}$).

The “Assessment based on” column specifies what idea underlies the proposed method for calculating each assessment. It can be seen that CPU and Memory assessments are based on Top-down approach developed by Intel [4, 20] (however, we have not seen the application of this approach not for one specific application but for the entire job flow, as done in our study). $\text{score}_{\text{mpi}}$ aggregates information about all MPI-related performance issues automatically detected by TASC [13]; the same is true for I/O assessment. Specific formulas for calculating these metrics were given in [17]. No specific formulas have been previously given for GPU-related assessments, but this has been done in this paper, see Section 2.

Table 1. A brief description of previously proposed assessments

Resource type	Assessment based on	Description
CPU	Top-down approach	Estimates the fraction of the CPU time when it was not fully utilized performing useful computations.
Memory	Top-down approach	Estimates the fraction of time that the processor was somehow idle, waiting for data from memory to be read or written.
MPI network	TASC	Evaluates the number and criticality of MPI-related issues causing a job to spend execution time on data exchange and not computations.
I/O	TASC	The same as above, but for I/O network.
GPU	—	GPU analogue of CPU assessment.
GPU memory	—	GPU analogue of Memory assessment.

A review of related work is given in the previous paper. Here, we only briefly note that at the moment no studies have been found in which analogues for the proposed assessments were proposed. However, it should be noted that some of these assessments are based on existing research, in particular, the aforementioned Top-down approach.

2. Implementation of Methods for Calculating Assessments

The formulas for calculating assessments $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ given in the previous paper have been slightly improved. So, a slightly more precise formula is now used to calculate $\text{score}_{\text{mem}}$, and the formula for the $\text{score}_{\text{cpu}}$ has been changed to comply with the general rule: each score must take values from 0 to 100, where 0 is the best value and corresponds to no interference with useful computations, while 100 is the worst value, meaning that usage of specified resource degrade efficiency all the time. The new formulas are shown in (1) and (2). All specified names in formulas are the processor sensors (in Linux `perf` notation).

$$\text{score}_{\text{cpu}} = 100 - 100 * \text{uops_retired.retire_slots} / (2 * \text{cpu_clk_unhalted.thread_any}) \quad (1)$$

$$\begin{aligned} \text{score}_{\text{mem}} = & (\min(\text{cpu_clk_unhalted.thread}, \text{cycle_activity.stalls_ldm_pending}) + \\ & \text{resource_stalls.sb}) / \text{cpu_clk_unhalted.thread} \end{aligned} \quad (2)$$

We tested this implementations on the Lomonosov-2 supercomputer installed at Lomonosov Moscow State University. All the data needed to calculate these assessments on Lomonosov-2 has been collected using DIMMon monitoring system [14]. For these purposes, a new DiMMon module was developed which collects data from the required performance monitoring counters (PMC). It should be noted that at this stage we encountered unexpected technical challenges. To obtain all the needed data, it was necessary to start using the multiplexing mode, since Intel Xeon E5-2697 v3 and Intel Xeon Gold 6126 processors used at Lomonosov-2 (like many other modern processors) allow simultaneously collecting data from only 4 sensors, and we needed a

total of 8 sensors (5 for assessments, 3 for other purposes). And it turned out that the PAPI [15] library, which we previously used to obtain data from processor counters, gives quite a noticeable overhead working in the multiplexing mode, thereby slowing down user applications.

Starting to investigate this topic, it turned out that at the moment there was no detailed study of the overhead caused by multiplexing, so we did this study ourselves [19]. It showed that the least overhead is obtained when using the LIKWID [9] library, and this is the solution we started using in practice on the Lomonosov-2 supercomputer. However, it should be noted that even in this case, the overhead can still be noticeable: on average, application execution time slows down by 2.78%, rarely reaching a maximum of more than 10%. With this in mind, it was decided to collect an extended set of sensors (i.e. using multiplexing mode) for every third job, so that the average slowdown of all user applications would be less than 1%. In order to obtain $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ for other jobs as well, we plan to use machine learning methods, which is discussed in Section 4 of this paper.

The implementation of $\text{score}_{\text{mpi}}$ and score_{io} has not changed: each score aggregates information on all MPI-related (or I/O-related) performance issues automatically detected using primary analysis implemented in TASC [12]. The only change is a technical one: the assessments are now collected automatically for all jobs using TASC workflow, which greatly simplifies their calculation and analysis.

These scores are not so precise as $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$, since they build on only those issues that can be detected using TASC, and automatic detection in TASC is quite limited due to the general constraints of constant performance monitoring and analysis. But, unlike mentioned scores, they are collected for all the jobs running on the supercomputer. We are conducting a study of possible approaches to obtain more accurate assessment for MPI, for example, based on the metrics proposed in the PoP project [3]. We have implemented a software prototype using the PnMPI [10], a lightweight tool for automatically collecting information about MPI calls for all running HPC jobs. However, this prototype, although making it possible to obtain a noticeably more accurate assessment, shows too high overheads, therefore does not currently allow applying it in practice. We are working on further improvement of it, trying to reduce the overhead.

In case of GPU, we want to develop assessments similar to $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$, i.e. Top-down-related ones. There are studies suggesting Top-down metrics for GPU (for example, [21]), but we need to calculate assessments for all jobs and therefore can not collect a lot of data (as not to increase the overhead), so it was necessary to strike a balance between accuracy and the amount of data collected. The current proposed formulas for $\text{score}_{\text{gpu}}$ and $\text{score}_{\text{gpumem}}$ are shown in (3) and (4).

$$\text{score}_{\text{gpu}} = 100 - 100 * \text{Eligible_Warps} / \text{Theoretical_Occupancy} \quad (3)$$

$$\text{score}_{\text{gpumem}} = 100 * (\text{Memory_Dependency} + \text{Memory_Throttle}) / \text{Active_Warps} \quad (4)$$

We intended to collect this data using a module of the DiMMon monitoring system, with the use of CUPTI [1] supported in PAPI. However, it turned out that apparently in our case it is technically impossible: most of our GPUs do not support external PMC data collection (capability ≥ 7.5 is required). There are other ways to collect the required data, such as using NVIDIA Nsight, but these solutions are likely to lead to significant overheads. At the moment, we are investigating ways for solving this technical problem.

3. Analyzing Real-life Statistics and Examples

In this section, we will show how suggested assessments can be used in practice to obtain useful and interesting insights. For this purpose, we analyzed all jobs that were executed on the Lomonosov-2 supercomputer during September 2022.

During this period, user run 14920 jobs in total. $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ were calculated for 9.5% of them. As mentioned before, data for these assessments is collected for every third job, but some other factors should be taken into account: there are many very short jobs — less than 1 minute — for which no monitoring data could be collected (usually, test or incorrect launches) due to data collection granularity; also, sometimes, monitoring system fails to collect all needed data. This explains why data was collected for only $\sim 10\%$ of jobs. $\text{score}_{\text{mpi}}$ and score_{io} can be potentially collected for any job, but they also rely on the presence of monitoring data. Moreover, there are many cases when an application does not use communication network at all or use it efficient enough (thus no MPI-related issues are detected), so the value of $\text{score}_{\text{mpi}}$ is non-zero for 17.15% of jobs (where at least MPI-related issue was detected), while score_{io} is non-zero in 0.34% cases. Such a situation with score_{io} indicates that we probably should revise I/O issue detection in TASC, since it is unlikely that any kind of such issues occur so rarely.

Figure 1 shows distribution of $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ for all jobs during September 2022. Each dot corresponds to a specific job, axis X is $\text{score}_{\text{cpu}}$ for a job, axis Y is $\text{score}_{\text{mem}}$. The general trend is noticeable — $\text{score}_{\text{cpu}}$ grows together with $\text{score}_{\text{mem}}$, which is generally logical, since the more memory interferes, the less the processor is able to perform useful computations. Also, it can be seen that “good” $\text{score}_{\text{cpu}}$ values, i.e. values that are close to zero, are not common. This is also not surprising, since the actual performance when using modern processors is very rarely close to peak performance. On the contrary, values of $\text{score}_{\text{mem}}$ stay in the low range of 10–40%. This means that memory usage on average does not interfere much with useful calculations. However, we can see jobs with $\text{score}_{\text{mem}}$ of almost 100%, meaning that usage of memory degrades application performance constantly during job execution. We can also notice, for example, that there is a job showing $\text{score}_{\text{cpu}}$ of $\sim 95\%$, but $\text{score}_{\text{mem}}$ is very low in this case with less than 10% (right bottom dot). This means that this program definitely has serious performance issues, but these issues are not related to memory usage.

Such a representation allows getting a general idea about the quality of usage of the selected resources in general and, in some cases, to identify anomalies.

Let us look at the available statistics from a different angle. Figure 2 shows top 25 jobs during the selected time period (September 2022) with the highest assessments overall (ranked by the sum of all assessments). With this we want to look at jobs with the most noticeable performance issues. Jobs are shown along the X axis, indicating the name of the user who ran them. Axis Y shows the value of three assessments — $\text{score}_{\text{cpu}}$, $\text{score}_{\text{mem}}$ and $\text{score}_{\text{mpi}} + \text{score}_{\text{io}}$ (last two are combined for convenience, all the more so as the second assessment is almost always zero, and both assessments are calculated in a similar way and therefore can be summed).

We can see that values of all these estimates range from 40% to 80%, and there seems to be a certain trend: the farther — the lower the $\text{score}_{\text{mem}}$ is, while it is not observed for two other assessments.

The most interesting fact in this figure is that 19 out of 25 jobs belong to one user (user1), as it can be seen in the X-axis label. This clearly indicates that performance issues have occurred repeatedly in the jobs of this user, and judging by the number of such jobs, this does not look

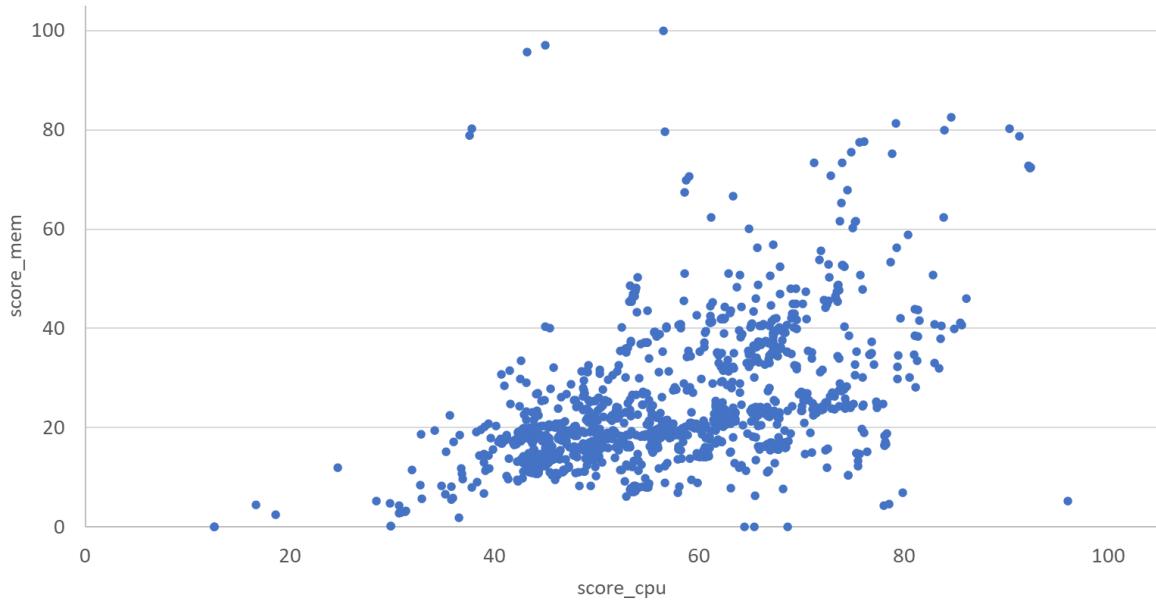


Figure 1. Distribution of $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ for all jobs during September 2022

like single random cases, but the presence of constant issues. Further analysis showed that these jobs in fact did show very low performance, and this was due to the fact that the user ran a series of test runs with intensive use of collective MPI-operations, which led to such bad assessments. In this case, the nature of the tests being run does not allow improving the use of computational resources, however, given that these tests are short and consume few node-hours, this is not a noticeable problem at the level of the entire supercomputer.

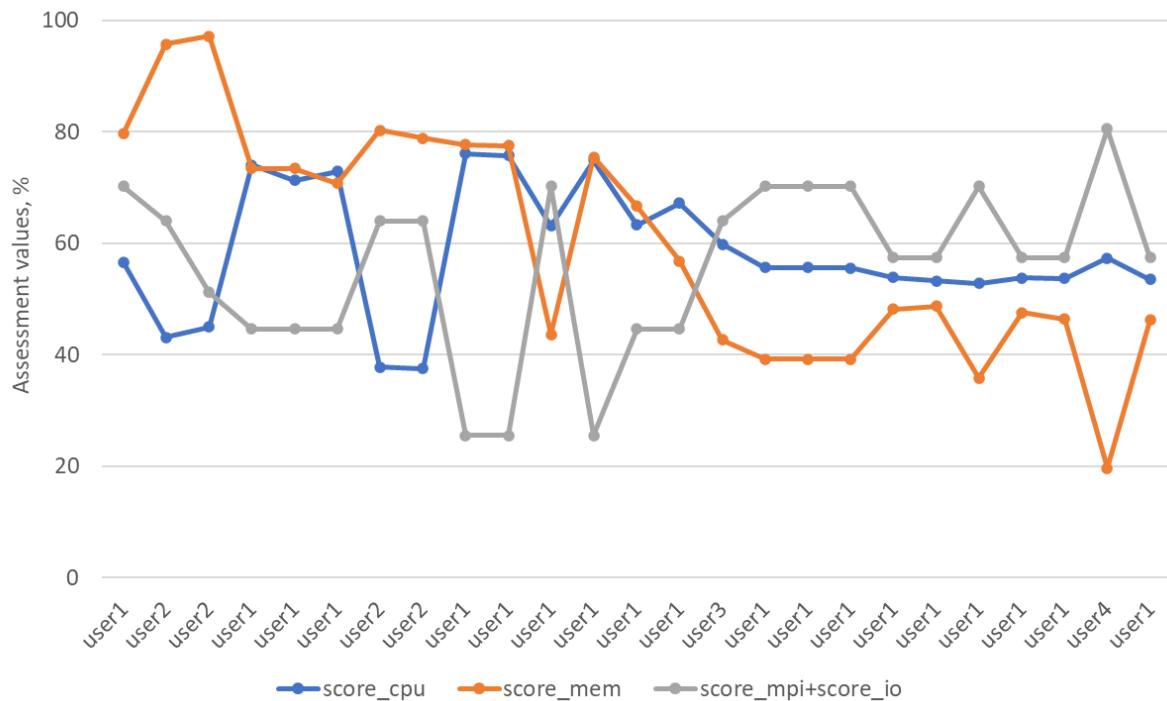


Figure 2. Top25 jobs with the highest sum of assessments

Considering the above, system administrators should pay attention to job #15 launched by user3: this job is much more computationally expensive than all the others presented in the Fig. 2. At the moment, this user has had only few launches, but they all consume noticeable amount of node-hours and all show bad assessment values, so it is worth paying attention to them: if their number increases, it is worth contacting the user and trying to find out what is the reason for this behavior. Moreover, all such jobs have a high $\text{score}_{\text{mpi}} + \text{score}_{\text{io}}$ score and they occupy quite a lot of compute nodes, which means that the task of optimizing MPI usage should be of the priority.

Now let us move on from considering individual jobs to studying aggregated information on individual users. In Fig. 3, top 50 users based on their weighted $\text{score}_{\text{cpu}}$ average during September 2022 are shown. Blue line shows weighted $\text{score}_{\text{cpu}}$ average for specific users. Here, weights are node-hours, i.e., value for each user is calculated as $\text{sum}(\text{score}_i * \text{nodehour}_i) / \text{sum}(\text{nodehour}_i)$, where score_i is $\text{score}_{\text{cpu}}$ for i -th job of selected user, and nodehour_i is the amount of node-hours i -th job has consumed. Orange dots correspond to total node-hours consumed by each user during the selected time period (note that these values are shown on the secondary Y axis).

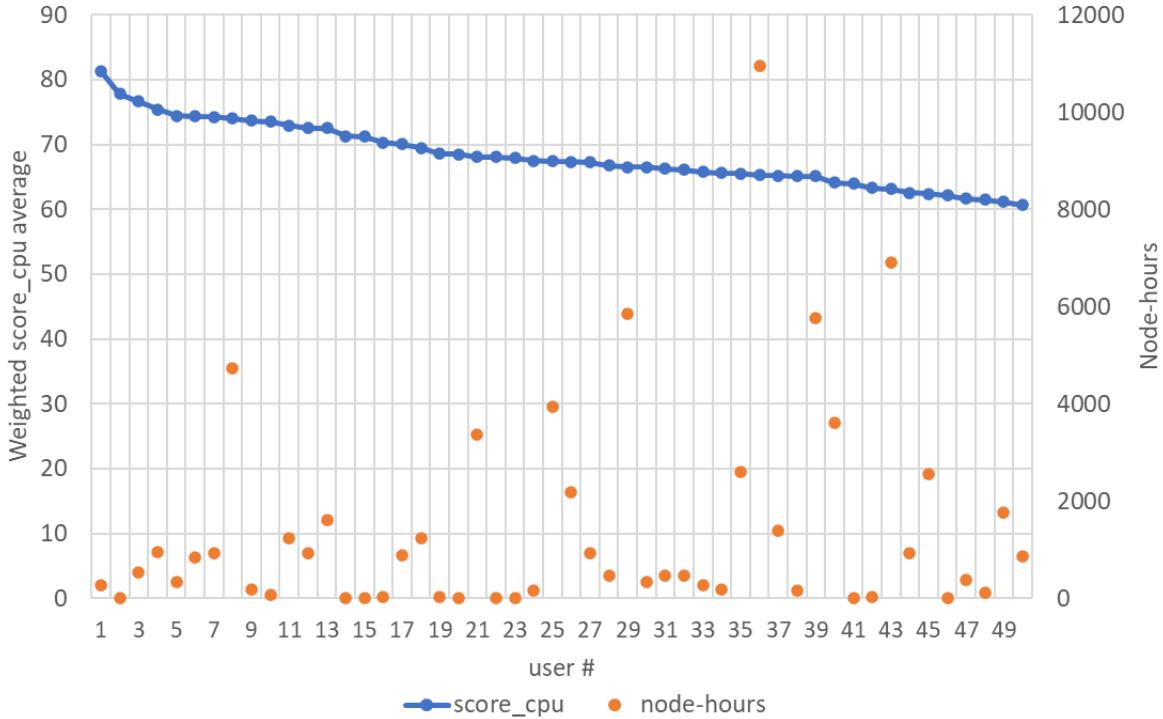


Figure 3. Top user ranking based on weighted $\text{score}_{\text{cpu}}$ average

Such information can be useful in different ways. For example, we noticed that users #1, #4, #5 and #11 are from the same project. Moreover, they all actively use LAMMPS software package [16], which suggests that these users are highly inefficient at using this package (because many other LAMMPS users have noticeably better $\text{score}_{\text{cpu}}$ values). Further analysis has shown that many of job launches under discussion seems to be incorrect: they have ended with FAILED finish state, e.g. one third of user #1 jobs has this state. This most likely means that users of this project have problems with the correct and efficient usage of the LAMMPS package, and system administrators should pay attention to this. It is interesting to note that the common

metrics for evaluating the processor load, CPU user load or load average, for all the considered jobs show almost optimal values, that is, it would be impossible to detect this situation using the standard approach.

These are just a few examples of how the proposed assessments can be applied in practice. The proposed solution is currently fully operational and available on the Lomonosov-2 supercomputer; in the future, we plan to integrate their use within the standard operating procedure for system administrators.

4. Using Machine Learning to Predict Assessment Values

It has been mentioned earlier that $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ are collected not for all jobs. But it would definitely be great to evaluate these assessments for as much jobs as possible. In order to achieve that, it is planned to apply machine learning methods. Previously, two methods for detecting similar supercomputer applications have been developed at the Research Computing Center of Lomonosov Moscow State University [11]. The first method is based on the use of the Doc2Vec neural network [6] to analyze static information on function and variable names obtained from binary and object files. The second method for detecting similar applications is based on applying the DTW algorithm [5] to multivariate time series, which are formed from the performance characteristic values obtained from the monitoring system during application execution. Such characteristics include, for example, CPU user load, L1 cache miss rate, the amount of data transferred over the MPI network per second, etc.

These methods have already shown high accuracy in practice, and therefore it has been decided to adapt these methods to solve a related problem: determining the values of proposed assessments based on historical data for jobs with no assessments obtained using the standard way described above. The solution should be mainly based on the dynamic method, as it is more suitable for fine-grain similarity detection. But, it is possible to combine both methods: first, we perform a primary filtering of jobs which can potentially be similar using a static method, and then apply a dynamic method to the remaining jobs. As the first results showed, this not only makes it possible to speed up the analysis process (since the dynamic method is much slower than the static one), but also to increase the accuracy. Thus, we propose the following general working algorithm:

- We collect a knowledge base, which accumulates historical information about assessment values for real-life Lomonosov-2 jobs (for which the standard method described above has worked).
- When a new job is found, and standard method failed to assess it, we do the following:
 - We find all jobs in the knowledge base that are similar to the new one using static analysis.
 - If we find statically similar jobs, we search for similar jobs among them using dynamic method with coarse threshold (step A).
 - If we do not find statically similar jobs, we select similar jobs in the whole knowledge base by dynamic method using a lot finer threshold (step B).

The idea is that if we find statically similar jobs the probability that dynamic analysis will falsely find similar job is low, so we can use coarse threshold to get more jobs of comparison.

After the analysis, we have a set of jobs that are similar to the target. For each similar job assessment values are known (they are stored in the knowledge base), so the question arise: how

should we predict the target's assessment value based on them? We considered three methods: median, average and softmax weighted average. Softmax weighted average is computed as:

$$\text{softmax weighted average} = \frac{\sum_i e^{-\text{distance}_i} * \text{score}_i}{\sum_i e^{-\text{distance}_i}}.$$

There is also a question of how to evaluate the accuracy of our prediction. Because the task of predicting assessment values is the regression task, we can use the regression evaluation metrics. The most common are Mean Absolute Error (MAE) and Mean Squared Error (MSE). In terms of the values, both MSE and MAE are positive numbers, and lesser the value — the better. MAE shows average error of the regression model, and higher the value — higher the error on average. This metric is very important in showing whether the predictions are close or not, but it has a big drawback — if there is a small percentage of predictions with very high error then the MAE will not be affected that much, because huge amount of accurate predictions will average the score out. That is why analyzing MSE is also important: it squares the error, and only then average is computed. This ensures that predictions with high error have greater effect on the score, thus, if the MSE is higher than the percentage of predictions with higher error will be higher too.

To test whether our proposed algorithm will work, we collected data on ~ 1500 jobs with execution time over 30 minutes. In this case, we can consider only quite long jobs because if execution time is small there is very little useful information about job behavior. Each job has assessments of the $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ that we can use for further comparison with the predicted value.

During testing, we changed 2 parameters of the specified algorithm: aggregation function (median, average or softmax weighted average) and selected approach. We decided to test three different approaches:

- Combined approach #1 — the default approach proposed earlier, using both steps A and B in the working algorithm.
- Combined approach #2 — the default approach, but with step A only, no step B, since step B allows analyzing more jobs but tend to decrease the accuracy.
- Dynamic analysis only — using dynamic analysis only, with no static analysis involved.

Tables 2 and 3 show the evaluation results for $\text{score}_{\text{cpu}}$ and $\text{score}_{\text{mem}}$ correspondingly.

Table 2. Accuracy of predicting $\text{score}_{\text{cpu}}$

Selected configuration	% of jobs predicted	MAE	MSE	% of jobs with AE >10
Combined approach #1, softmax	56.62	1.39	7.42	0.65
Combined approach #2, softmax	33.06	1.66	8.12	0.47
Dynamic analysis only, softmax	80.53	3.71	37.97	8.01
Combined approach #1, median	56.62	1.35	7.88	0.95
Combined approach #2, median	33.06	1.68	9.09	0.71
Dynamic analysis only, median	80.53	3.89	50.96	8.90
Combined approach #1, average	56.62	1.48	8.06	0.65
Combined approach #2, average	33.06	1.77	8.79	0.47
Dynamic analysis only, average	80.53	4.12	43.81	9.61

Table 3. Accuracy of predicting score_{mem}

Selected configuration	% of jobs predicted	MAE	MSE	% of jobs with AE >10
Combined approach #1, softmax	56.60	1.39	7.42	0.65
Combined approach #2, softmax	33.19	1.64	13.02	0.54
Dynamic analysis only, softmax	80.81	3.11	34.97	5.54
Combined approach #1, median	56.85	1.32	9.98	0.77
Combined approach #2, median	33.19	1.67	14.14	0.72
Dynamic analysis only, median	80.81	3.23	40.33	6.67
Combined approach #1, average	56.85	1.41	9.91	0.60
Combined approach #2, average	33.19	1.79	13.77	0.54
Dynamic analysis only, average	80.81	3.42	37.89	6.56

According to the provided results, dynamic only analysis is a clear winner in terms of the percentage of jobs predicted, as it covers >80% of jobs. But the quality of such prediction is quite low: more than 8% of jobs have the absolute error (AE) greater than 10, which is very high for a score ranging from 0 to 100. Combined approach #2 has the least percent of detected jobs with MAE and MSE slightly higher than if we use combined approach #1, but the percent of jobs with AE greater than 10 is the least among all configurations. Therefore, the combined approach #1 can be considered as the best option. It is in the middle ground between the other two variants in terms of percentage of detected jobs, has the best MSE and MAE, and also has a lot less percentage of jobs with high AE than dynamic analysis only. The benefits can also be seen in Fig. 4. Graphs show the distribution histogram, where each bar corresponds to a specific difference interval between true and predicted scores, and its height represents how many jobs show the specified difference interval. Two combined approaches are very similar in “tails” (where error is quite high), but approach #1 has a higher amount of accurate predictions (bars around zero value), almost by a factor of 2.

In terms of aggregation methods, there is no clear winner. Median shows that it has the least MAE, but highest MSE and percentage of jobs with high AE. This means that most of the time predictions are very accurate, but the cases of wrong predictions are way off the true values. Average has the least percentage of jobs with high AE, but has almost the same MSE as median. This means that there are many cases when the prediction is not so close to the ground truth, but the AE is still less than 10. Softmax weighted average is in the middle of the previous two: it has the average MAE and percentage of jobs with high AE, but a lot better MSE. This means that the predictions are very close to the median yet excluding some cases with high AE. But, overall, all aggregation variants are similar. This can clearly be seen in distribution histogram in Fig. 5, where all the graphs are practically the same. It tells us that in this case the aggregation function almost does not matter, and the defining factor whether the regression model will be accurate or not is determined by the selected approach. It is worth investigating on how to increase the percentage of found jobs in combined approach #1 while still retaining the quality of the predictions.

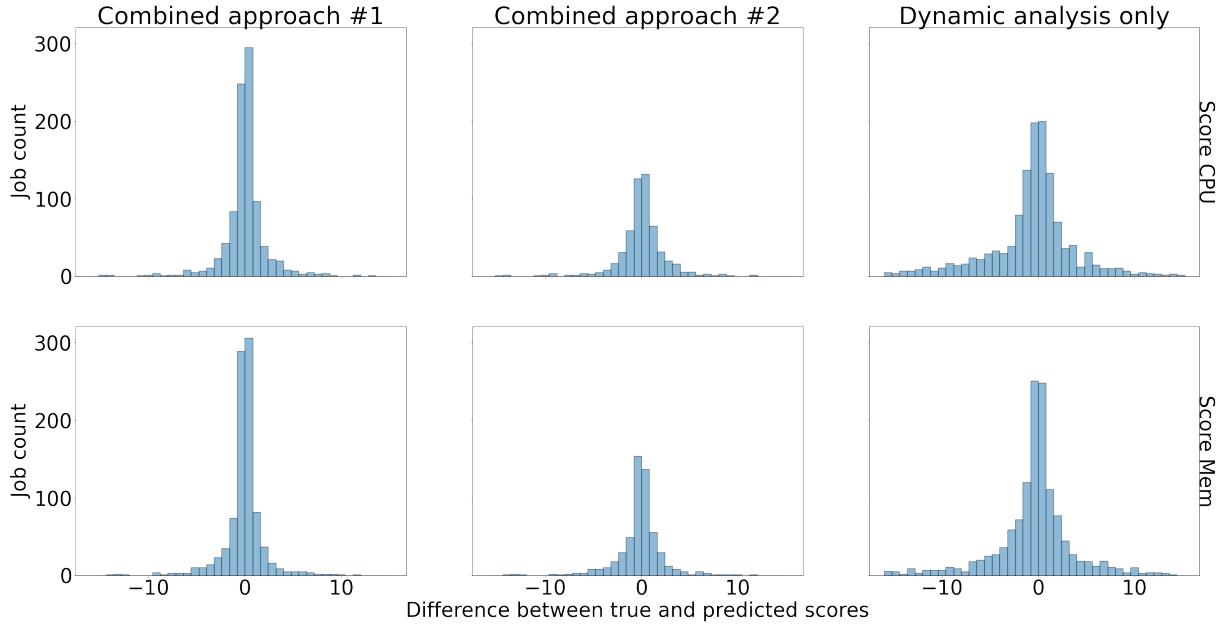


Figure 4. The distribution histogram for different approaches, softmax weighted average

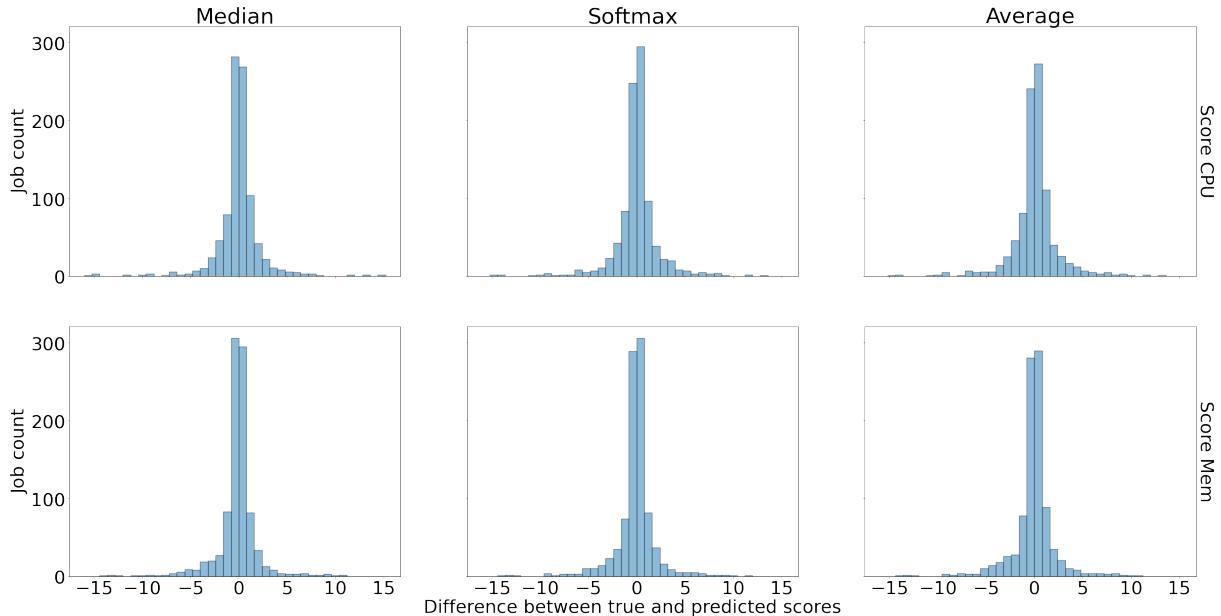


Figure 5. The distribution histogram for different aggregation methods, combined approach #1

5. Assessing the Quality of HPC Resource Provisioning

The assessment of the quality of using HPC resources would be incomplete if we do not take into account the issues of organizing the provision of these resources and the level of user interaction with the supercomputing center. In other words, it is expedient to jointly consider and assess the quality of computing resources provisioning. There are two main stages to be distinguished here:

- getting access to HPC resources;
- computing and user support.

Getting access for the first time can be a relatively complex process, mostly when getting access to the centers of collective use on a gratuitous basis, within the framework of collaborations, etc. Here, requests for confirmation of membership in a particular project, guarantees for the

use of resources for certain purposes and only for them, confirmation of the amount of resources provided, etc. are quite likely. All these questions are individual for each HPC facility, but, at the same time, they have much in common, which means that they can be formalized and automated in many ways. In particular, the Octoshell supercomputer center management system [8] provides such intuitive entities as “application”, “surety”, “access”, which can be omitted in the system, or brought to the required form with relative ease.

At the same time, even when obtaining access on a reimbursable basis, one still needs to declare the work aims and prove allocated resources needed, even at a high level of abstraction. Indeed, for large centers, the correct use of computing power is not only a matter of profit, but also of reputation. So, in many domestic and foreign HPC centers, both for calculations and for other services, for example, application optimization, reviewing has been introduced, and it will not be possible to receive even paid services without passing it. However, given that obtaining access is a one-time task for a research project, the significance of the degree of development of solving this issue is strictly proportional to the number of projects and their activity. At the same time, an inefficient solution to this issue adds difficulties and issues to the system holder to a much greater extent than to users.

The situation is quite different with the support phase of the calculations. It is obvious that at this stage there is a number of chains of interaction between the user and the system holder. Let us consider the main ones in descending order of importance based on a survey of representatives of the largest HPC centers in Russia and some of the leading European centers.

The first group with the highest priority includes the following two areas.

1. Availability of up-to-date user documentation

It includes frequently asked questions sections, documentation on the software and hardware stack, a detailed description of typical work scenarios and use cases (registration, expertise, correct project workflow, links to resources, related publications, etc.).

The phase requires little but constant attention from the system holder, and up-to-date information is extremely important for users. Unfortunately, the general practice is that, despite being in demand, users prefer to report bugs and inability to conduct computing or research before carefully studying the documentation. However, this is precisely the task of the system owner: to keep the documentation section in such a clear and up-to-date form that users intuitively access the information they are looking for on their own, thereby reducing the unnecessary burden on administrators.

2. User support, solving user problems

Helpdesk is usually implemented as a ticket system. The fundamental importance of full-fledged user support is to minimize the time for solving emerging problems of a different nature. This is not possible without a substantial staff of administrators, effective rubrics/templates for user requests, a detailed history for each issue, the ability to reassign and engage participants to solve the problem at hand. Absolutely all surveyed HPC centers put this functionality in first place immediately after ensuring the availability of all necessary up-to-date documentation.

The next area with a moderate priority is almost unanimously recognized as improving the efficiency of user tasks.

3. Improving the efficiency of user applications

Indeed, the return of the entire HPC center consists of all the results obtained within all ongoing projects, that is, they depend on the effectiveness of each launch. There is a clear division into the commercial use of resources and gratuitous use (including paid by a third-party source).

In commercial exploitation, the user's interest in this issue is higher, despite the fact that all users are interested in minimizing the time to obtain the final result.

It would seem that it is not so important for the owner of a computing system how quickly the applied tasks will be solved if the resources spent are paid in proportion to the time and scale of the tasks. However, almost all centers are interested in demonstrating their efficiency and productivity, both for maintaining the image/reputation and to substantiate directions for further own development, as well as to attract new users and just to help users solve their tasks faster.

In a number of foreign HPC centers, special departments have been allocated to analyze user applications, mainly on a commercial basis. In addition, support staff from equipment vendors is sometimes involved in resolving such issues, while users receive assistance, and the system manufacturer receives feedback, which is extremely important when designing new systems.

There are not very many developments in this direction among local supercomputing facilities: the shortage of staff of administrators and support groups is affecting, nevertheless, we can clearly see the demand by a number of users [18]. Works at the HPC center of Moscow State University stand out, namely the TASC system, aimed at assisting in the analysis of the efficiency of user jobs. It is important that the system provides users with data and recommendations on each run and with the possibility of feedback.

The areas with the lowest priority were identified by the system holders interviewed as follows.

4. Organization of the user's workspace

This includes user's personal account, organization of notifications about significant events in workflows, a "one-stop" service. Often, such solutions are bundled with the system provided by the supplier, so only large centers with complex workflows face the need for improvements.

5. Promoting user results

This direction is image-building, largely based on bullet 3. The presentation of success stories, vivid illustrations of the tasks being solved, announcements, publications: all this, of course, is necessary and contributes to the formation of the correct image of the supercomputer center, but it is relevant only when its productive functioning is already ensured.

Given that this distribution is built on the basis of the answers of large HPC systems holders, the considered gradation can be used as a basis for assessing the quality of user service provision, for example, as a basis for a scoring system with an appropriate distribution of weight coefficients according to the priorities mentioned above.

Conclusions and Future Work

This paper describes the implementation and evaluation of the previously proposed assessment system for analyzing the quality of HPC resource usage. For each supercomputer job, assessments evaluate how "inefficiently" or "poorly" this job is using the given computational resource.

The exact formulas for calculating all assessments are proposed, and their implementation (except for GPU-related assessments) was carried out on the Lomonosov-2 supercomputer. The collected real-life statistics and examples given in this paper show that proposed solution can be useful in many ways for system administrators, providing different insights on the quality of resource usage by particular jobs, users, projects, software packages, etc.

The authors also outline that granting high quality of resource provisioning and interaction with users is a necessary step when improving the efficiency of HPC resource usage. Five most popular types of such interaction are given in the paper according to the survey of large HPC centers in Russia and Europe.

Acknowledgements

The results described in this paper, except for Section 5, were achieved at Lomonosov Moscow State University with the financial support of the Russian Science Foundation (agreement No. 21-71-30003). The results described in Section 5 were achieved at Lomonosov Moscow State University with the financial support of RFBR, project number 20-07-00864. The research is carried out using the equipment of shared research facilities of HPC computing resources at Lomonosov Moscow State University.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. CUPTI :: CUDA Toolkit Documentation, <https://docs.nvidia.com/cuda/cupti/index.html>
2. High Performance Computing Market Size to Surpass USD 64.65, <https://www.globenewswire.com/news-release/2022/04/04/2415844/0/en/High-Performance-Computing-Market-Size-to-Surpass-USD-64-65-Bn-by-2030.html>
3. POP Standard Metrics for Parallel Performance Analysis | Performance Optimisation and Productivity, <https://pop-coe.eu/node/69>
4. Top-down Microarchitecture Analysis Method using VTune, <https://software.intel.com/en-us/vtune-cookbook-top-down-microarchitecture-analysis-method>
5. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03. pp. 359–370. AAAI Press (1994). <https://doi.org/10.5555/3000850.3000887>
6. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, June 21-26, 2014. JMLR Workshop and Conference Proceedings, vol. 32, pp. 1188–1196. JMLR.org (2014), <http://proceedings.mlr.press/v32/le14.html>
7. Nikitenko, D.A., Shvets, P.A., Voevodin, V.V.: Why do users need to take care of their HPC applications efficiency? Lobachevskii Journal of Mathematics 41(8), 1521–1532 (2020). <https://doi.org/10.1134/s1995080220080132>
8. Nikitenko, D., Voevodin, Vad.V., Zhumatiy, S.: Driving a petascale HPC center with Octoshell management system. Lobachevskii Journal of Mathematics 40(11), 1817–1830 (2019). <https://doi.org/10.1134/S1995080219110192>

9. Röhl, T., Eitzinger, J., Hager, G., Wellein, G.: LIKWID Monitoring Stack: A flexible framework enabling job specific performance monitoring for the masses. In: 2017 IEEE International Conference on Cluster Computing, CLUSTER 2017, Honolulu, HI, USA, September 5-8, 2017. pp. 781–784. IEEE (2017). <https://doi.org/10.1109/CLUSTER.2017.115>
10. Schulz, M., de Supinski, B.R.: P^n mpi tools: a whole lot greater than the sum of their parts. In: Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing, SC 2007, Reno, Nevada, USA, November 10-16, 2007. ACM Press (2007). <https://doi.org/10.1145/1362622.1362663>
11. Shaikhislamov, D., Voevodin, Vad.: Solving the problem of detecting similar supercomputer applications using machine learning methods. In: Parallel Computational Technologies. CCIS, vol. 1263, pp. 46–57. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-55326-5_4
12. Shvets, P., Voevodin, V., Zhumatiy, S.: Primary automatic analysis of the entire flow of supercomputer applications. In: CEUR Workshop Proceedings. pp. 20–32 (2018)
13. Shvets, P., Voevodin, Vad., Nikitenko, D.: Approach to workload analysis of large HPC centers. In: Parallel Computational Technologies. CCIS, vol. 1263, pp. 16–30. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-55326-5_2
14. Stefanov, K., Voevodin, Vl., Zhumatiy, S., Voevodin, Vad.: Dynamically Reconfigurable Distributed Modular Monitoring System for Supercomputers (DiMMon). Procedia Computer Science 66, 625–634 (2015). <https://doi.org/10.1016/j.procs.2015.11.071>
15. Terpstra, D., Jagode, H., You, H., Dongarra, J.J.: Collecting performance data with PAPI-C. In: Proceedings of the 3rd International Workshop on Parallel Tools for High Performance Computing, September 2009, ZIH, Dresden. pp. 157–173. Springer (2009). https://doi.org/10.1007/978-3-642-11261-4_11
16. Thompson, A.P., Aktulga, H.M., Berger, R., et al.: LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. Comp. Phys. Comm. 271, 108171 (2022). <https://doi.org/10.1016/j.cpc.2021.108171>
17. Voevodin, Vad., Zhumatiy, S.: Universal assessment system for analyzing the quality of supercomputer resources usage. In: Supercomputing. RuSCDays 2021. CCIS, vol. 1510, pp. 427–442. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92864-3_33
18. Voevodin, Vad.V., Chulkevich, R.A., Kostenetskiy, P.S., et al.: Administration, Monitoring and Analysis of Supercomputers in Russia: a Survey of 10 HPC Centers. Supercomputing Frontiers and Innovations 8(3), 82–103 (Oct 2021). <https://doi.org/10.14529/jsfi210305>
19. Voevodin, Vad.V., Stefanov, K.S., Zhumatiy, S.A.: Overhead analysis for performance monitoring counters multiplexing. In: Russian Supercomputing Days, RuSCDays 2022. LNCS, Springer, Cham (2022, in print)
20. Yasin, A.: A Top-Down method for performance analysis and counters architecture. In: ISPASS 2014 - IEEE International Symposium on Performance Analysis of Systems and Software, Monterey, CA, USA, March 23-25, 2014. pp. 35–44. IEEE (2014). <https://doi.org/10.1109/ISPASS.2014.6844459>
21. Zhou, K., Krentel, M.W., Mellor-Crummey, J.: Tools for top-down performance analysis of GPU-accelerated applications. In: Proc. of the 34th ACM Int. Conf. on Supercomputing, Barcelona, Spain, June, 2020. pp. 1–12. ACM (2020). <https://doi.org/10.1145/3392717.3392752>

A General Neural-Networks-Based Method for Identification of Partial Differential Equations, Implemented on a Novel AI Accelerator

Mikhail A. Krinit斯基^{1,2} , Victor M. Stepanenko^{2,3} ,
Alexey O. Malkhanov⁵ , Mikhail E. Smorkalov^{4,5} 

© The Authors 2022. This paper is published with open access at SuperFri.org

Partial differential equations (PDEs) are pervasive in vast domains of science and engineering. Although there is huge legacy of numerical methods for solving direct and inverse PDE problems, these methods are computationally expensive for many fundamental and real-life applications, demanding supercomputer resources. Moreover, existing methods for PDEs identification assume concrete functional forms for the coefficients to be found, significantly limiting the range of possible solutions. The mentioned circumstances lead to increasing interest in AI-based methods for direct solving and identification of PDEs. In this study, we propose a novel method based on artificial neural networks (ANNs) for the identification of partial differential equations. The method does not require any strong *a priori* assumptions regarding the family of the functions approximating PDE coefficients. It allows one to approximate the coefficients of a PDE based on the observed evolution of PDE direct solution. We demonstrate efficacy and high accuracy of ANN-based method in case of diffusion equation and nonlinear diffusion-advection equation (Richards equation) applied to the simulation of heat and moisture transfer in soil. We demonstrate that the novel method implemented on Ascend platform using the mixed precision floating point operations overperforms the classical gradient descent method in Barzilai–Borwein stabilized modification (BBstab, realized on a conventional central processor), in terms of MAPE (mean absolute percentage error) and RMSE (root mean square error) of approximated coefficients at least an order of magnitude. We also found that ANN-method is much less sensitive to initial guess of parameters compared to BBstab approach. Since the considered equations are of generic form, we anticipate that the proposed ANN-based method can be successfully exploited in other applications. These potential applications include hydrodynamic-type problems, e.g., optimization of turbulence closures, where the assumed reference solutions of PDEs are usually obtained from high-resolution direct Navier–Stokes simulations.

Keywords: partial differential equations, artificial neural networks, machine learning, inverse problems, land surface model, Richards equation, Ascend platform.

Introduction

Partial differential equations (PDE) are pervasive in many domains of science and engineering. Over the past decades, multiple methods have been proposed which allow to solve PDEs numerically, such as finite elements method (FEM), finite differences method (FDM), finite volumes method (FVM), etc. Such methods allow to achieve good accuracy of the numerical solution, but often are computationally expensive, which becomes an important limitation in applications when one needs to perform thousands or even millions of simulations. Such scenarios are common in engineering, e.g., the problem of finding optimal configuration of design parameters may require a large number of forward simulations for individual parameter sets. Another category of problems where traditional numerical methods often require heavy computations are inverse

¹Shirshov Institute of Oceanology, Russian Academy of Sciences, Moscow, Russian Federation

²Lomonosov Moscow State University, Moscow, Russian Federation

³Moscow Center of Fundamental and Applied Mathematics, Moscow, Russian Federation

⁴Skolkovo Institute of Science and Technology, Moscow, Russian Federation

⁵Huawei Nizhny Novgorod Research Center, Nizhny Novgorod, Russian Federation

or data assimilation problems where the governing PDEs need to be reconstructed based on the measurement data or data of very-fine-scale simulations recognized as “truth” (e.g., direct numerical simulations in turbulence [35]).

The limitations of classical numerical methods resulted in the increasing interest in AI-based methods for solving and/or identification of PDEs which hold a perspective of effectively addressing the above mentioned issues. These AI-based methods can be split into two major categories: data-driven methods and physics-informed ones. The former category includes methods which use only observation data to implicitly reproduce the physics of natural phenomena, with Fourier neural operator (FNO) [29] and deep operator networks (DeepONet) [32] being some of well-known examples. Physics-informed methods, in contrast, leverage known information about governing physical laws. The most famous and likely most widely used method in this category is physics-informed neural networks (and its flavours) [22, 38–40], developed by research group of professor G. Karniadakis at Brown University.

The application of AI-based methods allowed to achieve notable results in areas like protein structure prediction [4, 25], photonics [44], the solution to Schrödinger equation for fermions [37], quantum transport [45], molecular dynamics [23], climate analytics [28], weather prediction [43], computational fluid dynamics (CFD) [16, 30], solid mechanics [19], Earth radiation belt modeling [10] and others.

While the interest of scientific community in AI-based methods of solving PDEs does increase quickly, most case studies presented so far leverage single precision arithmetic and thus focus on commodity hardware such as CPU and general-purpose computing on graphics processing units (GPGPU), even though specialized deep learning accelerators could be a perfect hardware target for such technologies in theory, due to their higher computational performance. However, the important question which needs to be addressed to evaluate the usefulness of AI accelerators in the field of scientific computing is whether the half precision arithmetic typically supported by those accelerators would allow to achieve acceptable accuracy in the problems of interest. This motivates us to thoroughly consider the question of datatype precision in the current paper where to the best of our knowledge we present the first case study of using specialized AI accelerator for solving the PDE identification problem relevant to land surface modeling field via artificial neural network (ANN) based algorithm.

The paper is organized as follows: in the Section 1, we present the context of the problem of PDE identification and the specifics of the downstream tasks we consider in this study that are: soil heat conductance and soil water content dynamics. We introduce the PDEs that describe these physical processes. These PDEs are subjects to identify in our study. In Section 2, we present the methods for the identification of PDEs we developed in our study. Both classical method (Section 2.3) and the one based on artificial neural networks (ANNs) (Section 2.4) are presented in this section in a unified manner in order to make it easy for a reader to compare the approaches and find the key differences. In Section 2.4 we describe our method based on ANNs in detail, including the physics-guided regularizations, our improved scalars and tensors scaling scheme, and the scheme of random re-weighting of individual elements of loss sums. In Section 3, we present the results we deliver in this study. We compare the accuracy and the performance of classical identification method we reproduce in this study, with the accuracy and performance of the method based on ANNs we propose in this study. In Conclusions section, we summarize the paper and draw the conclusions based on our results.

1. Problem Setup

1.1. General Remarks on the PDE Identification Problem

Let us assume we have a partial differential equation or a system of PDEs supplemented with appropriate initial and boundary conditions:

$$B\mathbf{f} + A[\lambda(\mathbf{f})]\mathbf{f} = \mathbf{g}, \quad (1)$$

where B and A are differential operators, $\lambda(\mathbf{f})$ is a coefficient of the operator A which is a function of solution \mathbf{f} , \mathbf{g} is a given right-hand side (r.h.s.) function of time and space. Below we confine ourselves to operators A which are linear on λ . The inverse problem is formulated as follows: the PDE solution \mathbf{f} is given, B and A are known, $\lambda(\mathbf{f})$ has to be found. For any test function (coefficient) $\lambda^*(\mathbf{f}) = \lambda(\mathbf{f}) + \delta\lambda(\mathbf{f})$, we get a residual of (1), defined as ϵ :

$$B\mathbf{f} + A[\lambda(\mathbf{f}) + \delta\lambda(\mathbf{f})]\mathbf{f} - \mathbf{g} = \epsilon, \quad (2)$$

$$A[\delta\lambda(\mathbf{f})]\mathbf{f} = \epsilon. \quad (3)$$

The residual ϵ is zero if $\delta\lambda(\mathbf{f}) = 0$. The opposite is not true in general case, i.e., there might be non-trivial solutions $\delta\lambda(\mathbf{f})$ of equation (3) with ϵ . This means the solution of inverse problem is not unique in general case. The practical consequence of this fact is that minimizing ϵ to zero over a space of $\lambda^*(\mathbf{f})$ functions does not necessarily provide $\delta\lambda(\mathbf{f}) \rightarrow 0$. In this study, we impose additional constraints on $\lambda^*(\mathbf{f})$, which ensure the correct solution of the inverse problem, including the monotonicity and matching the known points of $\lambda(\mathbf{f})$ (see Section 2.4.3).

Let us assume now, that we have a solution of (1), \mathbf{f} , at a set of nodes in time and space (say, data of fine-scale measurements or simulations with *a priori* correct but computationally expensive mathematical model). Then, the equation (1) can be discretized as follows:

$$\epsilon_{h\tau}^* \doteq B_{h\tau}\mathbf{f}_{h\tau} + A_{h\tau}[\lambda(\mathbf{f}_{h\tau}) + \delta\lambda(\mathbf{f}_{h\tau})]\mathbf{f}_{h\tau} - \mathbf{g}_{h\tau} = \epsilon_{h\tau} + O(h^n) + O(\tau^m), \quad (4)$$

where h and τ stand for spatial and temporal spacing of the mesh where the above mentioned data is available, $\mathbf{f}_{h\tau}$, $\mathbf{g}_{h\tau}$, $\epsilon_{h\tau}$ are the projections of \mathbf{f} , \mathbf{g} , ϵ onto the mesh, n and m are the orders of approximation of A and B by discrete analogues $A_{h\tau}$ and $B_{h\tau}$, respectively. Now, under fixed h and τ , minimizing the norm of residual of (4), $\|\epsilon_{h\tau}^*\|$, over $\lambda^*(\mathbf{f}_{h\tau})$ does not lead to vanishing of $\epsilon_{h\tau}$ and $\delta\lambda(\mathbf{f}_{h\tau}) \rightarrow 0$. This is because the residual of discretized PDE is caused by both the deviation of test coefficient function from the true one and discretization errors, so that vanishing $\|\epsilon_{h\tau}^*\|$ means $\epsilon_{h\tau} \rightarrow O(h^n) + O(\tau^m)$. In this study, we minimize the discretization issue by using fine-resolution grids for PDE approximation.

1.2. Equations for Soil Thermodynamics and Hydrodynamics in Weather and Climate Models

The land surface scheme is a necessary compartment of any numerical weather forecast system or the Earth system model. Its key feature is the presence of a number of loosely constrained parameters of PDE coefficients which need to be identified using observation data. This includes the biophysical properties of soil and vegetation responsible for simulation of soil moisture regime and river runoff, *inter alia*. In this paper, we suggest a new ANN-based approach for identification of heat conductance and Richards equations. These equations are standard for

land surface schemes, and are used in the land surface scheme jointly developed by the Institute of Numerical Mathematics (INM) RAS and Lomonosov Moscow State University (MSU). This land surface scheme is a module of the national INMCM Earth system model [42] and SLAV numerical weather forecast system [13].

The basic numerical kernel of any land surface scheme is a solver for an equation system governing heat and water transport in soil. This system includes heat equation:

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial z} \left(\lambda_T \frac{\partial T}{\partial z} \right) + \rho_d (L_i F_i - L_v F_v), \quad (5)$$

where T is temperature, ρc is volumetric specific heat of soil, ρ_d is dry soil density, L and F are specific heat and rate of water phase transition (subscript i standing for freezing/melting and v denoting evaporation/condensation processes), t is time, z is a spatial coordinate directed along gravity. Importantly, the heat conductivity coefficient λ_T is a function of liquid water content W (expressed in kg/kg or m³/m³), i.e., it depends on the solution of the equation set. The l.h.s. represents the change of soil enthalpy, and terms to the r.h.s. stand for heat conductance and heat release/consumption due to phase changes. An equation for liquid water content is referred to as Richards equation and takes into account vertical transport (diffusion and gravitational infiltration), freezing/melting and evaporation/condensation:

$$\frac{\partial W}{\partial t} = \frac{\partial}{\partial z} \left(\lambda_W \frac{\partial W}{\partial z} \right) + \frac{\partial \gamma}{\partial z} - F_i - F_v - R_{roots} - R_{runoff}, \quad (6)$$

where γ is gravitational water flux, R_{roots} is soil moisture uptake by roots, and R_{runoff} is a sink of water due to horizontal runoff. This equation involves a concept of soil moisture potential (or water retention curve, WRC), Ψ , and hydraulic conductivity, γ , which are dependent on moisture W , defining coefficients λ_W, γ as functions of PDE solution. The liquid water diffusivity $\lambda_W(W)$ and hydraulic conductivity (HC) $\gamma(W)$ are related functions, i.e.:

$$\lambda_W(W) = \gamma(W) \frac{\partial \Psi(W)}{\partial W}. \quad (7)$$

At least 22 semi-empirical forms are proposed for the WRC function [12], fitting different sets of empirical data with different performance. The WRC function explicitly enters the hydraulic conductivity function. For instance, choosing Mualem approach for HC quantification [36], one arrives to:

$$\gamma = \gamma_s \widetilde{W}^{1/2} \left[\int_0^{\widetilde{W}} \frac{d\widetilde{W}'}{\Psi(\widetilde{W}')} \left(\int_0^1 \frac{d\widetilde{W}'}{\Psi(\widetilde{W}')} \right)^{-1} \right]^2, \quad (8)$$

where $\widetilde{W} \doteq (W - W_r)/(W_s - W_r)$ is a degree of soil moisture saturation, subscripts “s” and “r” standing for saturated and residual quantities of liquid water in a soil matrix. Thus, introducing in (8) and (7) $n = 22$ forms of WRC yields 22 possible pairs of functions (λ_W, γ) , based on Mualem equation. Given existence of different theoretical approaches (including Mualem’s formulation (8)) to quantify hydraulic conductivity [9, 14, 36], say, m approaches, we get $m * n$ possible functional forms for a pair (λ_W, γ) , which is ~ 100 for the current state of the field. This clearly shows that no generally accepted formulations for these two coefficients are available, so that a method, deriving coefficients $\lambda_W(W)$ and $\gamma(W)$ *independently* from data on measurable variables such as W with very general *a priori* assumptions on these functions (such as positive

definiteness, monotonicity) would ease the collection of more empirical data on WRC and HC functions and eventually serve the development of unified theory.

The dynamics of water vapor (V , kg/kg) in soil pores in the INM RAS-MSU land surface model is governed by diffusion equation with evaporation/condensation term:

$$\frac{\partial V}{\partial t} = \frac{\partial}{\partial z} \lambda_V \frac{\partial V}{\partial z} + F_v, \quad (9)$$

(λ_V – diffusivity coefficient for the water vapor) while the evolution of ice content I (kg/kg) is defined by phase transitions only:

$$\frac{\partial I}{\partial t} = F_i. \quad (10)$$

The system of equations (5), (6), (9), (10) is supplemented in the land surface model by boundary conditions, representing heat and moisture balance at the top and bottom margins of the soil column:

$$-\lambda_T \frac{\partial T}{\partial z} \Big|_{z=0} = R_{net} - H_s - LE_s, \quad (11)$$

$$-\lambda_W \frac{\partial W}{\partial z} \Big|_{z=0} = r_{pr} - E_s, \quad (12)$$

$$V|_{z=0} = V_0(t), \quad (13)$$

$$\frac{\partial T}{\partial z} \Big|_{z=H} = \frac{\partial W}{\partial z} \Big|_{z=H} = \frac{\partial V}{\partial z} \Big|_{z=H} = 0. \quad (14)$$

Here, R_{net} is the net radiation, H_s and LE_s are sensible and latent heat fluxes in the surface air layer, respectively, L – specific heat of evaporation/condensation, r_{pr} – the liquid water flux to the soil from rain or melted snow. The lower boundary conditions mean zero flux. The infiltration flux, γ is non-zero at the lower bound of soil column, and is assumed to contribute the river runoff. The aforementioned boundary conditions are prone to uncertainties induced by quantifying radiation and heat fluxes, both in measurements and theoretical approaches. Thus, in this study, we make use of a fact that at the diurnal scale, the temperature and moisture remain almost constant at sufficiently large depth, and follow sinusoidal pattern at the soil-atmosphere interface (see Problem Specification section).

1.3. Problem Specification

The key feature of the soil model described above is a set of PDE coefficients, which are dependent on PDE solution. These coefficients are routinely not measured. Our objective is to find coefficients as functions of solution, given the solution is known. The temperature T and moisture W can be available either from observations or from reference direct problem solution with *a priori* given coefficients. In order to simplify the task, we neglect water phase transitions, root uptake, horizontal runoff, which leaves us with a set of two equations – (5), (6) – with $F_i = F_v = R_{roots} = R_{runoff} = 0$:

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial z} \left(\lambda_T \frac{\partial T}{\partial z} \right), \quad (15)$$

$$\frac{\partial W}{\partial t} = \frac{\partial}{\partial z} \left(\lambda_W \frac{\partial W}{\partial z} \right) + \frac{\partial \gamma}{\partial z}, \quad (16)$$

and the simplified Dirichlet-type boundary conditions are specified as:

$$f|_{z=0} = f_1 + \frac{1}{2}(f_2 - f_1)(1 + \sin(\omega t - \pi/2)), \quad (17)$$

$$f|_{z=H} = \frac{1}{2}(f_1 + f_2), \quad (18)$$

where $f = T, W$, $W_1 = W_{min}$, $W_2 = W_{max}$, and $T_1 = T_{max}$, $T_2 = T_{min}$ ($T_{min} = 0$ K, $T_{max} = 5$ K, $W_{min} = 0.2$ kg/kg, $W_{max} = 0.65$ kg/kg), and ω is a frequency of the surface forcing, in this study set as $\omega = 2\pi/T_d$, $T_d = 24$ h. We substituted the conventional flux-type boundary conditions (11)–(14) with (17)–(18) to avoid additional uncertainties associated to parameterization of heat and moisture turbulent fluxes to the atmosphere (whereas Dirichlet conditions are directly measurable). The initial profiles for temperature and moisture are:

$$f|_{t=0} = f_1 + (1 - \exp(-\lambda_{dec}z))(f_2 - f_1), \quad (19)$$

with $f = T, W$, $\lambda_{dec} = 3$ m⁻¹.

The coefficients λ_T , λ_W and γ are assumed to be the functions of W only, i.e., no explicit dependence on z is imposed owing to the vertical soil structure. In other words, all soil physical parameters (e.g., soil porosity, heat conductivity of solid particles, hydraulic conductivity at saturation, etc.) potentially affecting thermal conductivity and Richards equation coefficients are taken as depth-independent.

Equations (9) and (10) are no longer a part of the problem, due to neglecting the phase transitions. The depth of soil is set as $H = 1$ m, a depth where diurnal oscillations are typically vanishing in real soils. In this study, the direct problem is solved using the reference (“true”) functions describing λ_T , λ_W , γ using finite difference scheme and thus providing $T_{h\tau}$, $W_{h\tau}$ at a fine grid, with number of layers $n_z = 1000$, and timestep $\Delta t = 10$ s or $\Delta t = 1$ h. Then the reference λ_T , λ_W , γ are sought given $T_{h\tau}$, $W_{h\tau}$ only. The two methods for seeking the PDE coefficients are used:

- method constructed by joining the well-known elements (referred to hereafter as “classical method”): the functional form of coefficients is given by one of options well-established in soil science (Cote-Konrad [11] or Johansen [24] representations of λ_T , and Mualem-van Genuchten [17], Books-Corey [7] or Gardner [15] formulae for λ_W , γ), the parameters of these forms are optimized by stabilized Barzilai–Borwein algorithm [6, 8];
- new artificial neural network (ANN)-based method: the functional form of coefficients λ_T , λ_W , γ is represented by artificial neural network, and the parameters of ANN are optimized.

In the implementation of ANN method, a special focus is put on the impacts of using FP32 (floating point, 32 bits) and FP16 (floating point, 16 bits) precision on solution wall-clock time and accuracy of inverse problem solution using the Ascend platform provided by Huawei.

In both methods of PDE identification, the parameters of analytical forms for λ_T , λ_W , γ are sought to minimize a loss function of PDE residual $\epsilon_{h\tau}$ (see eq. (4)), where operators $B_{h\tau}$ and $A_{h\tau}$ represent discretization of heat conduction (15) and Richards (16) equations. The discretization of heat conduction equation and diffusive part of Richards equation is central-difference in space and 1st order implicit in time. The advective part of equation (16) is approximated by 1st order explicit scheme in time and central-differences in space. Thus, the Richards equation is solved by time-splitting by physical processes (terms to the r.h.s. of (16)).

2. Methods

2.1. Reference Solutions

2.1.1. Heat conduction equation

To produce the reference numerical solution of direct heat conduction problem, we use the Cote-Konrad model for coefficient of soil heat conductivity [11] (hereafter denoted as “C-K”):

$$\lambda_T = \lambda_{T,dry} + (\lambda_{T,sat} - \lambda_{T,dry})Ke, \quad (20)$$

$$\lambda_{T,dry} = \xi 10^{(-\eta * p)}, \quad (21)$$

$$\lambda_{T,sat} = \lambda_w^p \lambda_{sp}^{1-p}, \quad (22)$$

$$Ke = (K_{te} \widetilde{W}) / (1 + (K_{te} - 1) \widetilde{W}), \quad (23)$$

where $\lambda_{T,dry}$ and $\lambda_{T,sat}$ are the values of conductivity coefficient for dry and saturated soil, respectively, Ke is the so-called Kersten number, p is the soil porosity (taken as 0.496, from the measurements in soil of Meteorological Observatory of Moscow State University [27]), λ_w is heat conductivity of water, λ_{sp} is heat conductivity of soil solid particles (taken as $2.5 \text{ W m}^{-1} \text{ K}^{-1}$), and the soil-type specific constants are set to be representative of silt and clay: $K_{te} = 1.69$, $\eta = 1.8$, $\xi = 1.7 \text{ W m}^{-1} \text{ K}^{-1}$, $\rho c = 2.4 \times 10^6 \text{ J m}^{-3} \text{ K}^{-1}$. The Cote-Konrad model is a representative of a class of conductivity models based on normalized heat conductivity concept, where Kersten number is used to interpolate between dry and saturated soil state.

2.1.2. Richards equation

To produce reference numerical solutions of Richards equation (16), we involved widely used functions of the soil moisture developed by Mualem [36] and van Genuchten [17] for the liquid water diffusivity $\lambda_W(W)$ and hydraulic conductivity $\gamma(W)$ (hereafter abbreviated as “M-vG”):

$$\lambda_{W,M-vG}(W) = \frac{\gamma_s(1-m)}{\alpha m(W_s - W_r)} \widetilde{W}^{1/2-1/m} \left[\left(1 - \widetilde{W}^{1/m}\right)^{-m} + \left(1 - \widetilde{W}^{1/m}\right)^m - 2 \right], \quad (24)$$

$$\gamma_{W,M-vG}(W) = \gamma_s \widetilde{W}^{1/2} \left[1 - \left(1 - \widetilde{W}^{1/m}\right)^m \right]^2. \quad (25)$$

These functions monotonically increase with liquid moisture content W , and are zero in the origin ($W = W_r$). The parameters in the above formulae are taken from [27]: $m = 0.272$, $\gamma_s = 3.78 D_{sec}^{-1} \text{ ms}^{-1}$ (hydraulic conductivity of saturated soil), $\alpha = 0.7 \text{ m}^{-1}$, $W_r = 0.149$, $W_s = 0.7$, where $D_{sec} = 86400 \text{ s}$ is a day duration.

2.2. General Approach of the Study

In this paper, we present two methods for PDE identification: classical method and ANN-based approach. We compare them in terms of wall-clock computational time and accuracy. In both methods, the coefficients of PDEs are sought as those minimizing the PDE residual $\epsilon_{h\tau}$, where the $B_{h\tau}$ and $A_{h\tau}$ operators correspond to central differences in space and 1st order scheme in time.

Key Differences in Classical and ANN-based Methods. The methods presented in this study have a similar structure, differing in the components:

- loss function, which is a measure of $\epsilon_{h\tau}$ deviation from zero in classic method and is a more complex metrics in a case of ANN-based method,
- explicit form of coefficients $\lambda_T, \lambda_W, \gamma$ as functions of soil moisture W ,
- method for the optimization of parameters in the assumed coefficient functions of W .

In the later sections, the detailed description of classical and ANN-based methods highlight these features.

2.3. Classical Method

2.3.1. Loss function

The loss function in classical method is defined as RMSE of $\epsilon_{h\tau}$ computed in time and space, where the boundary mesh nodes are excluded, i.e.:

$$\mathcal{L}_{CM} = \left[\frac{1}{(n_z - 1)n_t} \sum_{i=2}^{n_z} \sum_{j=1}^{n_t} (\epsilon_{h\tau,i}^j)^2 \right]^{1/2}, \quad (26)$$

with i standing for index of depth level, j denoting the time instant, n_t being the number of timesteps (in classical method, $n_t = 24$, corresponding to $\Delta t = 1$ h).

2.3.2. Assumed form of solution

Having data on the measured evolution of physical variables (soil temperature T and moisture W) we *always* do not know the real functional forms of $\lambda_T(W), \lambda_W(W), \gamma(W)$, so that *any* functional form which is assumed during inverse problem solution differs from that provided by nature. The indirect corroboration of this statement is provided by existence of ~ 100 pairs of explicit functions $\lambda_W(W), \gamma(W)$ reported in literature, none of which performs best on all the empirical datasets. At the same time, it is generally accepted that the heat conduction (15) and Richards (16) equations are exact. In the classical method of PDE identification, the *difference* between real and assumed types of PDE coefficients inevitable in real applications is imitated by using the widespread representations of heat conductivity, liquid water diffusivity and hydraulic conductivity which are *different* from Cote-Konrad and Mualem-van Genuchten formulae, used to compute reference solution, respectively. At the same time, using the functional form of $\lambda_T(W), \lambda_W(W), \gamma$ in PDE identification algorithm which are *the same* to ones used to produce reference direct problem solution allows to check the correctness of the gradient-descent algorithm in classical method as the latter allows to provide exact inverse problem solution in this case.

For solution of inverse temperature conduction problem we use the Johansen parameterization of heat conductivity [24], which involves the normalized heat conductivity concept (20), and the geometric mean approach for conductivity coefficient of saturated soil (22), however, proposes the different (compared to C-K) formulations for dry soil conductivity and Kersten number (hereafter referred to as "Joh"):

$$\lambda_{T,dry} = \frac{C_1 \rho_{dry} + C_2}{\rho_p - C_3 \rho_{dry}}, \quad (27)$$

$$Ke = \widetilde{W}, \quad (28)$$

where dry soil density ρ_{dry} and mean soil particles density ρ_p are expressed in g cm⁻³, and $C_1 = 0.135$, $C_2 = 0.0647$, $C_3 = 0.947$ are empirical constants. The second equation in a set above has been suggested for fine soils.

For identification of the Richards equation (16), the Brooks–Corey parameterization [7] (hereafter referred to as “B–C”):

$$\lambda_{W,B-C}(W) = \frac{\gamma_s |\Psi_{max}| b}{W_s - W_r} \widetilde{W}^{b+2}, \quad (29)$$

$$\gamma_{B-C}(W) = \gamma_s \widetilde{W}^{2b+3}, \quad (30)$$

and the Gardner parameterization [15] (hereafter denoted as “Gard”):

$$\lambda_{W,Gard} = \frac{\gamma_s a^{1/c}}{c} \widetilde{W}^{1/2} W^{\frac{c+1}{c}}, \quad (31)$$

$$\gamma_{Gard} = \gamma_s \widetilde{W}^{1/2} W^{\frac{2c+2}{c}}, \quad (32)$$

are involved, a , b , c are empirical constants to be calibrated, see details in Section 2.3.3.

2.3.3. Optimization method

As the functional forms of λ_T , λ_W , γ are defined above, the task of PDE identification is reduced to finding optimal values of parameters in formulas (27)–(28), (24)–(25), (29)–(30), (31)–(32). These parameters are sought using gradient descent method in Barzilai–Borwein modification, having the form [6]:

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \Gamma_n \nabla \mathcal{L}_{CM}(\mathbf{x}^n), \quad (33)$$

$$\Gamma_{n,0} = \frac{|(\mathbf{x}^n - \mathbf{x}^{n-1})^T (\nabla \mathcal{L}_{CM}(\mathbf{x}^n) - \nabla \mathcal{L}_{CM}(\mathbf{x}^{n-1}))|}{\|(\nabla \mathcal{L}_{CM}(\mathbf{x}^n) - \nabla \mathcal{L}_{CM}(\mathbf{x}^{n-1}))\|^2}. \quad (34)$$

Here, \mathbf{x}^n is a vector of parameters at n -th iteration, vertical index T stands for vector transpose. The gradient of the loss function $\nabla \mathcal{L}(\mathbf{x}^n)$ at each iteration is found by finite-differencing, gradient descent rate Γ_n is set to $\Gamma_{n,0}$ in conventional BB method, and in this study we use the stabilized Barzilai–Borwein method [8], which introduces the following limiting of the descent rate:

$$\Gamma_n = \min(\Gamma_{n,0}, \Gamma_n^{stab}), \quad (35)$$

$$\Gamma_n^{stab} = \frac{\Delta}{\|\nabla \mathcal{L}_{CM}(\mathbf{x}^n)\|}, \quad (36)$$

where Δ is maximal value allowed for $\|\mathbf{x}^{n+1} - \mathbf{x}^n\|$. This method is referred to as BBstab, and is free from large “jumps” of parameter vector \mathbf{x}_n during iteration process, which are known to happen in conventional BB algorithm, and were observed in the course of Richards equation identification process in our study. As the physical parameters entering \mathbf{x} have different scales, the equation being identified is rewritten in terms of parameters, normalized by respective maximal values, so that \mathbf{x} is sought in a unit hypercube, and $\Delta \ll 1$.

The loss function $\mathcal{L}_{CM}(\mathbf{x})$ is defined as a RMS of finite-difference equation residual at a given set of test coefficients taken over all time and depth combinations of the mesh (equation (26)).

The parameter vector is dependent on a PDE being identified:

- $\mathbf{x} \doteq (p, \lambda_{sp}, W_r, W_s)$ in C–K case of temperature equation (15),
- $\mathbf{x} \doteq (p, \lambda_{sp}, \rho_p, W_r, W_s, \rho_{dry})$ in Joh case of temperature equation,

- $\mathbf{x} \doteq (m, \gamma_s, \alpha, W_r, W_s)$ in M-vG case of Richards equation (16),
- $\mathbf{x} \doteq (b, \gamma_s, \Psi_{max}, W_r, W_s)$ in B-C case of Richards equation,
- $\mathbf{x} \doteq (c, \gamma_s, a, W_r, W_s)$ in Gard case of Richards equation.

The criterion to stop iterations used in our study is satisfaction of any of the two inequalities: \mathcal{L}_{CM} changes negligibly with iteration number or $n > n_{max}$. The first guess \mathbf{x}_0 significantly affects the number of iterations needed to converge to true solution. Moreover, given that some of parameters in λ_W and γ vary between soil types orders of magnitude (e.g., γ_s), the initial guess far from optimal value leads to large error in solution. In the numerical experiments presented below, we use the values $\mathbf{x}_0 = 0.5 * \mathbf{x}_{true}$, $\mathbf{x}_1 = 0.6 * \mathbf{x}_{true}$ (Barzilai–Borwein method needs first two points in \mathbf{x} -space to start iterations by formula (33)–(34)), which ensured the correct convergence. Alternatively, to compute \mathbf{x}_1 , one may use (33) with some small Γ_0 . Other choices of initial guesses (e.g., $\mathbf{x}_0 = 0.1 * \mathbf{x}_{true}$, $\mathbf{x}_1 = 0.2 * \mathbf{x}_{true}$; $\mathbf{x}_0 = 1.9 * \mathbf{x}_{true}$, $\mathbf{x}_1 = 1.8 * \mathbf{x}_{true}$), have been tested and delivered comparable results in accuracy and number of iterations. In addition to numerical experiments, where all parameters listed above were optimized, a set of simulations were conducted, where all but two parameters were fixed to known reference values, and the these two were optimized only. It was found that $n_{max} = 100$ ensures convergence (i.e., negligible change of \mathcal{L}_{CM} with iteration number) of the algorithm for both PDEs studied in this paper and both types of optimization.

2.4. Neural-Network-Based Solution

2.4.1. PDE identification as a differentiable inverse problem

In this study, we consider PDE identification as an inverse problem of regression type (see Problem Specification section). In both cases of simulated evolution and a solution given by regular observations, we refer to the given evolution as the true one. However, when using the true evolution as a reference, one still cannot apply routine data science approach for approximating the coefficients λ_T , λ_W and γ with ANNs. Instead, in case of ANN-based method, we propose to minimize the error of an integrated evolution within the gradient minimization approach given the coefficients λ_T , or λ_W and γ approximated by ANNs that are essentially differentiable parametric functions of potentially unlimited expressive power. More precisely, we propose minimizing PDE residual or, equivalently, the error of the tendencies $\frac{\partial T}{\partial t}$ (in case of heat diffusion equation) and $\frac{\partial W}{\partial t}$ (in case of Richards equation). The tendencies are estimated using finite-differences approximation of the equations (5) and (6). Finite-differencing is a differentiable operation as well as loss function characterizing errors of tendency estimates. Thus, there is a straight-forward way for the computation of the gradients of the loss function w.r.t. parameters of the functions modeling λ_T , or λ_W and γ . In Fig. 1, we present our ANN-based approach in a form of data-flow and operations scheme for Richards equation as a demonstrative example. Due to similarity of the derivatives inference, optimization procedures and regularizations applied in cases of heat diffusion equation and Richards equation, we further use X as a substitute correspondingly for W or T (e.g., see Fig. 1). We also present the formulae considering Richards equation case with λ_W and γ terms further in this section. At the same time, the derivatives inference, the regularizations and optimization procedure are the same for λ_T in case of heat diffusion equation.

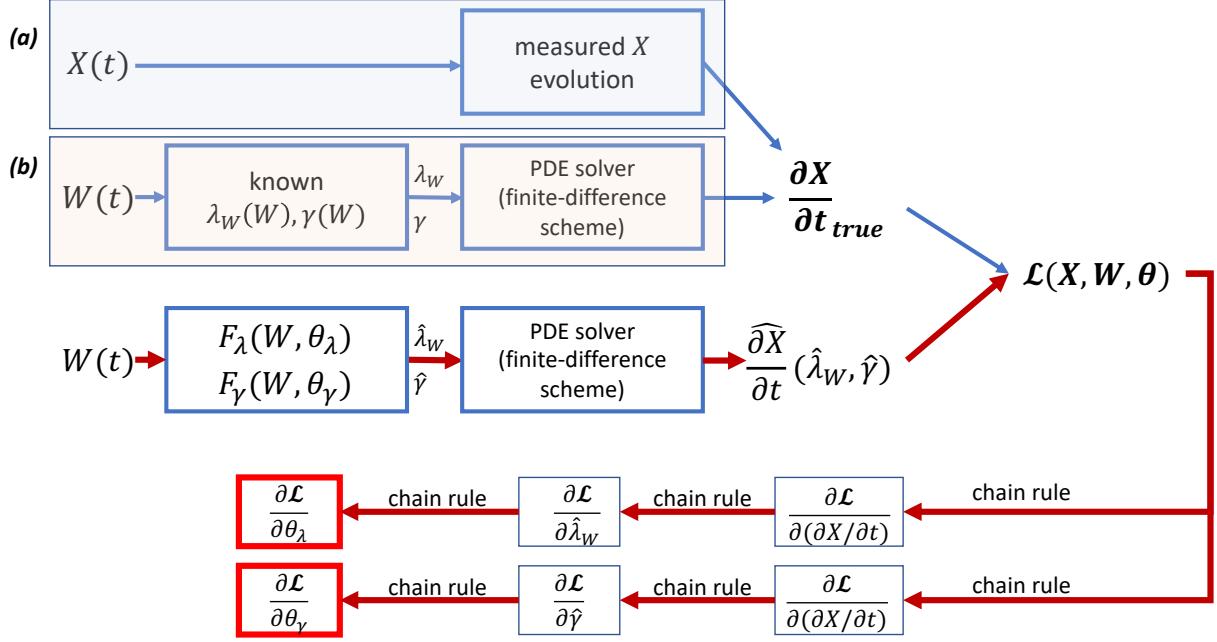


Figure 1. The PDE identification approach using ANNs F_λ and F_γ for Richards equation (16) in case of either (a) measured evolution, or (b) modeling the reference evolution using known coefficients λ_W and γ . Here X is a substitute for W , the similar algorithm is applied for temperature equation ($X = T$). The bold red arrows indicate the computational graph that involves chain rule (a.k.a. backpropagation) for evaluating the gradients of loss function \mathcal{L} w.r.t. the parameters θ_λ of F_λ and θ_γ of F_γ . In this paper, the case (b) is considered only

2.4.2. Assumed form of solution

In this method, the approximations of the coefficients λ_T , λ_W and γ of the heat conduction (15) and Richards equations (16) are represented by artificial neural networks (ANNs) denoted further as either F_{NN,λ_T} , F_{NN,λ_W} and $F_{NN,\gamma}$ or F_{λ_T} , F_{λ_W} and F_γ (for the sake of clarity). The architecture of the ANNs is the same for all the cases. It is the fully-connected ANNs (a.k.a. multilayer perceptron) with six layers. The layers are wide for the networks to have high enough expressive power for approximating functions like eqs. (20), (24) and (25). We optimized the structural hyperparameters of the networks ensuring the capability of approximating the target functions by solving simple supervised problem with target values generated by “true” forms like eqs. (20), (24) and (25). In order to improve convergence, we used *Mish* activation function [34] for all layers except the last one, where we used linear activation.

2.4.3. Loss function

The basic part of the loss function in this study is a weighted sum of the three components: MAPE of tendency, MSE (mean square error) and MAE (mean absolute error) of PDE residual:

$$\mathcal{L} = \alpha_{mse} \frac{1}{N} \sum \epsilon_{h\tau}^2 + \alpha_{mape} \frac{1}{N} \sum \left| \frac{(\widehat{\partial X/\partial t})_{h\tau} - (\partial X/\partial t)_{h\tau,true}}{(\partial X/\partial t)_{h\tau,true} + \epsilon} \right| + \alpha_{mae} \frac{1}{N} \sum |\epsilon_{h\tau}|, \quad (37)$$

where N is a number of mesh nodes over which the PDE residual and discrepancy between test and reference tendencies are summed, X stands for T or W , depending on equation being identified, the subscript “ $h\tau$ ” indicates variables discretized on the mesh, the hat $(\widehat{\dots})$ denotes

values calculated using test PDE coefficients represented by ANNs, and $\epsilon = 10^{-8}$ is a number added for numerical stability. Note that $(\widehat{\partial X / \partial t})_{h\tau} - (\partial X / \partial t)_{h\tau, true} = \epsilon_{h\tau}$.

Zero-Point Regularization. In order to stabilize the ANN training and improve the convergence, we complemented the loss function (37) with the regularization terms informed by physical constraints. First, one may note that dry soil should not conduct liquid water. This is in agreement with the reference solutions for λ_W and γ given by eqs. (24) and (25). Thus, $\lambda_W(W=0)=0$, and $\gamma(W=0)=0$. We implemented the respective regularization loss terms as:

$$\mathcal{L}_{reg,zp} = \alpha_{zp,\lambda} \times |F_{NN,\lambda_W}(W=0)|, \quad (38)$$

$$\mathcal{L}_{reg,zp} = \alpha_{zp,\gamma} \times |F_{NN,\gamma}(W=0)|, \quad (39)$$

where $\alpha_{zp,\lambda}$ and $\alpha_{zp,\gamma}$ are regularization coefficients related to the zero-point values of λ_W and γ approximated by the ANNs $F_{NN,\lambda_W}(W)$ and $F_{NN,\gamma}(W)$, correspondingly.

Known Points Regularization. In a real-world application case, one may improve the convergence with a regularization encouraging the ANNs to predict specific known (e.g., measured) values of PDE coefficients for a set of argument values X_{kp} . We call the respective loss term the known-points regularization:

$$\mathcal{L}_{reg,kp} = \alpha_{kp,\lambda} \times \sum_i |F_{NN,\lambda_W}(W_{kp,i}) - \lambda_W(W_{kp,i})|, \quad (40)$$

$$\mathcal{L}_{reg,kp} = \alpha_{kp,\gamma} \times \sum_i |F_{NN,\gamma}(W_{kp,i}) - \gamma(W_{kp,i})|, \quad (41)$$

where $\alpha_{kp,\lambda}$ and $\alpha_{kp,\gamma}$ are regularization coefficients related to the values of λ_W and γ correspondingly in a limited set of known points $W_{kp,i}$ approximated by the neural networks $F_{NN,\lambda_W}(W)$ and $F_{NN,\gamma}(W)$; i enumerates measured tuples of W_{kp} , λ_W and γ . In our study, the number of known points is 3.

Derivative Penalty. One may also note that the coefficients λ_W and γ are strictly increasing functions, which is a case for all analytical representations (including Mualem-van Genuchten, Brooks-Corey and Gardner) for physical reasons. According to this constraint, we introduce corresponding physics-guided regularization terms that we call derivative penalty:

$$\mathcal{L}_{reg,dp} = \alpha_{dp,\lambda} \times \sum_{\frac{\partial F_{NN,\lambda_W}(W)}{\partial W} < 0} \left| \frac{\partial F_{NN,\lambda_W}(W)}{\partial W} \right|, \quad (42)$$

$$\mathcal{L}_{reg,dp} = \alpha_{dp,\gamma} \times \sum_{\frac{\partial F_{NN,\gamma}(W)}{\partial W} < 0} \left| \frac{\partial F_{NN,\gamma}(W)}{\partial W} \right|. \quad (43)$$

The regularizations presented in eqs. (38)–(43) are “soft”, meaning they do not necessarily deliver the desired properties of the approximations. Thus, one may note that the derivative penalties in eqs. (42) and (43) do not necessarily guarantee strictly ascending ANN-approximations for λ_W and γ . In order to additionally encourage the ascending approximations, we complement

the loss function with one more term which encourages the networks to output less value at the left margin of the X range compared to the output at the right margin of X range:

$$\mathcal{L}_{reg,lr} = \begin{cases} \alpha_{lr,\lambda} \times |F_{NN,\lambda_W}(X_{max}) - F_{NN,\lambda_W}(X_{min})| & \text{if } F_{NN,\lambda_W}(X_{max}) < F_{NN,\lambda_W}(X_{min}) \\ 0 & \text{otherwise} \end{cases} \quad (44)$$

$$\mathcal{L}_{reg,lr} = \begin{cases} \alpha_{lr,\gamma} \times |F_{NN,\gamma}(X_{max}) - F_{NN,\gamma}(X_{min})| & \text{if } F_{NN,\gamma}(X_{max}) < F_{NN,\gamma}(X_{min}) \\ 0 & \text{otherwise,} \end{cases} \quad (45)$$

where X_{min} and X_{max} are the lower and higher limits for X that are either inferred from measurements or set by a researcher conducting a computational experiment. In case of X being substitute for W , one may set the lower limit to 0 and higher one to the maximum value inferred from data. In case of X being substitute for T , both lower and higher limits may be set to the values inferred from data. Here one needs to pay attention to the way we inject regularization terms into loss function. In particular, one should pay attention to derivative penalty $\mathcal{L}_{reg,dp}$, since gradients of this loss term w.r.t. network parameters includes second-order derivatives $\frac{\partial F_{NN}(W,\theta)}{\partial W \partial \theta}$, which is computationally expensive. In order to save computational power, we do not compute the derivative penalty using the W and T profiles themselves since the number of data points in this case is `batch_size*n_z`. Instead, we compute derivative penalty using additional sample of T and W values distributed uniformly in the range $[T_{min}, T_{max}]$ and $[W_{min}, W_{max}]$ correspondingly. The same applies to the regularization terms $\mathcal{L}_{reg,zp}$, $\mathcal{L}_{reg,kp}$. In this case, the number of data points is managed by a researcher. We set this number to n_z . Here, T_{min} and T_{max} are the maximum and minimum values of temperature in the limited set of measured known data points; W_{min} and W_{max} – the same for liquid water content W . This way, we managed to improve the performance approximately 6-8 times compared to that when using computations with full training set of W and T profiles.

Rescaling Neural Networks Output. ANNs are known to be strongly overparameterized and thus having too high expressive power for representing functions of relatively simple form. In the case of limited support of target value distribution, one may need to restrict the distribution of the output of the network. One way to do so is using a finite-bounds activation function for the output layer, e.g., tanh or sigmoid. These functions are known to saturate which may cause optimization difficulties and low approximation accuracy in some argument ranges. In our study, we use dynamic min-max rescaling of the network output for both λ_W and γ networks:

$$\zeta = F_{NN}(X), \quad (46)$$

$$\tilde{\zeta} = \zeta_{min,true} + \frac{\zeta - \zeta_{min}}{\zeta_{max} - \zeta_{min}} \times (\zeta_{max,true} - \zeta_{min,true}), \quad (47)$$

where ζ is the substitute for λ_W or γ ; $\tilde{\zeta}$ is the substitute for λ_W or γ rescaled using min-max scheme; $\zeta_{min,true}$ and $\zeta_{max,true}$ are the minimum and maximum values of true λ_W or γ in the $[X_{min}, X_{max}]$ range; ζ_{min} and ζ_{max} are the minimum and maximum values of ANN-based approximations for λ_W or γ in the $[X_{min}, X_{max}]$ range. One may note that the values ζ_{min} and ζ_{max} may not be the ones $F_{NN}(X_{min})$ and $F_{NN}(X_{max})$ due to inaccurate meeting of the strictly-ascending requirement by the neural network. On the contrary, here we consider once again the

known-points values for $\zeta_{min,true}$ and $\zeta_{max,true}$ to be the $\zeta_{X_{min,true}}$ and $\zeta_{X_{max,true}}$. These values are guaranteed to be minimum and maximum ones due to strictly ascending λ_W and γ .

Initial Guess. There is a well-known issue related to the choice of initial guess for the ANN parameters θ_{λ_T} , θ_{λ_W} and θ_γ of the networks F_{NN,λ_T} , F_{NN,λ_W} and $F_{NN,\gamma}$. One may even choose an initialization that would prevent a network from training. In [33], the importance of proper initialization is demonstrated. In this paper, the authors demonstrated that one needs to initialize deep ANNs with ReLU activations using random variables drawn from a uniform or normal distribution with parameters dependent on layers' sizes. This is done for the variance of activations of hidden layers and the output layer to remain nearly constant, which prevents output and gradients variances from vanishing or exploding. The most successful initialization at the moment is Kaiming He [21]. However, the so-called “gain” parameter in Kaiming He initialization [21] is computed using the size of layers’ weight tensors with the assumption of either ReLU or Leaky ReLU nonlinearity. In our study, we use *Mish* activation functions, thus, there may be a need for a derivation of proper gain value. We did not conduct this research. Instead, we assumed that *Mish* has the effect on the distributions of activations similar to ReLU due to their large-scale similarity. Thus, in our study, we used Kaiming He initialization with gain value similar to the one computed with ReLU nonlinearity assumption. The results of our ANN-based method presented in Section 3.2 confirm the validity of this decision.

Optimization of Hyperparameters. In this study, we demonstrate that the balance between the multipliers standing before the loss function terms is crucial for the stability of training, for Richards equation (16) identification. In order to stabilize the training, we explored the space of hyperparameters.

Learning Rate Schedule. In our study, we exploited stochastic gradient-based optimization procedure, which implies iterative increment of network’s parameters with adjusted loss gradients multiplied by a small factor namely learning rate. In recent studies, the importance of learning rate scheduling was shown [31]. However, we found that with the stability improvements presented further in this section, there is no need to apply sophisticated learning rate schedules. Thus, we employ exponential decay of learning rate. Typically, the starting leaning rate is 10^{-4} , and the decay rate is 10^{-2} . Thus, to the end of the training, learning rate decreases to 10^{-6} level.

2.4.4. Optimization and implementation details

In case of ANN-based PDE identification of our study, in the numerically-simulated evolution, the number of z -levels is $n_z = 1000$, and the number of time steps is $n_t = 8640$ ($\Delta t = 10$ s) which corresponds to 24 hours of modeled time. The optimization of ANNs implies stochastic optimization algorithms (in this study, we employ Adam [26]). For this algorithm, to evaluate loss function, we implemented data sampler, which draws `batch_size` time steps from the reference PDE solution dataset.

In our study, we found that Adam optimization is not enough for the stable loss function minimization and for reaching the solution with low uncertainty in case of Richards equation problem. Inverse PDE problems are known for sensitivity of the solution to input data. One more issue compromising the solution stability is the mixed precision computations which requires particular attention in case of small and large loss values and gradient values. There are known

approaches of loss scaling and corresponding gradient inverse scaling by NVIDIA [1]. However, we faced the issue of partly implemented method of NVIDIA loss scaling in Tensorflow 1.15, which fails to run in case of provided software and hardware stack (see Section 2.5). In order to achieve stable and converging training of ANNs for λ_W and γ approximations, we implemented several additional approaches described below.

Scaling Scheme for Scalars and Tensors. Optimization of ANNs using mixed precision floating-point calculations is complicated by the narrow range of values represented in FP16. General description of computational issues related to FP16 is presented on NVIDIA web-site. The NVIDIA loss scaling [1] is implemented in detail in Tensorflow 2.x, and is *partially* implemented in Tensorflow 1.15. On Huawei Ascend platform, Tensorflow 1.15 was more matured at the time that the study was conducted, thus, we developed a new scheme for loss scaling which may be characterized as more “gentle” yet efficient. The same scheme may be employed for gradient scaling in a case the tensors are found vanishing.

The main goal of the scheme is to preserve the scale of loss function values close the order of 10^0 . To do so, we introduce the following variables:

1. a factor for exponential moving average f_{ema} . We typically set it to 0.1;
2. loss which is subject to scale L . This loss value is the one we compute directly using networks outputs with equations (37)–(45);
3. current loss scale s ;
4. scaled loss L_s ;
5. $\log_2(L_s)$, denoted as l_2 ;
6. exponentially moving averaged value of the loss L_{ema} ;
7. exponentially moving averaged value of $\log_2(L)$, denoted as $l_{2,ema}$;
8. gradients g_s computed using backpropagation implemented in auto-differentiation schemes of Tensorflow or other autodiff frameworks. These gradients are calculated from scaled loss values L_s ;
9. inverse-scaled gradients g .

The operation of loss scaling and gradients inverse scaling are linear operations:

$$L_s = L \times s, \quad (48)$$

$$g_s = \nabla_\theta L_s, \quad (49)$$

$$g = g_s / s, \quad (50)$$

where θ is a set of parameters (a.k.a. weights) of ANNs involved in loss function computations, and ∇_θ is a gradient over this parameter set.

The operations for scale s adjustment are the following:

$$L_{ema}^{(\tau)} = L_{ema}^{(\tau-1)} \times (1 - f_{ema}) + L^{(\tau)} \times f_{ema}, \quad (51)$$

$$l_2^{(\tau)} = \log_2 L_{ema}^{(\tau)}, \quad (52)$$

$$l_{2,ema}^{(\tau)} = l_{2,ema}^{(\tau-1)} \times (1 - f_{ema}) + l_2^{(\tau)} \times f_{ema}, \quad (53)$$

$$s^{(\tau)} = 2^{-\lfloor l_{2,ema}^{(\tau)} \rfloor}, \quad (54)$$

where $\lfloor x \rfloor$ denotes rounding to a nearest integer; τ is the number of iteration of adjustment (typically coincides with the number of optimization iteration). With this scheme, the loss scale remains to be of order 10^0 .

With the same scheme, one may apply gradient scaling for its expected absolute value to be of order 10^0 . To do so, the only corrections to the scheme presented above are the following: instead of L , one needs to use the scale of $\tilde{L} = \text{mean } |g_\theta|$, where g is gradient tensor of a neural network w.r.t. parameters tensor of a particular layer of a network. In case of some other variable of interest A subjected to scaling, the scheme looks the following:

$$A_s = A \times s_A, \quad (55)$$

$$\tilde{A}_{ema}^{(\tau)} = A_{ema}^{(\tau-1)} \times (1 - f_{ema}) + A^{(\tau)} \times f_{ema}, \quad (56)$$

$$a_2^{(\tau)} = \log_2 \tilde{A}_{ema}^{(\tau)}, \quad (57)$$

$$a_{2,ema}^{(\tau)} = a_{2,ema}^{(\tau-1)} \times (1 - f_{ema}) + a_2^{(\tau)} \times f_{ema}, \quad (58)$$

$$s_A^{(\tau)} = 2^{-\lfloor l_{2,ema}^{(\tau)} \rfloor}, \quad (59)$$

where \tilde{A} is sample mean of $|A|$ in case of A being a tensor with rank greater than zero, and $\tilde{A} = A$ in case of A being a scalar. This way, we scale the gradients and values of $(\partial X / \partial t)_{h\tau}$ preventing them from becoming small enough to vanish to FP16-zero level in case of quadratic loss components $\text{MSE} \left[(\widehat{\partial X / \partial t})_{h\tau}, (\partial X / \partial t)_{h\tau, true} \right]$.

The gradient scaling scheme has an impact on the optimization procedure. First, using this scheme, one introduces noise into stochastic optimization when the scale s is changing, since the change is made in a discrete multiplicative manner (see eq. (54)). Second, when one applies gradient scaling presented in this study, the magnitude of gradients is preserved of order 10^0 . In common gradient optimization algorithms, the gradients' magnitudes are assumed to decrease over time, resulting in sampling of the loss landscape with increasing resolution and, thus, finding the solution with increasing accuracy. This behavior is essentially equivalent to learning rate decrease, which does not take place in case gradient scaling applied. Thus, one needs to pay additional attention to setting the learning rate schedule, which may play the role of a factor increasing the resolution of loss landscape sampling.

Reducing the Interaction between Loss Sum Components. There is one more issue of the ANN-based identification of the Richards equation related to smooth nature of γ function. The finite-difference approximation of the gravitational infiltration term in the Richards equation (16) is:

$$\begin{aligned} \left(\frac{\partial W}{\partial t} \right)_{\gamma,i} &= \left(\frac{\partial \gamma(W)}{\partial z} \right)_i \approx \frac{\gamma(W_{i+1/2}) - \gamma(W_{i-1/2})}{\Delta z_i}, \\ \text{MSE} \left[\left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau}, \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true} \right] &= \frac{1}{N \Delta z} \sum_{i=1}^N \left[\left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau,i} - \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i} \right]^2, \\ \text{MAE} \left[\left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau}, \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true} \right] &= \frac{1}{N \Delta z} \sum_{i=1}^N \left| \left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau,i} - \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i} \right|, \\ \text{MAPE} \left[\left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau}, \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true} \right] &= \frac{1}{N \Delta z} \sum_{i=1}^N \left| \frac{\left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau,i} - \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i}}{\left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i}} \right|. \end{aligned} \quad (60)$$

Here, we replaced residual by difference of tendencies, i.e., $\epsilon_{h\tau} = (\widehat{\partial W / \partial t})_{h\tau} - (\partial W / \partial t)_{h\tau, true}$ and Δz_i with Δz for the uniform z mesh, i enumerates individual components of loss sums. One may note that the consequent terms of MSE, MAE and MAPE sums interact between each other

due to smoothness of the γ function (see details in Appendix A). We also found empirically that the gradients of these loss function terms are zero almost everywhere except for small subset of depths including the boundaries at $z = 0$ and $z = H$ (Fig. 2a).

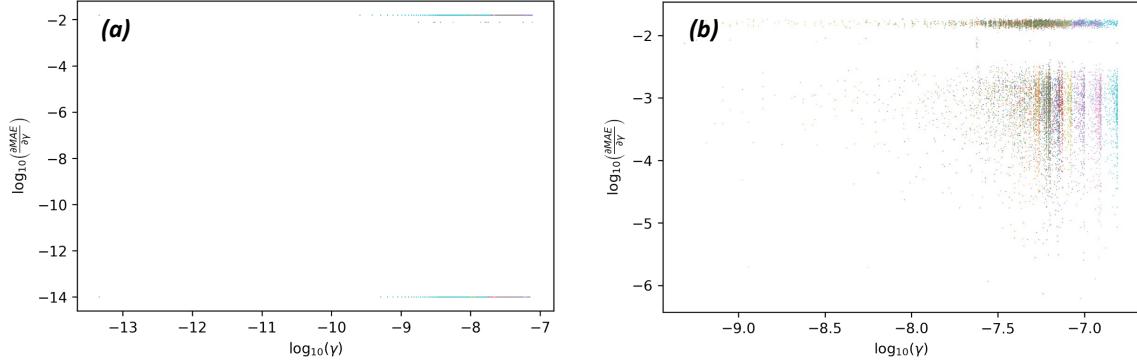


Figure 2. MAE_γ loss gradients w.r.t. γ as individual sum terms vs. γ : (a) in case of standard MAE_γ formulation; (b) in case of random re-weighting of individual MAE_γ sum terms according to eq. (61). The logarithmic scale uses additive log-stability term 10^{-14}

In order to alleviate this issue, we introduce the technique of random re-weighting of loss sum components. We reformulate the loss function terms the following way:

$$\begin{aligned} \mathcal{L}_{\text{MSE}} &= \frac{1}{N\Delta z} \sum_{i=1}^N \mathcal{W}_{\text{MSE},i} \times \left[\left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau,i} - \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i} \right]^2, \\ \mathcal{L}_{\text{MAE}} &= \frac{1}{N\Delta z} \sum_{i=1}^N \mathcal{W}_{\text{MAE},i} \times \left| \left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau,i} - \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i} \right|, \\ \mathcal{L}_{\text{MAPE}} &= \frac{1}{N\Delta z} \sum_{i=1}^N \mathcal{W}_{\text{MAPE},i} \times \left| \frac{\left(\widehat{\frac{\partial W}{\partial t}} \right)_{\gamma,h\tau,i} - \left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i}}{\left(\frac{\partial W}{\partial t} \right)_{\gamma,h\tau,true,i}} \right|. \end{aligned} \quad (61)$$

Here, we generate the weights as Gaussian random variable: $\mathcal{W} \sim \mathcal{N}(1, \sigma^2)$, where σ may be optimized during hyperparameters optimization stage. In our study, we set $\sigma = 0, 1$. As a result, the loss gradients became more informative and are not zero, as shown in Fig. 2b.

2.4.5. Further details of the ANN-based solution for Richards equation

We found that λ_W -network converges much faster compared to the one approximating γ due to the strongly interacting loss sum components in the latter case described in the previous section. Moreover, the λ_W -network converges to a very good approximation with $\text{MAPE}(\lambda_W) \approx 10^{-2}$. Thus, in this study, we used the approach of consequent training of γ - and λ_W -networks. First, we trained λ_W -network with the γ -network initialized to output zeros. Then, after a fixed training period, we freeze the λ_W -network and optimize the one approximating γ .

2.5. Hardware and Software

Numerical experiments with classical method of PDE identification were performed on MacBook Pro, 2.3 GHz Intel Core i5, 8 Gb RAM, 2133 MHz, LPDDR3.

Computations using ANN-based approach were conducted on Atlas 800 Training Server (Model 9010) with 8-socket Huawei Ascend 910 NPUs, 16 Huawei 64-bit TaiShan CPU cores at 2.4 GHz. Each NPU provides three HCCS links and a maximum bandwidth of 90 Gbit/s. The server supports two CPUs with up to 3.8 GHz frequency, 38.5 MB L3 cache, and three 10.4 GT/s UPI links, up to 24 DDR4 RDIMMs, 16 GB, 32 GB or 64 GB per DIMM. The capacity of HBM is 32 GB with bandwidth 1228 Gbit/s.

In this study, we developed the software package implementing our ANN-based method using Python 3.5.7 [41] with Tensorflow 1.15 (within the `session.run()` approach and explicit computational graph definition) [2]. We also used numpy [20] and other python modules typical for data science data processing and visualization.

2.6. Metrics for Quality of the Inverse Problem Solution

The values of coefficients λ_W and γ range of orders of magnitude depending on the argument W , as well as corresponding tendency of the solution $\partial W/\partial t$ in Richards equation (16). Thus, in order to represent the accuracy of the inverse problem solution, we use several quality measures: RMSE(λ_X); MAPE(λ_X); RMSE(γ); MAPE(γ). We calculate these measures using conventional RMSE(.) and MAPE(.) formulations without any re-weighting. We also use the following error metrics: RMSE($(\partial X/\partial t)_{h\tau}$) which is equivalent to root mean squared $\epsilon_{h\tau}$, and MAPE($(\partial X/\partial t)_{h\tau}$) ($X = W, T$):

$$\mathcal{Q}_{rmse,\epsilon_{h\tau}} = \text{RMSE} \left[\left(\frac{\partial X}{\partial t} \right)_{h\tau} \right] = \sqrt{\frac{1}{N} \sum_1^N \epsilon_{h\tau}^2}, \quad (62)$$

$$\mathcal{Q}_{mape,\epsilon_{h\tau}} = \text{MAPE} \left[\left(\frac{\partial X}{\partial t} \right)_{h\tau} \right] = \frac{1}{N} \sum_1^N \left| \frac{\epsilon_{h\tau}}{\left(\frac{\partial X}{\partial t} \right)_{h\tau, true} + \epsilon} \right|. \quad (63)$$

3. Results and Discussion

3.1. Stabilized Barzilai–Borwein Method

First, we consider the loss function \mathcal{L}_{CM} of two parameters, where the rest parameters are fixed to reference values, and where \mathcal{L}_{CM} is computed at a regular grid 30×30 (Fig. 3 and 4). The range of parameters covers most of variability reported in soil science. Note the single minimum of loss function for each of three functional forms of test coefficients. The minimum in parameter space of reference C-K and M-vG form delivers exact solution of the inverse problem. However, in a case, where all (from 4 to 6) parameters are optimized, the minimum of \mathcal{L}_{CM} is no more unique (including a scenario of inverse problem solution, where C-K and M-vG forms are used), which is manifested by the fact that the argument of loss minimum found by BBstab method does depend on initial guess (not presented in this paper).

As a result of multiple minima of \mathcal{L}_{CM} in parameter space, optimal coefficient functions, provided by all analytical forms, significantly differ from the reference ones (Fig. 5ab, 6a–c, 7a–c). The thermal conductivity coefficient remains the same order of magnitude over the W range, and the absolute error does the same. Using the Joh formulation provides significantly larger errors compared to a case of using C-K representation. For Richards equation (16), the optimal test coefficients significantly deviate from reference ones as well, with absolute error increasing

towards W values close to saturation. Surprisingly, seeking Richards coefficients in M-vG way, does not necessarily provide the smallest MAPE and RMSE respective to “true” coefficients, compared to other scenarios (e.g., compare B-C and M-vG results for γ), which means the choice of initial M-vG parameter values close to exact ones is crucial for solution accuracy.

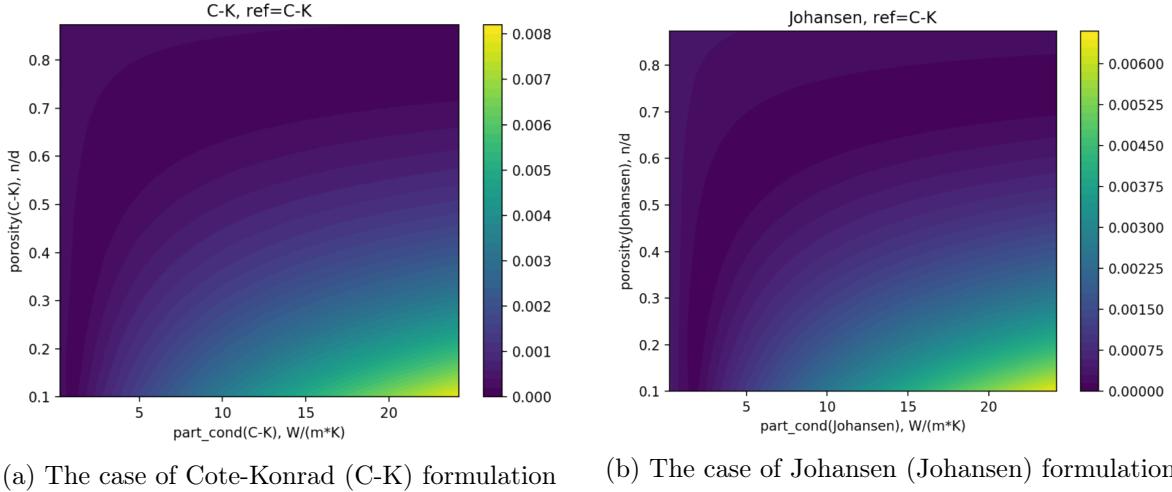


Figure 3. Loss function \mathcal{L}_{CM} of two parameters (porosity, p , and heat conductivity of soil particles, λ_{sp}) being optimized in the cases: Cote-Konrad (C-K, left) and Johansen (Johansen, right) formulas for thermal conductivity in test evaluation of thermal conductivity equation (15) residual $\epsilon_{h\tau}$, the realistic reference solution T of PDE being produced using C-K formulation

3.2. Neural-Network Method

Here, we present results of the optimization of ANNs within the framework described in Section 2.4, for the Richards (16) and temperature conductance (15) equations with realistic reference direct problem solution. We use soil column temperature and moisture profiles generated by direct solution of PDEs for 24 hours of model time. This way, we obtained 8640 profiles. For each of these profiles we apply our scheme described in Section 2.4. We use stochastic optimization with `batch_size=128` number of profiles per iteration, each containing $n_z = 1000$ number of z -levels.

We optimized some of the hyperparameters of the method using Optuna framework [3]. To optimize the hyperparameters, we compute quality metrics of the solutions with λ_T in case of heat conductance equation or λ_W, γ in case of Richards equation modeled by ANNs for each set of hyperparameters; the learning rate schedule was set to exponential decay with starting value `1e-4`, decay rate `1e-2` and number of steps 8192. We found that in this setup, the ANNs optimization can converge with reasonable quality, thus we have the opportunity to assess the quality metrics for each set of hyperparameters. We performed 3500 runs optimizing the following hyperparameters: $\alpha_{mse}, \alpha_{mape}, \alpha_{mae}, \alpha_{zp}, \alpha_{dp}, \alpha_{kp}$. A sample for each of these hyperparameters was generated using log-uniform sampling with the distribution support $[1, 10^{14}]$. As a result, we found the following hyperparameters that let us train our networks reaching appropriate quality $\mathcal{Q}_{rmse, \epsilon_{h\tau}}$ and $\mathcal{Q}_{mape, \epsilon_{h\tau}}$: $\alpha_{mse} = 10^{11}, \alpha_{mape} = 10, \alpha_{mae} = 1, \alpha_{zp} = 10^8, \alpha_{dp} = 10, \alpha_{kp} = 10^8$ (see (37), (38), (41), (42)). With these hyperparameters, we then trained the networks F_{NN, λ_W} and $F_{NN, \gamma}$ in case of Richards equation (16), and F_{NN, λ_T} in case of heat conductance problem (15).

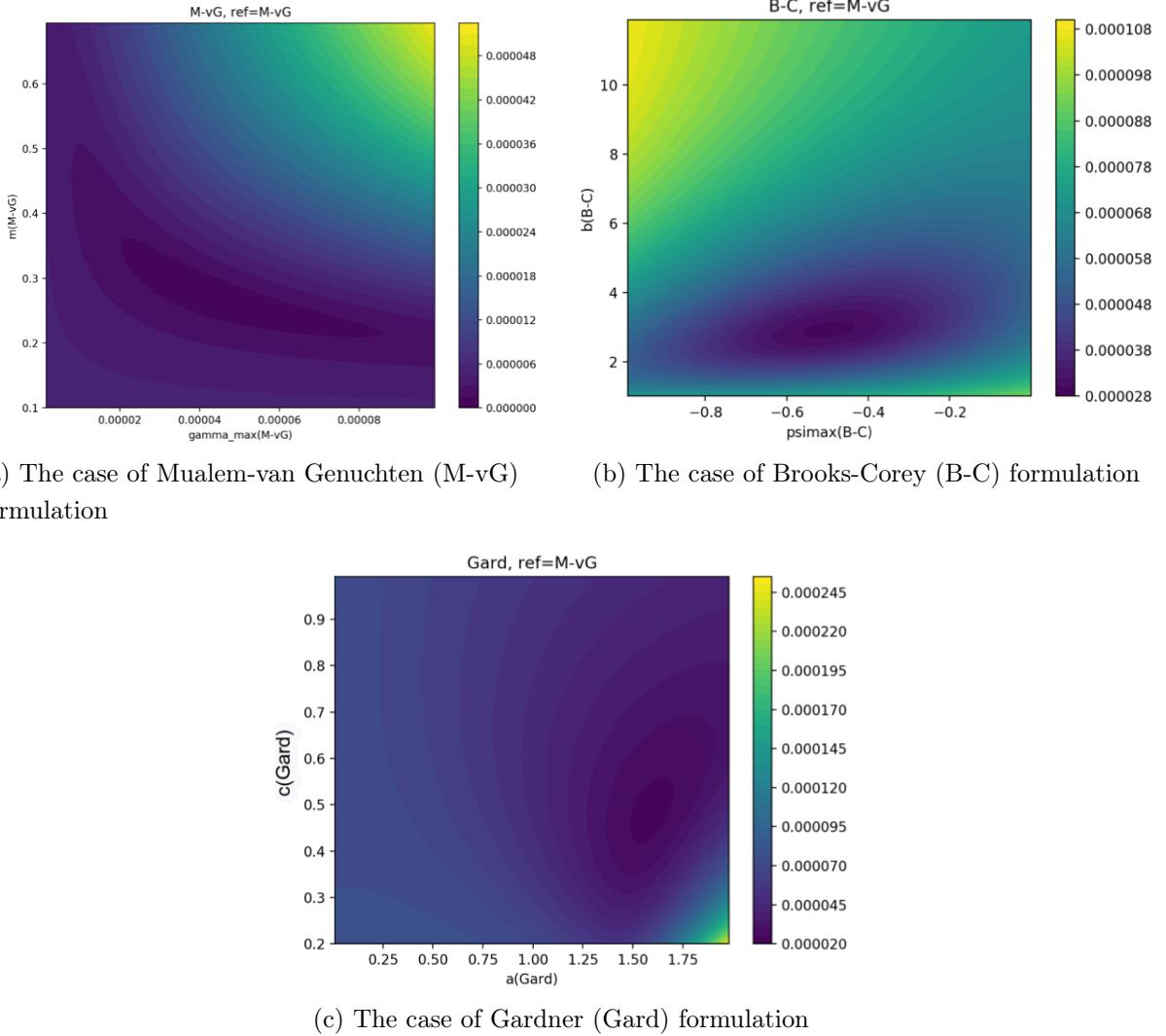


Figure 4. Loss function \mathcal{L}_{CM} of two parameters being optimized in the cases: Mualem-van Genuchten (M-vG, (a), parameters are γ_{max} , m), Brooks-Corey (B-C, (b), parameters are Ψ_{max} , b) and Gardner (Gard, (c), parameters are a , c) formulas for moisture diffusivity and hydraulic conductivity coefficients used in test evaluation of Richards equation (16) residual $\epsilon_{h\tau}$, the realistic reference PDE solution W being produced using M-vG formulation

Important part of our study was focused on computations in mixed precision (MP). We observed that the out-of-box MP settings resulted in the drop of accuracy comparing to single precision, but we managed to stabilize those using the black-list of operations which were prohibited to run in FP16: "Exp", "Tanh", "TanhGrad", "Div", "Sub", "Pow", "Add", "Mul", "Log". With this black-list and the stability improvements described in detail in Section 2.4, we managed to reach the quality of the approximation similar to FP32 computations (see results in Tab. 1 and 2) without negative impact to the computational efficiency. We also tested the computational efficiency (performance) in FP32 and mixed-precision. The quantitative results for heat conductance problem are presented in Tab. 3. Corresponding measures for Richards equation case are presented in Tab. 2 and 4. One may note, that in Tab. 4, we show the performance estimates in iterations per second. Here, one iteration corresponds to 128000 profile elements per second since the number of moisture profiles is exactly 128 per batch (iteration), and number of z -levels is 1000 for ANN-based experiments in this study.

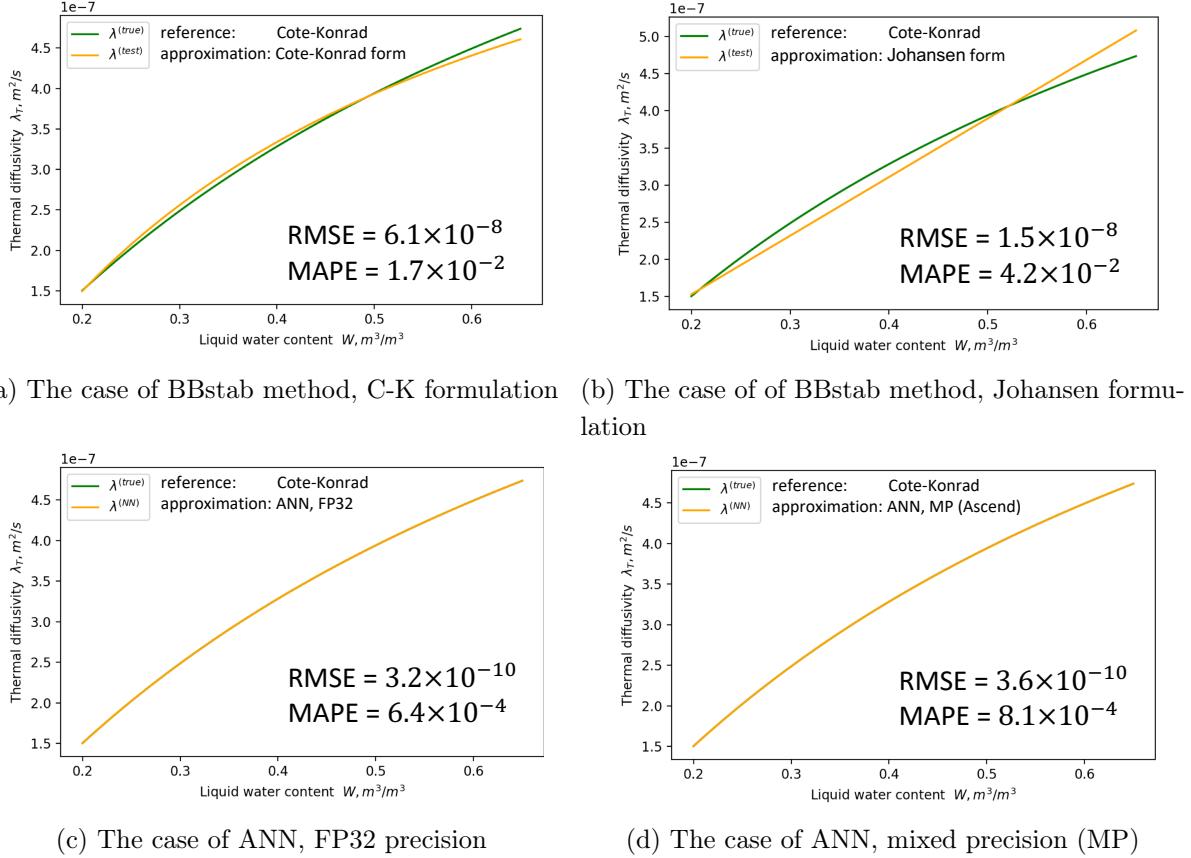
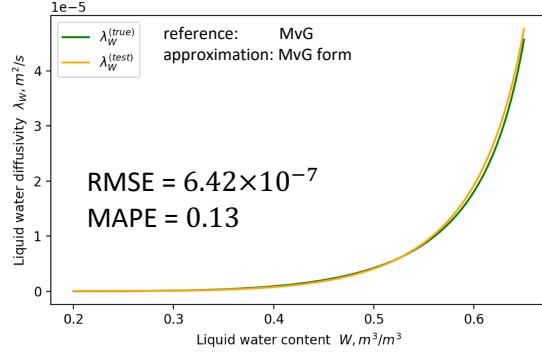


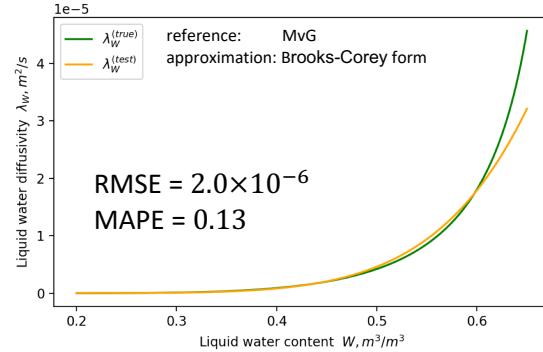
Figure 5. Thermal diffusivity coefficient λ_T for soil as a function of volumetric moisture content W being approximated (a) in a form given by C-K formulae in test solution of heat conduction equation using BBstab optimization procedure; (b) in a form given by Johansen formula in test solution of heat conduction equation using BBstab optimization procedure; (c) in a form of artificial neural network, FP32 precision computations; (d) in a form of artificial neural network, MP computations employed on Ascend compute platform. Here, we produce the reference solution using the C-K formulation

The results of PDE identification using ANN-based method are shown in Fig. 5cd, 6d, 7d. One may see excellent correspondence between the reference forms of the coefficients and the approximations made by ANNs.

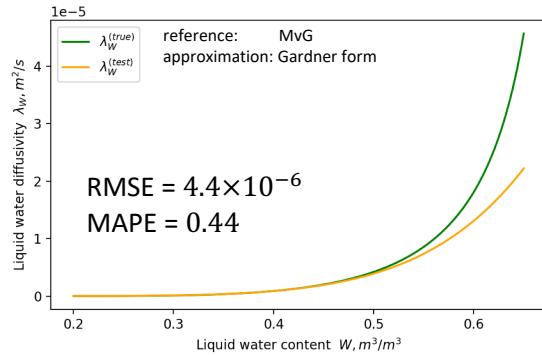
We found that ANNs optimization process may begin to diverge if one continues to run it long enough in case of Richards equation problem. Thus, there is a room for decision at which point to get the snapshots of the networks and corresponding quality measures. In case of real-world application, one cannot compute quality measures for the parameters λ_T , λ_W and γ due to unavailability of their ground truth. Thus, one cannot formulate the stopping rule for ANNs optimization based on $\text{RMSE}(\lambda_W)$, $\text{MAPE}(\lambda_W)$, $\text{RMSE}(\gamma)$ or $\text{MAPE}(\gamma)$. The only indicators one may use to judge on solution accuracy during optimization process are the measures of discrepancy between true and approximated PDE tendencies $\mathcal{Q}_{\text{rmse}, \epsilon_{h\tau}}$ or $\mathcal{Q}_{\text{mape}, \epsilon_{h\tau}}$. In our study, we used $\mathcal{Q}_{\text{rmse}, \epsilon_{h\tau}}$ as the indicator for the stopping rule, thus, in eqs. (1) and (2) we demonstrate the quality measures corresponding to optimal (minimal) $\mathcal{Q}_{\text{rmse}, \epsilon_{h\tau}}$



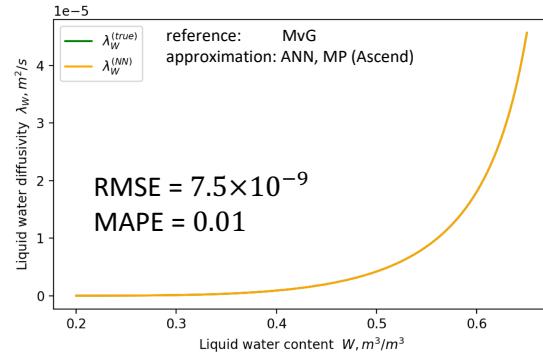
(a) The case of BBstab method, M-vG formulation



(b) The case of BBstab method, B-C formulation



(c) The case of BBstab method, Gard formulation



(d) The case of ANN, mixed precision (MP)

Figure 6. Diffusion coefficient for soil moisture λ_W as a function of volumetric moisture content W being approximated given forms of (a) M-vG, (b) Brooks-Corey (B-C) and (c) Gardner (Gard) formulae for moisture diffusivity λ_W and hydraulic conductivity γ coefficients in test solution of Richards equation (16), BBstab optimization procedure applied. In (d), we present diffusion coefficient for soil moisture being approximated given artificial neural networks for moisture diffusivity λ_W and hydraulic conductivity γ coefficients in test solution of Richards equation (16), MP computations employed on Ascend compute platform. Here, we produce the reference solution using the M-vG formulation

Table 1. Quality measures of the temperature conductance equation (15) identification. The ANN-based solutions are obtained using Ascend compute platform

method;	MAPE ($\lambda_T/\rho c$)	RMSE ($\lambda_T/\rho c$)	$\mathcal{Q}_{mapc,\epsilon_{h\tau}}$	$\mathcal{Q}_{rmse,\epsilon_{h\tau}}$
FP32/MP	-	$m^2 s^{-1}$	-	$K s^{-1}$
ANN, MP	8.1×10^{-4}	3.6×10^{-10}	7.04×10^{-3}	1.9×10^{-6}
ANN, FP32	6.4×10^{-4}	3.2×10^{-10}	4.1×10^{-4}	5.2×10^{-7}
BBstab(C-K), FP32	1.7×10^{-2}	6.1×10^{-9}	3.6×10^{-2}	1.6×10^{-6}
BBstab(Joh), FP32	4.2×10^{-2}	1.5×10^{-8}	8.7×10^{-2}	4.1×10^{-6}

3.3. Relative Performance of ML-Based Method vs. Classical PDE Identification Method

Tables 1–2 and 3–4 allow the direct comparison of BBstab and ANN methods in terms of error metrics and computational efficiency, respectively. Though being faster in wall-clock time, BBstab method significantly underperforms in accuracy. In MAPE and RMSE for temperature

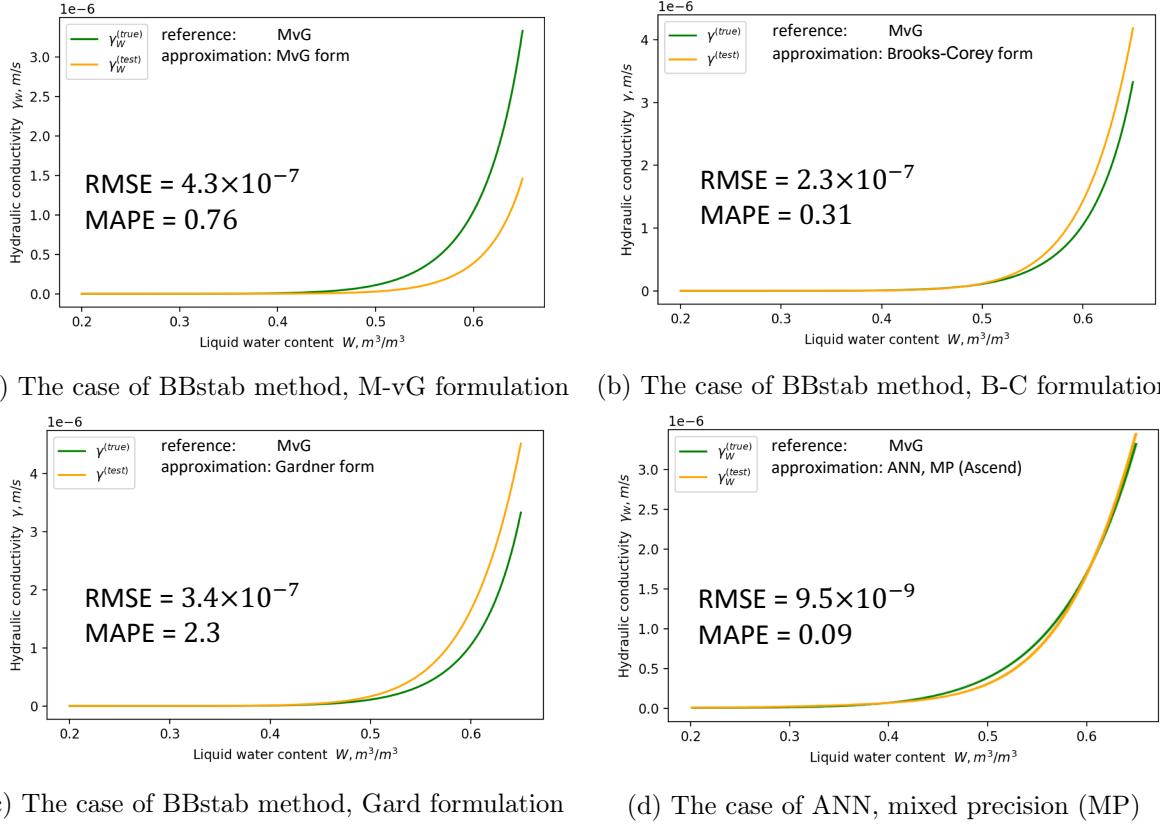


Figure 7. Hydraulic conductivity coefficient for soil moisture γ as a function of volumetric moisture content W being approximated given forms of (a) M-vG, (b) Brooks-Corey (B-C) and (c) Gardner (Gard) formulae for moisture diffusivity λ_W and hydraulic conductivity γ coefficients in test solution of Richards equation (16), BBstab optimization procedure applied. In (d), we present hydraulic conductivity coefficient for soil moisture being approximated given artificial neural networks for moisture diffusivity λ_W and hydraulic conductivity γ coefficients in test solution of Richards equation (16), MP computations employed on Ascend compute platform. Here, we produce the reference solution using the M-vG formulation

Table 2. Quality measures of the Richards equation (16) identification. The ANN-based solutions are obtained using Ascend compute platform

method;	MAPE (λ_W)	MAPE (γ)	RMSE (λ_W)	RMSE (γ)	$\mathcal{Q}_{mape,\epsilon_{h\tau}}$	$\mathcal{Q}_{rmse,\epsilon_{h\tau}}$
FP32/MP	-	-	$m^2 s^{-1}$	$m s^{-1}$	-	s^{-1}
ANN	1.01×10^{-2}	8.9×10^{-2}	7.5×10^{-9}	9.5×10^{-9}	5.8×10^{-1}	6.1×10^{-8}
MP						
BBstab(M-vG)	1.3×10^{-1}	7.6×10^{-1}	6.4×10^{-7}	4.3×10^{-7}	1.2×10^0	1.3×10^{-6}
FP32						
BBstab(B-C)	1.3×10^{-1}	3.1×10^{-1}	2.0×10^{-6}	2.3×10^{-7}	1.2×10^0	1.2×10^{-6}
FP32						
BBstab(Gard)	4.4×10^{-1}	2.3×10^0	4.4×10^{-6}	3.4×10^{-7}	8.1×10^{-1}	3.1×10^{-6}
FP32						

conductivity, liquid water diffusivity, and hydraulic conductivity, BBstab demonstrates error values 1–3 orders of magnitude higher than those obtained with ANN method.

Table 3. Performance estimates of the temperature conductance equation (15) identification. The ANN-based solutions were obtained using Ascend compute platform

method	FP32/MP	perf., it/sec	T_c , iter	T_c , w.time
ANN	MP	8.68	11800	2086 s.
BBstab(C-K)	FP32	9.1×10^{-2}	100	1096 s.
BBstab(Joh)	FP32	9.1×10^{-2}	100	1053 s.

Table 4. Performance estimates of the Richards equation (16) identification. The ANN-based solutions were obtained using Ascend compute platform

method	FP32/MP	perf., it/sec	T_c , iter	T_c , w.time
ANN	MP	9.1	15507	2500 s.
BBstab(M-vG)	FP32	1×10^{-1}	100	994 s.
BBstab(B-C)	FP32	1.2×10^{-1}	100	843 s.
BBstab(Gard)	FP32	1.1×10^{-1}	100	939 s.

3.4. Relation to Results of Other Groups

An idea to reconstruct the Richards equation from measured data on soil moisture has got development during recent years. In [5], the ANN with monotonicity constraints were used to approximate coefficients of the Richards equation of the same form as used in our study. The reference solutions were produced by HYDRUS-1D model with Mualem-van Genuchten formulation for PDE coefficients. The results of inverse problem solution are thoroughly analyzed from the point of soil physics, however, no comparison with the baseline PDE identification method (like BBstab in our case) is provided and the visual inspection of figures in the cited paper allows to conclude that in our study the ANN-based algorithm produces solution with much higher accuracy. In [18], the ANNs are involved to recover the linear Richards equation containing additional terms of higher-order derivatives. In contrast, our approach sticks to classic nonlinear form of Richards equation, having solid physical basis. In both mentioned papers, the input data for identification problem was synthetic, produced by direct problem solution with parameters and/or boundary conditions, instructed from real measurements; the same strategy applied in our work.

Conclusions

In this study, we propose a novel method based on ANNs for the identification of partial differential equations. We demonstrated its efficacy and high accuracy in a case of diffusion equation applied to the problem of heat conduction in soil, and nonlinear diffusion-advection equation (Richards equation) applied to the soil moisture simulation. We also propose physics-guided and other types of regularizations for the stabilization of neural networks training. We found that in case of the identification of advection term, one may face the issue of strong interaction between loss sums elements in MSE, MAE and MAPE, which may cause uninformative gradients

and, in turn, poor identification quality and high uncertainty in approximated PDE coefficient. To alleviate this issue, we elaborated random re-weighting of the individual components of these terms' sums, which resulted in informative gradients, stabilized training and good approximation of the PDE coefficient. The usage of novel AI Ascend platform allows to significantly speedup implementation of ANN-based algorithm.

Regarding the quality of the PDE identification, we draw the following conclusions:

- novel method based on ANNs for the identification of PDE coefficients describing heat and moisture transport in soil implemented on Ascend platform using mixed precision floating point operations overperforms the classical gradient descent method in Barzilai–Borwein stabilized (BBstab) modification, in terms of MAPE and/or RMSE at least an order of magnitude;
- BBstab method requires good initial guess of parameters being optimized to converge to true solution, whereas, for ANN method, the sensitivity of training results to initial guess is much less limiting the accuracy of solution.

The ANN-based method we developed for PDE identification may be used in research areas other than soil thermodynamics. The Richards equation is a diffusion-advection type equation, with highly nonlinear advective term. We expect applicability of suggested approach to hydrodynamic-type problems, e.g., developing turbulence closures, where the reference solutions of PDEs are usually obtained from high-resolution direct Navier-Stokes simulations. Moreover, our method of ANN-based identification is not limited to the cases where a PDE solver is implemented in a form of finite-difference approximation, it can be used in conjunction with any other differentiable numerical PDE solvers.

Acknowledgements

The work is partially supported by the Russian Ministry of Science and Higher Education, project No. 075-15-2019-1621 (the general statement of the inverse problem), by Research and Educational School of Moscow State University “Brain, cognitive systems, artificial intelligence” (development of the neural-network based algorithm), and by the Russian Science Foundation, grant no. 21-71-30003 (creation and application of new high-performance algorithms for the development of Earth system models, using novel computing architectures).

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Mixed precision training, <http://docs.nvidia.com/deeplearning/frameworks/mixed-precision-training/index.html>
2. Abadi, M., Agarwal, A., Barham, P., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), <http://tensorflow.org/>
3. Akiba, T., Sano, S., Yanase, T., et al.: Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference

- on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. pp. 2623–2631. ACM (2019). <https://doi.org/10.1145/3292500.3330701>
4. Baek, M., DiMaio, F., Anishchenko, I., *et al.*: Accurate prediction of protein structures and interactions using a three-track neural network. *Science* 373(6557), 871–876 (2021). <https://doi.org/10.1126/science.abj8754>
 5. Bandai, T., Ghezzehei, T.A.: Physics-Informed Neural Networks With Monotonicity Constraints for Richardson-Richards Equation: Estimation of Constitutive Relationships and Soil Water Flux Density From Volumetric Water Content Measurements. *Water Resources Research* 57(2) (feb 2021). <https://doi.org/10.1029/2020WR027642>
 6. Barzilai, J., Borwein, J.M.: Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis* 8(1), 141–148 (1988). <https://doi.org/10.1093/imanum/8.1.141>
 7. Brooks, R., Corey, A.: Hydraulic Properties of Porous Media. Tech. rep., Colorado State University, Fort Collins (1964)
 8. Burdakov, O., Dai, Y.H., Huang, N.: Stabilized Barzilai-Borwein method. *Journal of Computational Mathematics* 37(6), 916–936 (2019). <https://doi.org/10.4208/jcm.1911-m2019-0171>
 9. Burdine, N.: Relative Permeability Calculations From Pore Size Distribution Data. *Journal of Petroleum Technology* 5(03), 71–78 (mar 1953). <https://doi.org/10.2118/225-G>
 10. Camporeale, E., Wilkie, G.J., Drozdov, A., Bortnik, J.: Machine-learning based discovery of missing physical processes in radiation belt modeling (2021)
 11. Côté, J., Konrad, J.M.: A generalized thermal conductivity model for soils and construction materials. *Canadian Geotechnical Journal* 42(2), 443–458 (apr 2005). <https://doi.org/10.1139/t04-106>
 12. Du, C.: Comparison of the performance of 22 models describing soil water retention curves from saturation to oven dryness. *Vadose Zone Journal* 19(1) (jan 2020). <https://doi.org/10.1002/vzj2.20072>
 13. Fadeev, R.Y., Ushakov, K.V., Tolstykh, M.A., Ibrayev, R.A.: Design and development of the SLAV-INMIO-CICE coupled model for seasonal prediction and climate research. *Russian Journal of Numerical Analysis and Mathematical Modelling* 33(6), 333–340 (dec 2018). <https://doi.org/10.1515/rnam-2018-0028>
 14. Fuentes, C., Chávez, C., Brambila, F.: Relating Hydraulic Conductivity Curve to Soil-Water Retention Curve Using a Fractal Model. *Mathematics* 8(12), 2201 (dec 2020). <https://doi.org/10.3390/math8122201>
 15. Gardner, W.R.: Field Measurement of Soil Water Diffusivity. *Soil Science Society of America Journal* 34(5), 832 (1970). <https://doi.org/10.2136/sssaj1970.03615995003400050045x>
 16. Gasmi, C.F., Tchelepi, H.: Physics informed deep learning for flow and transport in porous media (2021)

17. van Genuchten, M.T.: A Closed-form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils. *Soil Science Society of America Journal* 44(5), 892–898 (sep 1980). <https://doi.org/10.2136/sssaj1980.03615995004400050002x>
18. Ghorbani, A., Sadeghi, M., Jones, S.B.: Towards new soil water flow equations using physics-constrained machine learning. *Vadose Zone Journal* 20(4) (jul 2021). <https://doi.org/10.1002/vzj2.20136>
19. Haghighat, E., Raissi, M., Moure, A., *et al.*: A deep learning framework for solution and discovery in solid mechanics: linear elasticity. *CoRR* abs/2003.02751 (2020), <https://arxiv.org/abs/2003.02751>
20. Harris, C.R., Millman, K.J., van der Walt, S.J., *et al.*: Array programming with NumPy. *Nature* 585(7825), 357–362 (Sep 2020). <https://doi.org/10.1038/s41586-020-2649-2>
21. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (December 2015), https://www.cv-foundation.org/openaccess/content_iccv_2015/html/He_Delving_Deep_into_ICCV_2015_paper.html
22. Jagtap, A., Karniadakis, G.: Extended Physics-Informed Neural Networks (XPINNs): A Generalized Space-Time Domain Decomposition Based Deep Learning Framework for Non-linear Partial Differential Equations. *Communications in Computational Physics* 28, 2002–2041 (11 2020). <https://doi.org/10.4208/cicp.OA-2020-0164>
23. Jia, W., Wang, H., Chen, M., *et al.*: Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC ’20, IEEE Press (2020). <https://doi.org/10.5555/3433701.3433707>
24. Johansen, O.: Thermal conductivity of soils. Ph.D. thesis, University of Trondheim (1975)
25. Jumper, J., Evans, R., Pritzel, A., *et al.*: Highly accurate protein structure prediction with alphafold. *Nature* (2021), <https://doi.org/10.1038/s41586-021-03819-2>
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd Int. Conf. on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), <http://arxiv.org/abs/1412.6980>
27. Kokoreva, A.A., Dembovetskiy, A.V., Ezhelev, Z.S., *et al.*: Simulating water transport in porous media of urban soil. *IOP Conference Series: Earth and Environmental Science* 862(1), 012042 (oct 2021). <https://doi.org/10.1088/1755-1315/862/1/012042>
28. Kurth, T., Treichler, S., Romero, J., *et al.*: Exascale deep learning for climate analytics. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis, SC 2018, Dallas, TX, USA, November 11-16, 2018. pp. 51:1–51:12. IEEE / ACM (2018). <https://doi.org/10.5555/3291656.3291724>
29. Li, Z., Kovachki, N.B., Azizzadenesheli, K., *et al.*: Fourier neural operator for parametric partial differential equations. *CoRR* abs/2010.08895 (2020), <https://arxiv.org/abs/2010.08895>

30. Li, Z., Kovachki, N.B., Azizzadenesheli, K., *et al.*: Fourier neural operator for parametric partial differential equations. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021), <https://openreview.net/forum?id=c8P9NQVtmn0>
 31. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), <https://openreview.net/forum?id=Skq89Scxx>
 32. Lu, L., Jin, P., Karniadakis, G.E.: DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. CoRR abs/1910.03193 (2019), <http://arxiv.org/abs/1910.03193>
 33. Mishkin, D., Matas, J.: All you need is a good init. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), <http://arxiv.org/abs/1511.06422>
 34. Misra, D.: Mish: A self regularized non-monotonic activation function (2020), <http://arxiv.org/abs/1908.08681>
 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 (apr 2019). <https://doi.org/10.1515/rnam-2019-0010>
 36. Mualem, Y.: A new model for predicting the hydraulic conductivity of unsaturated porous media. Water Resources Research 12(3), 513–522 (jun 1976). <https://doi.org/10.1029/WR012i003p00513>
 37. Pfau, D., Spencer, J.S., Matthews, A.G.D.G., Foulkes, W.M.C.: Ab initio solution of the many-electron Schrödinger equation with deep neural networks. Physical Review Research 2(3) (Sep 2020). <https://doi.org/10.1103/physrevresearch.2.033429>
 38. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378, 686–707 (2019). <https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045>
 39. Sitzmann, V., Martel, J.N.P., Bergman, A.W., *et al.*: Implicit neural representations with periodic activation functions. CoRR abs/2006.09661 (2020), <https://arxiv.org/abs/2006.09661>
 40. Tancik, M., Srinivasan, P.P., Mildenhall, B., *et al.*: Fourier features let networks learn high frequency functions in low dimensional domains. CoRR abs/2006.10739 (2020), <https://arxiv.org/abs/2006.10739>
 41. Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009)
-

42. Volodin, E.M., Gritsun, A.S.: Simulation of Possible Future Climate Changes in the 21st Century in the INM-CM5 Climate Model. Izvestiya, Atmospheric and Oceanic Physics 56(3), 218–228 (may 2020). <https://doi.org/10.1134/S0001433820030123>
43. Weyn, J.A., Durran, D.R., Caruana, R.: Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. Journal of Advances in Modeling Earth Systems 12(9) (Sep 2020). <https://doi.org/10.1029/2020ms002109>
44. Wiecha, P.R., Arbouet, A., Girard, C., Muskens, O.L.: Deep learning in nano-photonics: inverse design and beyond. Photonics Research 9(5), B182 (Apr 2021). <https://doi.org/10.1364/prj.415960>
45. Ziogas, A.N., Ben-Nun, T., Fernández, G.I., *et al.*: A data-centric approach to extreme-scale ab initio dissipative quantum transport simulations. Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (Nov 2019). <https://doi.org/10.1145/3295500.3357156>

Appendix A. On the Interaction of the Loss Sum Elements in Approximated Advection Term

In Section 2.4.4, we stated that in Richards equation case, there is an issue of loss sum elements strongly interacting with each other resulting in uninformative gradients. In particular, this feature takes place when one considers the advective term $\frac{\partial \gamma}{\partial z}$ of the r.h.s. of Richards equation within our approximation approach. In this section, we present the derivation of the gradients of loss sums in eq. (60) w.r.t. γ in order to demonstrate either their reduction to zero or to come constant independent of ground truth, in case of smooth function $\gamma(W)$ and fine z -grid.

Let us consider the loss function of a tendency $\frac{\partial W}{\partial t}$ concerning the term $\frac{\partial \gamma}{\partial z}$:

$$\mathcal{L} = \text{Err} \left(\frac{\partial W}{\partial t} \Big|_{\gamma} \right) = \text{Err} \left(\frac{\partial \gamma}{\partial z} \right), \quad (64)$$

where Err is either MSE, MAE or MAPE.

A.1. MSE Term

First, we consider MSE $\left(\frac{\partial \gamma}{\partial z} \right)$ (see the details for MAE and MAPE further).

$$\mathcal{L} = \sum_{i=1}^N \left(\frac{\partial \gamma}{\partial z} \Big|_{NN,i} - \frac{\partial \gamma}{\partial z} \Big|_{true,i} \right)^2, \quad (65)$$

where i enumerates z -elements of a soil profile.

In our scheme, the r.h.s. of Richards equation is approximated using finite differences, thus, one may express the loss sum the following way:

$$\mathcal{L} = \sum_{i=1}^N \left(\frac{\Delta \gamma}{\Delta z} \Big|_{NN,i} - \frac{\Delta \gamma}{\Delta z} \Big|_{true,i} \right)^2, \quad (66)$$

In our method, z -grid is uniform, thus, all the Δz_i are equal: $\Delta z_i = \delta_z, \forall i$. Let us denote \mathcal{L}' the finite-difference approximated loss term multiplied by δ_z^2 : $\mathcal{L}' = \delta_z^2 \mathcal{L}$. Then this term is expressed the following way:

$$\mathcal{L}' = \sum_{i=1}^N \left(\Delta \gamma|_{NN,i} - \Delta \gamma|_{true,i} \right)^2, \quad (67)$$

Let us denote $\nu = \gamma_{NN}$, and $\tau = \gamma_{true}$ for the convenience and brevity. Consider the expansion of the sum in eq. (67):

$$\begin{aligned} \mathcal{L}' &= \sum_{i=2}^N (\nu_i - \nu_{i-1} - \tau_i + \tau_{i-1})^2 = \\ &= \dots + (\nu_j - \nu_{j-1} - \tau_j + \tau_{j-1})^2 + (\nu_{j+1} - \nu_j - \tau_{j+1} + \tau_j)^2 + \dots = \\ &= \dots + \nu_j^2 + \nu_{j-1}^2 + \tau_j^2 + \tau_{j-1}^2 - \\ &\quad - 2\nu_j \nu_{j-1} - 2\nu_j \tau_j + 2\nu_j \tau_{j-1} + 2\nu_{j-1} \tau_j - 2\nu_{j-1} \tau_{j-1} - 2\tau_j \tau_{j-1} + \\ &\quad + \nu_{j+1}^2 + \nu_j^2 + \tau_{j+1}^2 + \tau_j^2 - \\ &\quad - 2\nu_{j+1} \nu_j - 2\nu_{j+1} \tau_{j+1} + 2\nu_{j+1} \tau_j + 2\nu_j \tau_{j+1} - 2\nu_j \tau_j - 2\tau_{j+1} \tau_j + \dots \end{aligned} \quad (68)$$

Here, j is an index of some arbitrary sum element in the expansion. When one would like to compute the gradients of the loss term \mathcal{L} w.r.t. parameters θ_γ of the neural network F_γ approximating γ coefficient, the chain rule is applied in the following form:

$$\frac{\partial \mathcal{L}}{\partial \theta_\gamma} = \frac{1}{\delta_z^2} \frac{\partial \mathcal{L}'}{\partial \theta_\gamma} = \frac{1}{\delta_z^2} \frac{\partial \mathcal{L}'}{\partial \gamma} \frac{\partial \gamma}{\partial \theta_\gamma} = \frac{1}{\delta_z^2} \sum_{i=1}^N \frac{\partial \mathcal{L}'}{\partial \nu_i} \frac{\partial \nu_i}{\partial \theta_\gamma}.$$

Since the most of the expanded sum terms in eq. (68) are not dependent of ν_j , the finite-difference approximated gradient of the loss \mathcal{L}' w.r.t. γ for an arbitrary j -th element ν_j is expressed the following way:

$$\begin{aligned} \frac{\partial \mathcal{L}'}{\partial \gamma} \Big|_j &= \frac{\partial \mathcal{L}'}{\partial \nu_j} = \\ &= \frac{\partial}{\partial \nu_j} (2\nu_j^2 - 2\nu_j \nu_{j-1} - 2\nu_j \tau_j + 2\nu_j \tau_{j-1} - 2\nu_j \nu_{j+1} + 2\nu_j \tau_{j+1} - 2\nu_j \tau_j) = \\ &= 2\nu_j^2 + 2\nu_j (-\nu_{j-1} - \tau_j + \tau_{j-1} - \nu_{j+1} + \tau_{j+1} - \tau_j). \end{aligned} \quad (69)$$

Since the function γ is supposed to be smooth, and also the z grid has 1000 levels with fine resolution, one may observe the following approximate equalities:

$$\begin{aligned} \nu_{j+1} + \nu_{j-1} &\approx 2\nu_j, \\ \tau_{j-1} + \tau_{j+1} &\approx 2\tau_j. \end{aligned} \quad (70)$$

Thus, the gradient in eq. (69) is transformed the following way:

$$\frac{\partial \mathcal{L}'}{\partial \nu_j} = \frac{\partial}{\partial \nu_j} (2\nu_j^2 + 2\nu_j (-2\nu_j + 2\tau_i - 2\tau_j)) \approx -4\nu_j. \quad (71)$$

Thus, in case of smooth γ function and fine z -grid, one may clearly see that the gradients of MSE loss of the advective term do not depend on external data γ_{true} , and, thus, are uninformative.

A.2. MAE Term

In this section, we consider MAE $\left(\frac{\partial\gamma}{\partial z}\right)$.

$$\mathcal{L} = \sum_{i=1}^N \left| \frac{\partial\gamma}{\partial z} \Big|_{NN,i} - \frac{\partial\gamma}{\partial z} \Big|_{true,i} \right|, \quad (72)$$

where i enumerates z -elements of a soil profile. Similar to the derivation of MSE term, we use ν and τ notation for γ_{NN} and γ_{true} correspondingly. We also use the following notation here: $\mathcal{L}' = \delta_z \mathcal{L}$. Thus, the gradient of the MAE loss term is expressed the following way:

$$\frac{\partial \mathcal{L}}{\partial \theta_\gamma} = \frac{1}{\delta_z} \frac{\partial \mathcal{L}'}{\partial \theta_\gamma} = \frac{1}{\delta_z} \frac{\partial \mathcal{L}'}{\partial \gamma} \frac{\partial \gamma}{\partial \theta_\gamma} = \frac{1}{\delta_z} \sum_{i=1}^N \frac{\partial \mathcal{L}'}{\partial \nu_i} \frac{\partial \nu_i}{\partial \theta_\gamma}.$$

Let us consider the gradient of the expansion of the \mathcal{L}' term w.r.t. some arbitrary sum elements ν_j in case of finite-difference estimated derivatives $\frac{\partial\gamma}{\partial z}$:

$$\begin{aligned} \frac{\partial \mathcal{L}'}{\partial \nu_j} &= \frac{\partial}{\partial \nu_j} \sum_{i=2}^N |\nu_i - \nu_{i-1} - \tau_i + \tau_{i-1}| = \\ &= \frac{\partial}{\partial \nu_j} (\dots + s_i (\nu_j - \nu_{j-1} - \tau_j + \tau_{j-1}) + s_{i+1} (\nu_{j+1} - \nu_j - \tau_{j+1} + \tau_j) + \dots), \end{aligned}$$

where $s_j = 1$ in case $(\nu_j - \nu_{j-1} - \tau_j + \tau_{j-1}) \geq 0$, and $s_j = -1$ otherwise. Since ν and τ are smooth, and z -levels are small, one may consider the case $s_j = s_{j+1}$ most frequent compared to the case $s_j = -s_{j+1}$. In case $s_j = s_{j+1}$:

$$\frac{\partial \mathcal{L}'}{\partial \nu_j} \Big|_{s_j=s_{j+1}} = s_j \frac{\partial}{\partial \nu_j} (\nu_j - \nu_{j-1} - \tau_j + \tau_{j-1} + \nu_{j+1} - \nu_j - \tau_{j+1} + \tau_j) = 0,$$

which is an uninformative gradient.

In case $s_j = -s_{j+1}$:

$$\frac{\partial \mathcal{L}'}{\partial \nu_j} \Big|_{s_j=-s_{j+1}} = s_j \frac{\partial}{\partial \nu_j} (\nu_j - \nu_{j-1} - \tau_j + \tau_{j-1} - \nu_{j+1} + \nu_j + \tau_{j+1} - \tau_j) = 2s_j,$$

which is an uninformative gradient since it is not dependent of “true” values that are τ in this notation. Thus, in this section, we demonstrated that in case of MAE loss term, its gradients w.r.t. $\gamma_{NN,j}$ are uninformative constants - either zero or $\pm 2\delta_z$. This is in agreement with Fig. 2, where one may observe that the gradients with added 10^{-14} stability value are either 10^{-14} or some constant.

A.3. MAPE Term

In this section, we consider MAPE $\left(\frac{\partial\gamma}{\partial z}\right)$.

$$\mathcal{L} = \sum_{i=1}^N \left| \frac{\frac{\partial\gamma}{\partial z} \Big|_{NN,i} - \frac{\partial\gamma}{\partial z} \Big|_{true,i}}{\frac{\partial\gamma}{\partial z} \Big|_{true,i}} \right|, \quad (73)$$

One may notice that τ is smooth, thus, in case of finite-difference approximation of the derivatives in this loss term, $\Delta\tau_j \approx \Delta\tau_{j+1}$. The derivation of the gradients in this section is similar to the previous section A.2 with the only reservation of denominator $\Delta\tau_j$, thus, the gradients may be expressed the following way:

$$\begin{aligned}\frac{\partial \mathcal{L}'}{\partial \nu_j} \Big|_{s_j=s_{j+1}} &= 0, \\ \frac{\partial \mathcal{L}'}{\partial \nu_j} \Big|_{s_j=-s_{j+1}} &= \frac{2s_j}{\Delta\tau_j}.\end{aligned}\tag{74}$$

Here one may note once again that the case $s_i = s_{j+1}$ is much more frequent compared to the case $s_i = -s_{j+1}$ due to smoothness of γ coefficient and $F_{NN,\gamma}$ neural network. Thus, major gradient values are zeros which is uninformative. This is in agreement with Fig. 8, where one may observe that the gradients with added 10^{-14} stability value are mostly 10^{-14} with some rare exceptions. In Fig. 8b, one may also observe that random re-weighting of individual MAPE sum elements according to eq. (61) made the gradients informative.

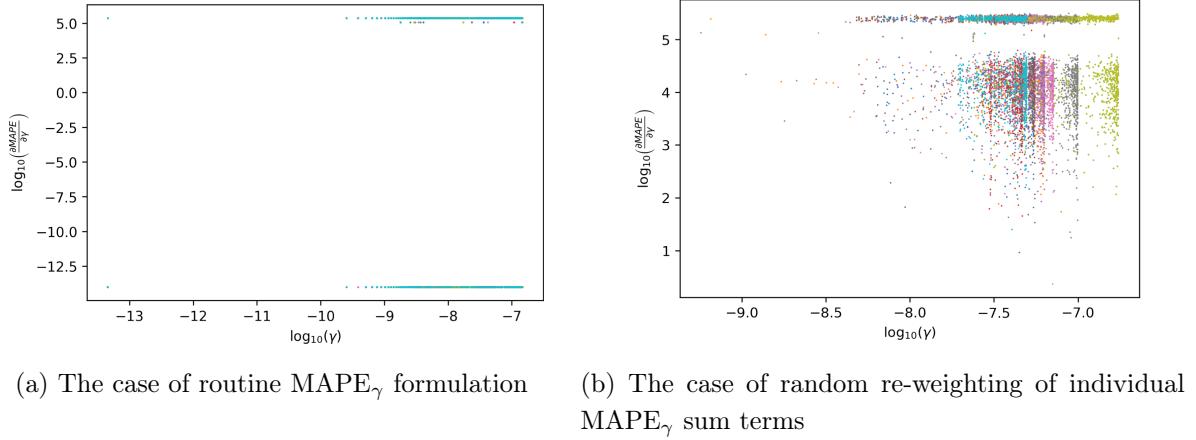


Figure 8. $MAPE_\gamma$ loss gradients w.r.t. γ as individual sum terms vs. γ : (a) in case of routine $MAPE_\gamma$ formulation; (b) in case of random re-weighting of individual $MAPE_\gamma$ sum terms according to eq. (61). Here we present it in logarithmic scale with additive log-stability term 10^{-14}

Scalability and Performance of a Program that Uses Domain Decomposition for Monte Carlo Simulation of Molecular Liquids

Alexander V. Teplyukhin¹ 

© The Author 2022. This paper is published with open access at SuperFri.org

The main factors hindering the development of supercomputer programs for molecular simulation by the Monte Carlo method within the framework of classical physics are considered, and possible ways to eliminate the problems that arise in this case are discussed. Thus, the use of molecular models with moderate stiffness of covalent bonds between fragments makes it possible not only to increase the efficiency of scanning the configuration space of the model, but also to abandon the complex apparatus of kinematics with rigid links, which significantly limits the possibilities of domain decomposition. Based on the domain decomposition strategy and a simplified treatment of the deformation energy of covalent bonds and angles, an original parallel algorithm for calculating the properties of large all-atomic models of aqueous solutions of biopolymers by the Monte Carlo method was developed. To speed up computations within the framework of this approach, each domain is assigned its own group of processors/cores using local data replication and splitting the loop over the interacting partners. The article discusses the logical scheme of the computational algorithm and the main components of the software package (fortran77, MPI 1.2). Test calculations performed for water and *n*-hexane demonstrated the high performance and scalability of the program in which the proposed algorithm was implemented.

Keywords: parallel calculation, biopolymers, Monte Carlo, MPI.

Introduction

Stochastic Monte Carlo algorithms [19] are successfully used to solve a wide range of problems in many areas of science and engineering [9, 13, 16]. So, when studying the physicochemical and structural properties of molecular aggregates containing hundreds or more atoms, calculations are practiced within the framework of classical physics based on the mathematical apparatus of Markov chains [2, 6, 18, 34]. Due to the statistical nature of this type of computational experiments, it takes weeks, and sometimes months, to process huge amounts of data in order to achieve acceptable accuracy of the results. Worse, the size of the modeled object is strictly limited from above by the amount of RAM in a typical computer. It is clear that the study of models represented by millions of atoms (for example, biopolymers in an aqueous solution) should be carried out on a supercomputer using a program code that implements distributed computing by hundreds of processor cores.

At first glance, it may seem that, unlike the molecular dynamics method [2], algorithms that generate a Markov chain in the configuration space of the model under study (the Metropolis et al. procedure [2, 18]) have an extremely low potential for parallelization. Indeed, in the first case, all particles of the model simultaneously change their position at each iteration, while in the second, usually only one, and even then with a probability of no more than 50%². Respectively, in the first case, parallelization can be organized by splitting the loops both over particles and over their interaction partners [22], and in the second case, only over interaction partners. Nevertheless, the success of using the Monte Carlo method in statistical physics [16] and polymer science [17] gives a serious stimulus to find solutions to this problem.

¹Institute of Mathematical Problems of Biology of RAS - the Branch of Keldysh Institute of Applied Mathematics of RAS, Pushchino, Russian Federation

²Cluster and rejection free algorithms do not greatly improve the situation.

A necessary condition for the parallelizability of an algorithm is the presence of data-independent operations in it. It turned out that there is a very important class of models for which the steps of the procedure of Metropolis et al. [2, 18] will have this property if a restriction is introduced on the radius of interaction of their components. In this case, very often the connection of the previous and subsequent states of the generated Markov chain will have a formal character, since the values of the parameters characterizing one of these states do not affect the probability of transition to the second and vice versa. Using information about the interactions in the model, one can significantly reduce the time of calculations by organizing the simultaneous execution of calculations for independent groups of components. For example, in the 2D Ising model, the spins do not interact along the diagonals of the lattice. Consequently, the entire system of spins can be divided into two subgroups like a chessboard [21] and the states of all spins can be updated either in one or the other group in turn. Such a scheme is highly scalable and allows efficient use of computing resources [23].

The transition to continual (off-lattice) models of condensed phases requires the use of more complex grouping methods. Thus, in [10], a scheme was proposed according to which the space of the system being modeled (a rectangular cell with periodic boundary conditions) is divided into cubic domains, and each domain into eight more subdomains (cube octants). The length of the subdomain edge must be greater than the radius of the sphere that limits the interactions of particles in the model. This scheme ensures the independence of the movement of particles in similarly oriented domain octants. In the simplest case, the number of processors³ used for calculations is equal to the number of domains, so that the processors perform the procedure of Metropolis et al. [2, 18] with particles of their domains selected from ‘active’ octants, looping through in the same sequence for all domains. Note that each processor ‘sees’ only particles in its own domain and in the boundary regions of 26 neighbors.

The domain decomposition strategy allows for a variety of practical implementations. In [10], the authors focused on optimizing the use of memory by proposing an algorithm with a very complex protocol of interprocessor communication, which adversely affected the speed of calculations. After Y2K, the problem of lack of memory lost its acuteness, which gave us the opportunity to develop a more advanced algorithm, significantly reducing the intensity of interprocessor data exchanges [14, 25].

Theoretically, spatial (domain) decomposition is capable of providing high scalability of computational algorithms by dividing a ‘big problem’ into several simultaneously soluble small subtasks. However, within the framework of this method, it is possible to reduce the size of domains only to a certain value (the cutoff radius of particle interactions). On the contrary, splitting the loop over particles interaction partners is very useful when studying objects of small and medium size, but it does not scale well and quickly exhausts the resources of the computer’s RAM as the model size grows. The combination of these approaches makes it possible to ‘neutralize’ their limitations and develop efficient supercomputer programs capable of outperforming popular packages using the molecular dynamics method when studying models consisting of hundreds of millions of atoms. As a result of further modifications, we developed a new algorithm for calculating the properties of simulated systems in an isobaric-isothermal (NpT) ensemble [2], in which each domain is serviced by its own group of processors [26]. The program built on the basis of this algorithm was successfully used in the study of the structural and thermodynamic characteristics of water in a wide range of pressure [29, 30].

³Hereinafter, the word ‘processor’ will mean a separate processor core.

Another obstacle that generates skepticism about the Monte Carlo method and limits its application in the applied fields of materials science and biomedicine is the problem of taking into account intramolecular degrees of freedom. As it was shown in 1980 [20], the procedure of Metropolis et al. [2, 18], applied to individual atoms of a peptide molecule, gives 11 times less diffusion of atoms than the molecular dynamics method for the same amount of computer time. The reason for this inefficiency is that random monatomic displacements in a covalently bound system lead to large changes in the strain energy of bonds and angles, which significantly exceed the contributions of van der Waals and electrostatic interactions.

The standard way out of this situation is based on the idea of excluding from consideration the stiff degrees of freedom of molecules [8] by ‘freezing’ covalent bonds and the angles between them (partially or completely), as well as through the use of united atoms [12] and coarse-grained models of polymers. [24]. Unfortunately, the kinematics of displacements of atoms (groups of atoms) in such models is implemented through laborious procedures that use difficult mathematical calculations, sometimes requiring the solution of systems of nonlinear algebraic equations. Worse, some transformations [33] cannot be performed in parallel computations if the moved atoms are in different domains.

To increase the effectiveness of Monte Carlo trial moves, we [27] proposed to completely eliminate the contribution of the strain potentials of covalent bonds and angles to the energy increment, which is taken into account in the procedure of Metropolis et al. [2, 18]. Within this approach, a molecule must be considered either as one or several rigid fragments (compact, as a rule, π -conjugated groups of atoms) connected by single covalent bonds with zero stiffness and finite extensibility. Hydrogen and halogen atoms should be included in the fragment to which they are bonded. Some fragments may consist of a single atom, such as an ether oxygen atom or a quaternary carbon atom. From a mechanical point of view, fragments are solid bodies that perform rotational and translational motions as a whole, which position in space is determined by the Cartesian coordinates of their center and three Euler angles. The integrity of the chemical structure of the molecule is maintained by simply checking the bond lengths and angles for compliance with the standard ranges of values [1] for specific types of neighboring atoms. In the course of subsequent computational experiments, it turned out that such drastic measures with respect to strain potentials are not necessary at all [31]. In fact, the moderate stiffness of interfragment single bonds and angles not only improves the shape of the distribution of bond lengths and angles, but also makes it possible to increase the maximum amplitude of trial displacement and fragment rotation. It should be noted that this approach showed very good results in calculating the structural and thermodynamic characteristics of liquid hydrocarbons [31, 32].

The purpose of this work is to study the performance and scalability of the program we developed (fortran77, MPI 1.2), which uses domain decomposition together with the splitting of loop over fragments interaction partners for calculations based on a model with a simplified treatment of the strain energy of covalent bonds and angles [27, 31, 32], mentioned in the previous paragraph.

The article is organized as follows. In Section 1 the data on the molecular models used in the calculations will be presented. Section 2 is devoted to a description of a new algorithm for parallel computing by the Monte Carlo method, as well as the main components of a prototype⁴ software package for studying aqueous solutions of biopolymers. The Section 3 discusses the

⁴In addition to the program code, it is necessary to prepare a set of force field parameters that is adequate to the model we use and allows us to accurately reproduce the properties of substances in the condensed phase.

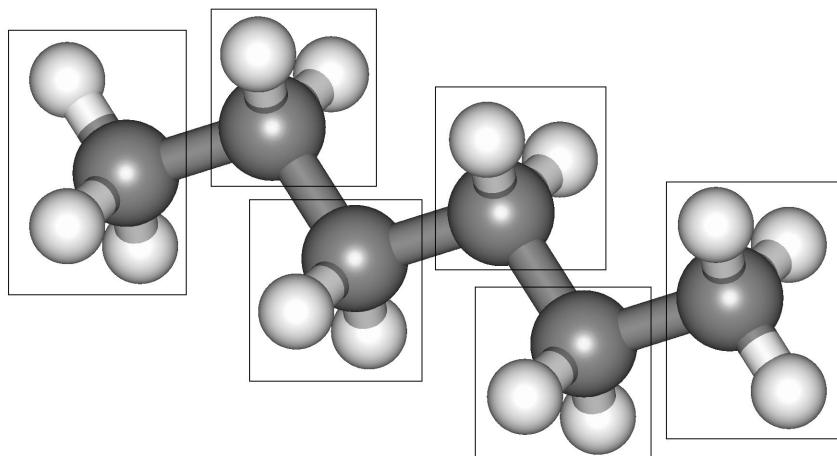


Figure 1. *n*-hexane molecule. Hydrogen atoms are light gray; carbon atoms are dark gray. Rigid fragments of the molecule are outlined by rectangles

results of computational experiments with all-atomic models of liquids of two kinds: *n*-hexane and water. Conclusion summarizes the study and points directions for further work.

1. Systems under Consideration

The model of the *n*-hexane molecule is a system of configurationally rigid fragments ($\text{CH}_3-(\text{CH}_2)_4-\text{CH}_3$: four CH_2 groups and two CH_3) connected by single bonds in accordance with the chemical structure of this molecule (Fig. 1). The CH bond length is 1.085 \AA , and the HCH bond angles are 109.47° . The potential energy characterizing the conformation of the *n*-hexane molecule is acquired due to non-covalent interactions of atoms of its fragments (short-range approximations of the Lennard-Jones and Coulomb potentials, see below), the deformation (strain) energy of interfragment covalent bonds and bond angles adjacent to them, as well as torsion potentials, hindering the rotation of molecular fragments around these bonds. The parameters of the Lennard-Jones potential functions are the same as in [31], the partial charges on atoms and the parameters of torsion potentials are taken from [32]. The strain energy is proportional to the squared deviation of bond lengths and angles from their equilibrium values with coefficients of $100 \text{ kcal/mol}/\text{\AA}^2$ and $25 \text{ kcal/mol}/\text{rad}^2$, respectively. For the equilibrium lengths of C–C chemical bonds, the value of 1.526 \AA was taken, and for the angles, 109.47° . This choice was largely motivated by the parameters of the force fields presented in articles [5, 11]. Atoms of neighboring fragments, connected by a covalent bond with each other or through a third atom, do not participate in the interactions described by the Lennard-Jones potential.

After a trial move (displacement and rotation) of the selected fragment, new values of the lengths of covalent bonds and angles are calculated, in the formation of which this fragment participates. If any bond or angle is not within the allowed range, the remaining bonds and angles are not checked, the trial move is rejected, the state before the move is again taken as the new state of the Markov chain, and the program continues to the next fragment. If all stereochemical parameters of the proposed configuration remained within the acceptable limits ($\pm 0.12 \text{ \AA}$ for bond lengths and $\pm 12^\circ$ for angles), the energy change and then the transition probability are calculated according to the formula of Metropolis et al. [2, 18]. The maximum displacement and rotation angle (for each coordinate and angle) is determined in a preliminary computer experiment so that $\sim 38\text{--}43\%$ of the trial configurations are accepted.

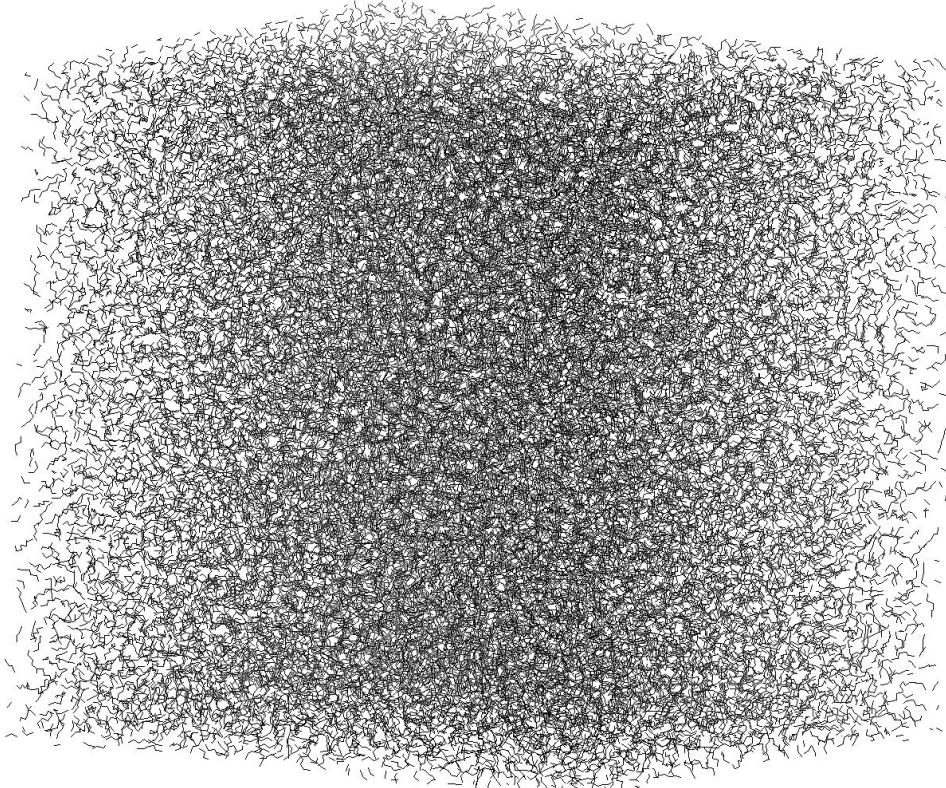


Figure 2. View of the main cell of the simulated system containing 19200 *n*-hexane molecules (384000 atoms). Hydrogen atoms not shown

To study water systems, we used the rigid three-center model of the water molecule SPC/E [4], which is an example of a molecule consisting of a single fragment.

Three simulated systems contained 2400, 19200 and 64800 *n*-hexane molecules each (48000, 384000 and 1296000 atoms, respectively), and three more – 33666, 269328 and 908982 water molecules (100998, 807984 and 2726946 atoms, respectively) in the main cubic cell with periodic boundary conditions (Fig. 2). The energy of intermolecular interactions was calculated using short-range pairwise additive atom-atom potential functions ($R_{cutoff} = 14 \text{ \AA}$, standard correction for the Lennard-Jones potential [2] and modification PF3 [28] for the Coulomb potential), which effectively take into account the contributions of all images of the main cell under periodic boundary conditions.

The initial configuration for each system is prepared using a special program that uses as many processors as there are domains in the simulated object. This program randomly places given amount of molecules within the boundaries of the main cell, taking into account periodic boundary conditions. The trial position and orientation of the molecule inserted is randomly set by the main processor (if the molecule is very large, the process is performed in batches). It also determines which domain a particular fragment falls into and sends the coordinates to the addressee. Upon completion of the distribution of fragments of the molecule intended for insertion, all processes determine, using a geometric criterion, whether there is an overlap of atoms with previously received fragments. The results of the check (0 or 1) are collected for all processes (MPI_ALLREDUCE, MPI_SUM). If 0 is received the attempt is accepted, otherwise it is retried. If the number of attempts exceeds the specified number, the program stops. To prevent this from happening on subsequent launches of the program, it is necessary to increase

the number of attempts or the volume of the main cell. If the program manages to fit all the molecules, each process writes all the information about its domain to a separate file on an external storage.

Obviously, the initial density of the simulated object will be much less than the real values corresponding to the given temperature and pressure. The relaxation of the studied models to the equilibrium state was carried out using the same program that is intended for calculating the structural and thermodynamic characteristics. Depending on the type of molecules, this requires 5–20 million trial moves per fragment. The main indicator in this process is the absence of a systematic trend in the batch-mean values of the density and heat of evaporation of the simulated liquids. To accelerate relaxation, the initial stages were performed at a pressure increased to 1 kbar.

The results were obtained using the equipment of Shared Resource Center of KIAM RAS (<http://ckp.kiam.ru>).

2. Computational Algorithm and Main Components of the Program

The simulated system occupies the main cell of a cubic shape, on which periodic boundary conditions are imposed. The volume of the main cell is divided into $N=n^3$ domains (n pieces along each coordinate axis). The computational task is submitted on $M=m \cdot n^3$ processors (processor cores), i.e. each domain is serviced by m processors (hereinafter referred to as the domain group of processors). Data about the simulated system are read from n^3 files on external media (each domain has its own data file) and written back to them after the completion of the next portion of the calculation. At startup, the head processor of each domain group reads data from the corresponding file and sends (MPI_BCAST) a copy of it to the rest of the processors in its group (replication). Processes of the same rank in a domain group are combined into corresponding communication groups with a 3D Cartesian topology (MPI_CART_CREATE). The algorithm described in [35] was used to generate pseudo-random numbers.

The actual computational procedure (see the logical scheme of the Algorithm 1) consists of two blocks of program code in the cyclic execution. The first block is responsible for performing an attempt, standard for the NpT ensemble [2], to randomly change the volume occupied by the model. To do this, one of the processors (rank 0 in MPI_COMM_WORLD) generates a new volume logarithm value and sends it (MPI_BCAST) to all processors. Then, domain group processors calculate trial values of atomic coordinates and exchange data with their topological neighbors (MPI_CART_SHIFT, MPI_SENDRECV). As a result, each processor receives information about the coordinates of atoms in the border zone from the side of the neighbor. Next, the processors calculate their portions in the change in the total energy of the system, and also, using simple geometric criteria, identify the partners in non-covalent and covalent interactions. All processors of each domain group, in parallel, carry out these operations by splitting of the corresponding loops over atoms. Upon completion of this, the processors of each domain group exchange data on the found pairs of neighbors (MPI_ALLGATHER), and all calculated energy increments are summed (MPI_REDUCE) with the result transferred to one of the processors (rank 0 in MPI_COMM_WORLD). It is this processor that decides (Metropolis et al. criterion [2, 18]) whether to accept the new volume value or keep the old one, and then sends the result to all M processors.

```
Initialization, input previously saved data;
while  $NStep \neq 0$  do
    //Begin block 1;
    Try to accept a new volume value;
    Generate random vector to shift domain-splitting grid;
    Generate random permutation for 8 subdomains selection order;
    //End of block 1;
    if Processor rank is 0 then
        Accumulate the data required to obtain averaged values of the density and the
            enthalpy of the simulated liquid;
    end
    //Begin block 2;
     $I \leftarrow 2$ ;
    while  $I \neq 0$  do
        if  $I = 2$  then
            | Set direct order of 8 subdomains selection;
        else
            | Set the selection in reverse order;
        end
        Try to change the position of a randomly selected fragments in 'active'
            subdomains;
         $I \leftarrow I - 1$ ;
    end
    //End of block 2;
     $NStep \leftarrow NStep - 1$ ;
end
Output data to next run;
```

Algorithm 1. The logical scheme of the computational algorithm

The second block is responsible for performing an attempt to randomly change the position (shift and rotation) of a randomly selected rigid fragment. This operation is performed by the head processors of each domain group in parallel in their domain. To ensure the statistical independence of n^3 simultaneous trial moves, each domain is divided into 8 ($2 \times 2 \times 2$) subdomains and moved molecules are taken from the subdomains in the same octant for all domains. This construction ensures that the selected fragments do not turn out to be partners in interactions, since all potential functions have a limited range of action, not exceeding half the length of the edge of the domain cell. Next, each head processor calculates the increment in the strain energy of the covalent bonds, bond and dihedral angles, and sends trial coordinates to the domain group. The processors of each domain group calculate the increment in the energy of non-covalent interactions from their portion of the neighbors of the tested fragment and 'drop' the results to the head processor of the group. After that, the head processor, having added the previously calculated 'covalent' contribution, decides (Metropolis et al. criterion [2, 18]) whether to accept the new position of the fragment or keep the old one, and then sends the result to all m processors of its group and proceeds to select the next fragment in the same subdomain. In practice, the loop over subdomain fragments should be long enough that on average each

fragment is selected 25 times. In this case, the radius of the criterion for inclusion in the list of interaction partners should be increased ($R_{neib}=R_{cutoff}+R_{fr}$, where R_{fr} is the radius of the minimum sphere containing all atoms of the largest fragment) by 1.5–2.5 Å (buffer zone). Before moving on to the next subdomains, the processors send to the topological neighbors the coordinates of the fragments from the subdomains that have just completed their ‘activity’. After fragments in all subdomains have been ‘processed’ in this way, the second block is completed, control is transferred to the beginning of the first block, and so on. To calculate the desired structural and thermodynamic characteristics, it is convenient to use the instantaneous configuration of molecular fragments formed before the start of the first block. Having performed the procedure described above for a specified number of times ($NStep$), the head processors write the information necessary for restart or further analysis to n^3 files on an external storage and the program stops.

An attentive reader may notice that in the above description of the logical scheme of the computational algorithm, some of the directly or indirectly mentioned episodes were left without discussion. Some of them have been omitted in order to reduce the size of the article. For example, these are procedures for transferring data between processors in neighboring domains, which can be implemented in different ways, depending on the preferences of the developer. The other, on the contrary, require separate consideration due to their importance. These include the problem of maintaining a detailed balance of transitions in the Markov chain while simultaneously performing the Metropolis et al. procedure [2, 18] in each domain. Recommendations on how to avoid violations of the detailed balance, as well as to improve the ergodicity of scanning the configuration space of the simulated systems, are given in the articles [3, 33].

Following these recommendations, our algorithm introduces a displacement of the boundaries of the system of domains by a random vector (the maximum displacement in each coordinate does not exceed half the length of the subdomain edge), as well as by random shuffling [7] of the order of selection of eight subdomains in the second block. The coordinates of this vector, as well as the sequence of subdomain numbers, are generated by the processor of rank 0 (in MPI_COMM_WORLD) and distributed (MPI_BCAST) along with the trial value of the volume logarithm to all processors at the very beginning of the first block. The second block is executed twice: first, in the forward order of the sequence of subdomain numbers, and then in the reverse order. The loop length over molecular fragments in a subdomain is equal to the number of fragments in it multiplied by 25. This ensures the (asymptotic) equality of the fragment selection frequencies regardless of local density fluctuations. Some unbalancing of the load of processors is compensated by a decrease in the number of potential interaction neighbors due to a decrease in the thickness of the buffer zone (R_{neib}). It should be noted that the fragments that crossed the subdomain boundary remain in the lists of the same subdomain in which they were at the time of the beginning of the second block. This guarantees the reversibility of the Markov chain transitions and ensures that the detailed balance is fulfilled [3, 33]. The determination of whether a fragment belongs to a particular subdomain occurs in the first block, after the boundaries of the entire system of domains have been moved.

Concluding this section, we note two important problems associated with the limited accuracy of the representation of floating point numbers in machine arithmetic. So, as a result of displacements, the length of the covalent bond between the atoms of neighboring fragments can be at the boundary established by the geometric criterion. In this case, it may turn out that the next time the criterion is applied the bond between them will be broken. To avoid this,

one should somewhat narrow (for example, by 0.0001 Å) the corridor of covalent bond lengths admissible as a result of trial moves. It is also necessary to reject trial configurations in which atoms that did not participate in the covalent bond are located at distances near the threshold of this criterion.

Another dangerous situation arises in parallel computing when it is required to determine which subdomain a fragment belongs to, i.e. to solve the problem of the position of its geometric center. In the case when the center is on the border of subdomains, the processors related to this fragment can find mutually contradictory solutions, which will result in the program crashing. To avoid this, the calculation algorithm should be organized in such a way that only the head processors of domain groups are involved in making such decisions, acting only on fragments in their subdomains. As a result of the trial move, the fragment may remain in the same subdomain or move to one of the 26 adjacent subdomains. To ensure conflict-free distribution of fragments across subdomains, it is enough to associate with each fragment a special index indicating the direction of the transition (from 1 to 27) and send it along with the coordinates. Conflict-freeness is guaranteed by the fact that the first time the fragments are distributed by single processor when preparing the initial configuration, and the control of their movement is performed by the head processors of domain groups, each in its own domain.

The computer codes (fortran77 + MPI 1.2) developed by author as well as instructions ‘how to run’ are available from the author on reasonable request.

3. Performance and Scalability of the Computational Algorithm

To verify the program and evaluate the performance and scalability of the underlying algorithm, two well-studied liquids were chosen: water and *n*-hexane. The calculation model of the first one is represented by molecules consisting of one fragment (H_2O), while the molecules of the second one are composed of six fragments connected via single covalent bonds (Fig. 1). For each substance, three variants of simulated systems were prepared and brought into thermodynamic equilibrium at a temperature of 298 K and atmospheric pressure, differing in the size and number of molecules in the main cell (see section ‘Systems under Consideration’). Small-sized systems were divided into 8 ($2 \times 2 \times 2$), medium-sized systems into 64 ($4 \times 4 \times 4$), and large ones into 216 ($6 \times 6 \times 6$) domains. The edge length of the domain of aqueous systems was, on average, 50.15 Å, while that of *n*-hexane systems was 40.3 Å.

In the course of computational experiments, we measured the execution time for the task of calculating the density and heat of evaporation of the simulated liquid, which requires, on average, 5000 trial moves of the Metropolis et al. procedure [2, 18] per each fragment. For each system, a series of calculations was performed, gradually increasing the number of processor cores in domain groups in the following order: 1, 2, 4, 7, 14, 28, and 35. Due to limited computing resources (no more than 80 nodes equipped with 28 cores), no tests were performed on medium-sized systems for 35, and for large models for 28 and 35 cores in domain groups. Recall that $M=m \cdot N$ processor cores are required to run the test. Here N is the number of domains, and m is the number of processor cores in the domain group.

Figure 3 shows the dependence of the test execution time on the number of processor cores in the domain group on a logarithmic scale. Data for large systems ($N=216$ domains, m from 1 to 14) are not shown, as they are only a fraction of a percent higher than the corresponding

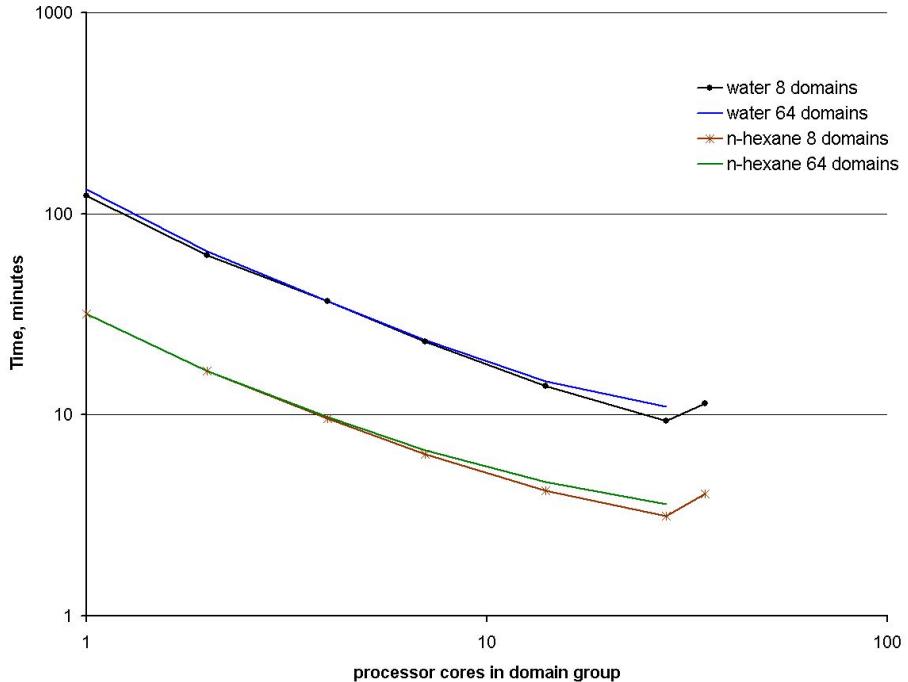


Figure 3. The test execution time for water and *n*-hexane models of small and medium size depending on the number of processor cores in the domain group. The symbols on the lines indicate the results for 1, 2, 4, 7, 14, 28 and 35 processor cores

medium sized systems. The difference in test execution time for small and medium systems is more noticeable and may be due to fewer data exchanges in the case of eight domains [14, 25, 26]. This figure clearly shows that after 28 a further increase in the number of processor cores in domain groups becomes impractical. Note that, obeying the Amdahls law, this effect is universal, since it practically does not depend on the kind of fragments and the presence of covalent bonds between them, as well as on the size of the domains (on average, the domain of the aqueous system contains 4208, and *n*-hexane – 1800 fragments) and the number of interaction partners for each fragment (for a water molecule, on average, about 1380, and for a fragment of an *n*-hexane molecule, 850). As it has been seen in Fig. 4, despite the fact that the Amdahls law limits the increase in performance due to the increase in the number of cores in domain groups, the tenfold acceleration of calculations is a very powerful incentive to use parallelization/splitting of the loop over interaction partners in addition to domain decomposition.

The most important result obtained in these computational experiments is the absence of a noticeable increase in the calculation time with an increase in the size of the simulated system, which is achieved by proportionally increasing the number of processor cores used. The curves in Fig. 3, which show the execution time of test calculations for systems of different sizes, begin to diverge noticeably only when large domain groups of processor cores are used.

As shown by additional calculations, in our model, the root-mean-square deviation of *n*-hexane atoms from the average position after performing 24000 trial moves (on average) for each fragment is 0.76 ± 0.19 Å at T=306 K. In the calculations of the protein molecule by the Monte Carlo method, where parameters of the force fields relevant at that time, a significantly lower value of 0.082 Å was obtained for this characteristic [20]. In turn, it was noted that the rmsd of atoms of the same molecule reaches 0.75 Å after 10^5 time steps performed by the molecular dynamics method. At the same time, the rmsd of *n*-hexane atoms after the same

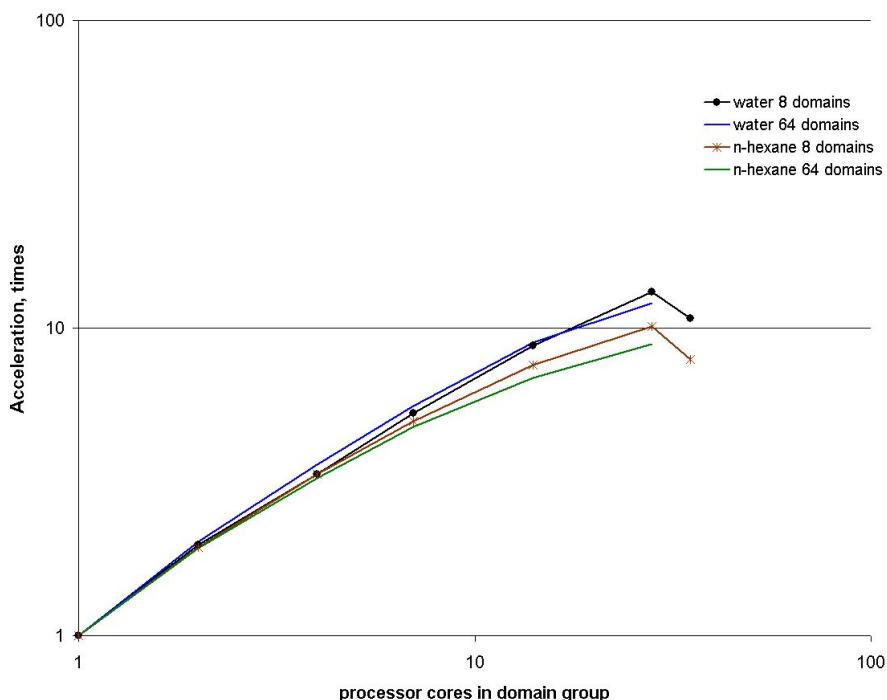


Figure 4. Acceleration of calculations of small and medium-sized water and *n*-hexane models depending on the number of processor cores in the domain group. The symbols on the lines indicate the results for 1, 2, 4, 7, 14, 28 and 35 processor cores

number of trial moves per each fragment is 1.36 ± 0.37 Å in our model. It can be seen that the use of molecular models with moderate stiffness of covalent bonds between fragments makes it possible to surpass the molecular dynamics method in terms of the efficiency of scanning the configuration space of the model by increasing the displacement amplitude of atoms in one step of the computational procedure.

To date, several articles have been published that provide data on the performance of popular software packages. Thus, in the article [15] one can find data on the calculations of the F₁-ATPase system (327506 atoms), comparable in size to one of the systems we considered: the medium-sized model of liquid *n*-hexane (384000 atoms). As seen in Fig. 4 of the article [15], it takes 100, 65 and 35 ms to execute one time step by the NAMD2 package when using 256, 448 (interpolation) and 1000 processor cores on the IBM Blue Gene/L supercomputer. In our case, it takes 117.6, 80.3 and 55.7 ms to perform similar work for the medium-sized liquid *n*-hexane model using 256, 448 and 896 processor cores on the K60 supercomputer (<https://ckp.kiam.ru>). This comparison shows that even without deep, including low-level, code optimization, our approach ensures high performance of the program developed on its basis.

Concluding this section, we recall that the development of a highly efficient program for molecular Monte Carlo calculations requires solving two main problems: taking into account intramolecular degrees of freedom and parallelizing the calculations over the Markov chain underlying the computational algorithm. The first of them is proposed to be eliminated by dividing biopolymer molecules into rigid compact fragments connected by single covalent bonds of moderate stiffness. To solve the second one, domain decomposition in combination with splitting loops over lists of interaction partners is well suited. Note that the implementation of domain decomposition in the Monte Carlo method is fundamentally different from its implementation in the molecular dynamics method, where all atoms are displaced at one time step. We add that

the problem of applying domain decomposition in calculations of molecular systems using the procedure of Metropolis et al. [2, 18] is very poorly covered in the literature, so we believe that this article will provide new information useful for developers of supercomputer programs.

Conclusion

In this paper, we have considered the main factors affecting the performance of Monte Carlo calculations for large models of molecular aggregates, and also proposed effective methods for solving the identified problems. Thus, the use of domain decomposition and distributed computing on hundreds of processor cores not only eliminates the problem of lack of RAM, but also provides the possibility of simultaneous execution of the Metropolis et al. procedure [2, 18] in each domain. The minimum size of a domain is determined by the cutoff radius for interactions between atoms and the thickness of the buffer zone; therefore, each domain can contain several thousand atoms. To further speed up calculations in domains, it is necessary to split the loop over the list of interaction partners, assigning to each domain not one processor core, but a group of 7–14 cores, using data replication. This will further reduce the calculation time by an order of magnitude. The problem of slow diffusion in the configuration space of molecules with internal degrees of freedom [20] is proposed to be solved using a simplified treatment of the strain energy of covalent bonds and angles [27, 31, 32].

As shown by computational experiments with all-atom models of water and *n*-hexane, the strategy of parallelization of calculations proposed by us makes it possible to develop highly efficient programs for modeling biopolymers in an aqueous solution by the Monte Carlo method in the framework of classical physics.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Allen, F.H., Kennard, O., Watson, D.G., *et al.*: Tables of bond lengths determined by X-ray and neutron diffraction. Part 1. Bond lengths in organic compounds. J. Chem. Soc., Perkin Trans. 2 pp. S1–S19 (1987). <https://doi.org/10.1039/P298700000S1>
2. Allen, M.P., Tildesley, D.J.: Computer Simulation of Liquids. Cambridge University Press, New York (1987)
3. Anderson, J.A., Jankowski, E., Grubbb, T.L., *et al.*: Massively parallel Monte Carlo for many-particle simulations on GPUs. J. Comput. Phys. 254, 27–38 (2013). <https://doi.org/10.1016/j.jcp.2013.07.023>
4. Berendsen, H.J.C., Grigera, J.R., Straatsma, T.P.: The missing term in effective pair potentials. J. Phys. Chem. 91, 6269–6271 (1987). <https://doi.org/10.1021/j100308a038>
5. Cornell, W.D., Cieplak, P., Bayly, C.I., *et al.*: A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. J. Am. Chem. Soc. 117, 5179–5197 (1995). <https://doi.org/10.1021/ja00124a002>

6. Dubbeldam, D., Torres-Knoop, A., Walton, K.S.: On the inner workings of Monte Carlo codes. *Mol. Simul.* 39, 1253–1292 (2013). <https://doi.org/10.1080/08927022.2013.819102>
7. Durstenfeld, R.: Algorithm 235: Random permutation. *Commun. ACM* 7, 420 (1964). <https://doi.org/10.1145/364520.364540>
8. Gō, N., Scheraga, H.A.: Analysis of the contribution of internal vibrations to the statistical weights of equilibrium conformations of macromolecules. *J. Chem. Phys.* 51, 4751–4767 (1969). <https://doi.org/10.1063/1.1671863>
9. Hammersley, J.M., Handscomb, D.C.: *Monte Carlo Methods*. Methuen, London (1964)
10. Heffelfinger, G.S., Lewitt, M.E.: A comparison between two massively parallel algorithms for Monte Carlo computer simulation: An investigation in the grand canonical ensemble. *J. Comput. Chem.* 17, 250–265 (1996). [https://doi.org/10.1002/\(SICI\)1096-987X\(19960130\)17:2<250::AID-JCC11>3.0.CO;2-N](https://doi.org/10.1002/(SICI)1096-987X(19960130)17:2<250::AID-JCC11>3.0.CO;2-N)
11. Jorgensen, W.L., Maxwell, D.S., Tirado-Rives, J.: Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.* 118, 11225–11236 (1996). <https://doi.org/10.1021/ja9621760>
12. Jorgensen, W.L., Tirado-Rives, J.: Monte Carlo vs molecular dynamics for conformational sampling. *J. Phys. Chem.* 100, 14508–14513 (1996). <https://doi.org/10.1021/jp960880x>
13. Kalos, M.H., Whitlock, P.A.: *Monte Carlo Methods*. Wiley-VCH, Weinheim (2008)
14. Kalugin, M.D., Teplukhin, A.V.: Study of caffeineDNA interaction in aqueous solution by parallel Monte Carlo simulation. *J. Struct. Chem.* 50, 841–852 (2009). <https://doi.org/10.1007/s10947-009-0126-8>
15. Kumar, S., Huang, C., Zheng, G., et al.: Scalable molecular dynamics with NAMD on the IBM Blue Gene/L system. *IBM Journal of Research and Development* 52(1-2), 177–188 (2008). <https://doi.org/10.1147/rd.521.0177>
16. Landau, D.P., Binder, K.: *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, New York (2000)
17. Mavrntzas, V.G.: Using Monte Carlo to simulate complex polymer systems: Recent progress and outlook. *Front. Phys.* 9, 661367 (2021). <https://doi.org/10.3389/fphy.2021.661367>
18. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., et al.: Equation of state calculations by fast computing machines. *J. Chem. Phys.* 21, 1087–1092 (1953). <https://doi.org/10.1063/1.1699114>
19. Metropolis, N., Ulam, S.: The Monte Carlo method. *J. Am. Statist. Assoc.* 44, 335–341 (1949). <https://doi.org/10.1080/01621459.1949.10483310>
20. Northrup, S.A., McCammon, J.A.: Simulation methods for protein structure fluctuations. *Biopolymers* 19, 1001–1016 (1980). <https://doi.org/10.1002/bip.1980.360190506>
21. Pawley, G.S., Bowler, K.C., Kenway, R.D., Wallace, D.J.: Concurrency and parallelism in MC and MD simulations in physics. *Comput. Phys. Comm.* 37, 251–260 (1985). [https://doi.org/10.1016/0010-4655\(85\)90160-2](https://doi.org/10.1016/0010-4655(85)90160-2)

22. Plimpton, S.: Fast parallel algorithms for short-range molecular dynamics. *J. Comput. Phys.* 117, 1–19 (1995). <https://doi.org/10.1006/jcph.1995.1039>
23. Preis, T., Virnau, P., Paul, W., Schneider, J.J.: GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model. *J. Comput. Phys.* 228, 4468–4477 (2009). <https://doi.org/10.1016/j.jcp.2009.03.018>
24. Rader, A.J.: Coarse-grained models: Getting more with less. *Curr. Opin. Pharmacol.* 10, 753–759 (2010). <https://doi.org/10.1016/j.coph.2010.09.003>
25. Teplukhin, A.V.: Multiprocessor simulation of mesoscopic DNA fragments hydration. *Matem. Model.* 16(11), 15–24 (2004), <http://www.mathnet.ru/links/037077e2238f8f9887f8fa8da9fae408/mm219.pdf>
26. Teplukhin, A.V.: Parallel and distributed computing in problems of supercomputer simulation of molecular liquids by the Monte Carlo method. *J. Struct. Chem.* 54, 65–74 (2013). <https://doi.org/10.1134/S0022476613010095>
27. Teplukhin, A.V.: A simplified treatment of the deformations of covalent bonds and angles during the all-atom Monte Carlo simulation of polymers. *Polym. Sci., ser. C* 55, 103–111 (2013). <https://doi.org/10.1134/S1811238213050044>
28. Teplukhin, A.V.: Short-range potential functions in computer simulations of water and aqueous solutions. *J. Struct. Chem.* 57, 1627–1654 (2016). <https://doi.org/10.1134/S0022476616080205>
29. Teplukhin, A.V.: Monte Carlo simulation of the local ordering of water molecules. II. Spatial correlations and hydrogen bonds. *J. Struct. Chem.* 59, 1624–1630 (2018). <https://doi.org/10.1134/S0022476618070144>
30. Teplukhin, A.V.: Thermodynamic and structural characteristics of SPC/E water at 290 K and under high pressure. *J. Struct. Chem.* 60, 1590–1598 (2019). <https://doi.org/10.1134/S0022476619100044>
31. Teplukhin, A.V.: Monte Carlo calculation of thermodynamic and structural characteristics of liquid hydrocarbons. *J. Struct. Chem.* 62, 7082 (2021). <https://doi.org/10.1134/S002247662101008X>
32. Teplukhin, A.V.: Parametrization of the torsion potential in all-atom models of hydrocarbon molecules using a simplified expression for the deformation energy of valence bonds and angles. *J. Struct. Chem.* 62, 1653–1666 (2021). <https://doi.org/10.1134/S0022476621110019>
33. Uhlherr, A., Leak, S.J., Adam, N.E., *et al.*: Large scale atomistic polymer simulations using Monte Carlo methods for parallel vector processors. *Comput. Phys. Comm.* 144, 1–22 (2002). [https://doi.org/10.1016/S0010-4655\(01\)00464-7](https://doi.org/10.1016/S0010-4655(01)00464-7)
34. Vitalis, A., Pappu, R.V.: Methods for Monte Carlo simulations of biomacromolecules. *Annu. Rep. Comput. Chem.* 5, 49–76 (2009). [https://doi.org/10.1016/S1574-1400\(09\)00503-9](https://doi.org/10.1016/S1574-1400(09)00503-9)
35. Wichmann, B.A., Hill, I.D.: Generating good pseudo-random numbers. *Comput. Statist. Data Anal.* 51, 1614–1622 (2006). <https://doi.org/10.1016/j.csda.2006.05.019>

Calculation of Electrostatic Potential Field of Coronavirus S Proteins for Brownian Dynamics Simulations

Ekaterina P. Vasyuchenko¹, Vladimir A. Fedorov¹ ,
Ekaterina G. Kholina¹ , Sergei S. Khruschev¹ ,
Ilya B. Kovalenko^{1,2,3,4} , Marina G. Strakhovskaya^{1,2} 

© The Authors 2022. This paper is published with open access at SuperFri.org

The Brownian dynamics method can give insight into the initial stages of the interaction of antiviral drug molecules with the structural components of bacteria or viruses. RAM of conventional personal computer allows calculation of Brownian dynamics of interaction of antiviral drugs with individual coronavirus S protein. However, scaling up this approach for modeling the interaction of antiviral drugs with the whole virion consisting of thousands of proteins and lipids is difficult due to high requirements for computing resources. In the case of the Brownian dynamics method, the main amount of RAM in the calculations is occupied by an array of values of the virion electrostatic potential field. When the system is increased from one S protein to the whole virion, the volume of data increases significantly. The standard protocol for calculating Brownian dynamics uses a three-dimensional grid with a spatial step of 1Å to calculate the electrostatic potential field. In this work, we consider the possibility of increasing the grid spacing parameter for calculating the electrostatic potential field of individual coronavirus S proteins. In this case, the amount of RAM occupied by the electrostatic potential field is reduced, which makes it possible to use personal computers for calculations. We performed Brownian dynamics simulations of interaction of an antiviral photosensitizer molecule with S proteins of three coronaviruses SARS-CoV, MERS-CoV, and SARS-CoV-2, and demonstrated that reduction of detailization of electrostatic potential field does not influence the results of Brownian dynamics much.

Keywords: *Brownian dynamics, coarse grain, spike protein, SARS-CoV-2, SARS-CoV, MERS-CoV, phthalocyanine, photosensitizer.*

Introduction

Over the past twenty years, the world has suffered three outbreaks of viral diseases caused by coronavirus, which are expressed mainly in the defeat of the respiratory system and are called acute respiratory syndrome. The causative agents of these epidemics include coronaviruses SARS-CoV, MERS-CoV and SARS-CoV-2. The last pathogen (SARS-CoV-2) was the cause of the COVID-19 pandemic, which affected almost all countries of the world and claimed the lives of 6.33 million people (as of June 2022). Coronaviruses are enveloped viruses consisting of a nucleocapsid and an outer envelope. The outer envelope is represented by a lipid membrane and contains spike (S), membrane (M) and envelope (E) proteins. While E proteins and M proteins are involved in the assembly of virions, the S protein plays a key role in the early stages of eukaryotic cell infection, as it recognizes and binds to the host ACE2 receptor and is responsible for the penetration of the genetic material of the virus into the cell [7]. In this regard, the structures of S proteins are being actively studied, since it is a good target for the search for small molecules that would limit or make it impossible for coronavirus virions to bind to cells. At the moment, the spatial structures of the S proteins of the above viruses are already known. Previ-

¹Lomonosov Moscow State University, Moscow, Russia

²Federal Scientific and Clinical Center of Specialized Types of Medical Care and Medical Technologies, Federal Medical-Biological Agency of Russia, Moscow, Russia

³Astrakhan State University, Astrakhan, Russia

⁴Institute of Computer Science and Mathematical Modeling, Sechenov First Moscow State Medical University (Sechenov University), Moscow, Russia

ously, a study of the interaction of zinc phthalocyanine and methylene blue molecules with the S protein of coronavirus was carried out using the Brownian dynamics method using the ProKSim software package, the main areas of landing of these small molecules on the protein surface and their interaction with key amino acids were shown [2]. In that study, the full-atomic structure of the S proteins of three coronaviruses was used, and the calculation of the electrostatic potential was carried out in 1 Å steps. However, scaling up this approach for modeling the interaction of small molecules with potential antiviral activity with the structure of the whole virion is difficult due to high requirements for computer RAM. In the case of the Brownian dynamics method, the main amount of RAM in the calculations is occupied by an array of values of the electrostatic potential field. When the system is increased from one protein to the whole virion, the volume of the data increases significantly. The standard protocol for calculating Brownian dynamics uses a three-dimensional grid with a step of 1 Å to calculate the electrostatic potential field. In this work, we will consider the possibility of increasing the grid spacing parameter for calculating the electrostatic potential field to 2 Å using the examples of individual SARS-CoV, MERS-CoV, and SARS-CoV-2 S proteins. Thus, the amount of RAM occupied by the electrostatic potential field is reduced by a factor of 8, which makes it possible to use personal computers for modeling the interaction of antiviral drugs with the whole virion.

The article is organized as follows. Section 1 is devoted to the methods used in this work and the specifics of the design of models of studied molecules. In Section 2, we described and discuss the main results of this study. The Conclusion summarizes the results of the study and indicates directions for further work.

1. Methods

1.1. 3D Protein Models

In this work, three-dimensional structures of S proteins of three human coronaviruses (MERS-CoV, SARS-CoV, SARS-CoV-2) were used. The spatial structure of the SARS-CoV-2 S protein was taken from [9]. The structures of the “heads” of the MERS-CoV and SARS-CoV S proteins were taken from the PDB (Protein Data Bank), PDB ID 6NB3 and 5X58. Unresolved remains in the “head” of the spike were completed using the i-TASSER program [10]. Based on the amino acid sequence from the UniProt database (A0A140AYW5 for MERS-CoV and P59594 for SARS-CoV), their secondary structure was predicted using the Jpred4 package [1]. Based on the predicted secondary structure, the unresolved parts of the “core” part of the S protein were completed using the Modeller-9.19 program [8] using the PDB ID 2WPQ as template.

The model of the Zn-PcChol⁸⁺ molecule was taken from [6]. In that work, to determine possible positions of choline substitutes in the phthalocyanine macrocyclic core, we calculated Fukui functions, reflecting the probability of electrophilic attack of substituents to isoindole rings. Taking into account these results, the steric effects and the electrostatic repulsion of substituents, we determined the most plausible structure of Zn-PcChol⁸⁺ with specific positions of choline substituents. Then quantum mechanical calculations were carried out in the framework of the density functional theory using PC GAMESS/Firefly software. We performed two-cycle geometry optimization of Zn-PcChol⁸⁺ molecule using the SBKJC basis set with energy-consistent pseudopotential (ECP) and cc-PVDZ basis set for all atoms but Zn for the final optimization. Atomic partial charges were fit to reproduce the ab initio electrostatic potential using the RESP algorithm. Then we obtained all-atom topology for the predicted Zn-PcChol⁸⁺ structure in the

GROMOS 54a7 force field using the online service Automated Topology Builder (ATB) and performed all-atomic MD for 50 ns. On the basis of auxiliary all-atom simulation models we created the CG model of Zn-PcChol⁸⁺ molecule in the Martini force field. The force field parameters of bonded interactions of CG model obtained from the series of short CG simulations (10 ns) were iteratively optimized based on all-atom MD simulation. The distributions of bond and angle terms governing the equilibrium bond/angle and force constant values at each CG iteration step were compared with all-atom MD simulation until 85% coverage between all-atom and CG distributions. The final CG model of Zn-PcChol⁸⁺ molecule consists of 62 CG beads, 50 bond terms, 18 constraints, 24 angle and 20 dihedrals terms.

To convert an all-atom structure of S protein to CG structure and create the suitable topology in Martini 3 force field, we used martinize2 program (<http://cgmartini.nl>). The general principle of Martini CG modeling of proteins is based on the suggestion that each amino acid residue consisted of one backbone bead and several side chain CG beads. Each CG bead represents group of 2–4 heavy atoms which belong to the certain chemical group. Martini CG force field suggests that the majority of CG bead types are neutrally charged, while few of them have +1 or -1 elementary electric charge.

1.2. Brownian Dynamics Simulation

For Brownian dynamics simulation, calculation of the electrostatic field and modeling of long-range electrostatic interactions, the ProKSim software package was used [4]. The values of the charges of coarse-grained models were taken according to the Martini-3.0.b.3.2 force field (<http://cgmartini.nl/index.php/martini3beta>). The values of the electrostatic potential created by the molecules were calculated on a cubic spatial grid using the Poisson-Boltzmann equation [3] (for more details, see [5]). The grid step was a parameter and was taken equal to 1 Å or 2 Å. Protein molecules and Zn-PcChol⁸⁺ were described as rigid particles placed in a 30x30x30 nm cubic virtual reaction volume with mirror boundary conditions such that the TM (transmembrane) and CP (cytoplasmic) domains were outside the reaction volume for imitation of the viral membrane. Electric permittivity of proteins $\epsilon=2$, water $\epsilon=80$. The ionic strength of the solution was 100 mM, temperature T=300K. Water was taken into account as an implicit solvent with a water viscosity coefficient corresponding to a temperature of 300 K. The effect of thermal motion of water molecules is taken into account by introducing a random force with normal distribution acting on Brownian particles. Van der Waals interactions are taken into account using the excluded volume effect: particles cannot approach a shorter distance than the sum of their van der Waals radii. For each of the three S proteins, 20000 Brownian dynamics simulations were performed. In each simulation, the Zn-PcChol⁸⁺ molecule was placed in a random position in the reaction volume and after diffusion and electrostatic interactions with S protein it approached the protein and formed encounter complex. In this process, the Zn-PcChol⁸⁺ molecule diffused until the energy of electrostatic attraction in such a complex would be not less than specified threshold value of 8kT. The structure of the complex was then saved for further analysis. Visualisation of the complexes was carried out in PyMol-2.5.2 software (<https://pymol.org>).

2. Results and Discussion

The main amount of memory in the Brownian dynamics calculation is occupied by the electrostatic potential of the interacting particles and its gradient. In this work, we calculated the electrostatic potential for the S protein molecules of MERS-CoV, SARS-CoV, and SARS-CoV-2 coronaviruses on the spatial grid with cell size of 1 Å and 2 Å. Table 1 shows the quantitative characteristics of these calculations; in particular, the amount of memory occupied by the arrays for storing the electrostatic potential and its gradient, as well as the total virtual memory occupied by the program.

Table 1. The amount of memory occupied for each S protein/Zn-PcChol⁸⁺ system

	Grid spacing for ES potential calculation (Å)	Number of cells per edge of the cube of the ES potential field	Array size of ES potential (MB)	Size of the ES potential gradient array (MB)	Process virtual memory (MB)
MERS-CoV	1	530	1136	3408	4586
SARS-CoV	1	542	1215	3645	4901
SARS-CoV-2	1	544	1228	3684	4955
MERS-CoV	2	266	144	431	595
SARS-CoV	2	272	154	460	635
SARS-CoV-2	2	272	154	460	635

RAM of conventional personal computer allows calculation of Brownian dynamics of interaction of antiviral drugs with individual coronavirus S protein. For whole SARS-CoV-2 virion envelope the amount of memory occupied would be as much as 150 GB, which significantly exceeds the memory of a typical personal computer. The amount of RAM required to store arrays of the electrostatic potential and its gradient is about 8 times less when using a 2 Å grid compared to a 1 Å grid. From Tab. 1 it follows that almost all the virtual memory of the program is occupied by the ES potential and its gradient.

We performed Brownian dynamics simulations using two grid spacing values of the electrostatic potential to reveal differences in the interaction of Zn-PcChol⁸⁺ with coronavirus S proteins. Figure 1 shows individual electrostatic encounter complexes of the photosensitizer molecule with S proteins of MERS-CoV, SARS-CoV and SARS-CoV-2 coronaviruses calculated with two different values of electrostatic potential grid spacing, namely 1 Å and 2 Å. The position of the Zn-PcChol⁸⁺ molecule in the encounter complex is represented by the zinc atom visualized with a marine-color sphere.

It can be seen that Zn-PcChol⁸⁺ molecules mainly bind to the protein in the upper part of the stalk, at the junction of the stalk with the head, which is consistent with the results of previous studies [2]. Zn-PcChol⁸⁺ molecules are distributed highly heterogeneously on the surfaces of S proteins, forming distinct dense areas. Size and population of each area are characterized by average pairwise RMSD of these molecules from each other, and the number of Zn-PcChol⁸⁺ molecules forming it (Tab. 2). Positions of these areas on S protein surface are pretty the same in cases of 1 Å and 2 Å electrostatic potential grid. In general, the size of the areas slightly differs (no more than 0.3 Å). With increased 2 Å step of the grid of electrostatic potential, small and thinly populated areas located at the head of coronavirus S protein disappear, thus the population of large areas increases and the portion of Zn-PcChol⁸⁺ bound with the stalk of coronavirus

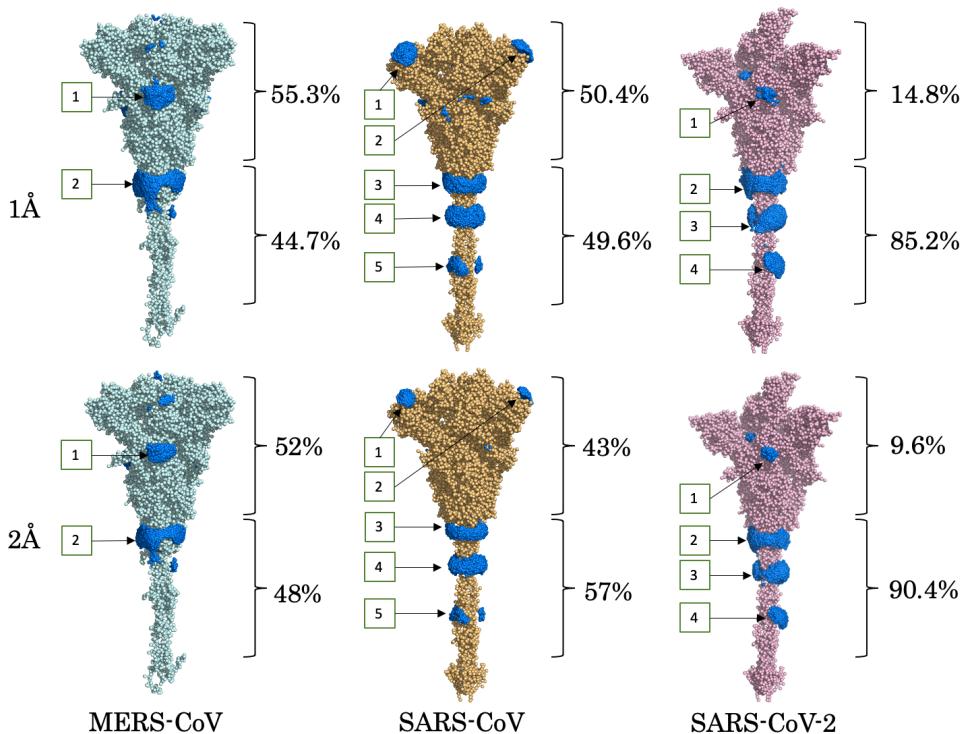


Figure 1. Encounter complexes of the Zn-PcChol⁸⁺ photosensitizer molecule in complexes with S proteins of the studied MERS-CoV, SARS-CoV and SARS-CoV-2 coronaviruses at a grid step for calculating the electrostatic potential of 1 Å (upper row) and 2 Å (lower row) obtained by the Brownian dynamics method. Final positions of Zn-PcChol⁸⁺ molecules obtained in individual BD simulations are represented by the zinc atoms (marine-color spheres). S proteins are presented as a coarse-grained model and colored, respectively, MERS-CoV – pale cyan, SARS-CoV – light orange, SARS-CoV-2 – light pink. Isolated Zn-PcChol⁸⁺ binding areas are numbered. Ratio of Zn-PcChol⁸⁺ bound with head and stalk of coronavirus is given in percent

Table 2. Percentage of structures forming individual binding areas and average pairwise RMSD for this area in Å (value given in parentheses)

	Grid spacing for ES potential calculation (Å)	1 binding area	2 binding area	3 binding area	4 binding area	5 binding area
MERS-CoV	1	15.7 (1.8)	44.7 (3.0)			
	2	17.8 (1.8)	48.0 (2.9)			
SARS-CoV	1	12.9 (1.5)	12.7 (1.4)	22.5 (2.7)	16.1 (2.4)	11.0 (2.3)
	2	14.9 (1.2)	12.7 (1.1)	22.9 (2.5)	28.8 (2.2)	5.4 (2.2)
SARS-CoV-2	1	9.7 (1.1)	47.4 (2.8)	25.1 (2.3)	12.8 (1.7)	
	2	2.0 (1.1)	49.7 (2.7)	27.7 (2.2)	12.9 (1.4)	

S protein increases. The results show that bigger step of electrostatic potential grid leads to slight changes in distribution of Zn-PcChol⁸⁺ molecules on coronavirus S proteins. Nevertheless,

the main areas of binding of photosensitizer molecules can be found using rough electrostatic potential field which takes not so much memory resources.

In calculations with different values of the grid spacing of the electrostatic potential, the regions of the protein interacting with Zn-PcChol⁸⁺ do not change. However, the occupation of minor binding sites differs between simulations with two cell grid steps (1Å and 2Å), since in these positions the specified energy of electrostatic attraction (8kT) cannot be achieved due to the coarse representation of the electrostatic potential field when using 2Å resolution. Nevertheless, this does not change the general distribution of Zn-PcChol⁸⁺. Thus, the coarsening of the spatial grid for calculation of electrostatic potential makes it possible to perform Brownian dynamics simulations using a smaller amount of RAM while obtaining a reliable result.

Conclusion

With the advent of a large number of molecular spatial structures of biological objects, there is a need to create models of large systems, such as viral envelopes, microtubules, membrane pigment-protein complexes, lipopolysaccharide membranes, etc. However, the available computing resources remain limited. In this work, we evaluated the possibility of reducing the requirements for computer RAM by reducing the detail of the electrostatic potential of molecules, without losing the quality of the data obtained. In the case of the Brownian dynamics method, the main amount of RAM in the calculations is occupied by an array of values of the electrostatic potential field. We considered the possibility of increasing the grid spacing for calculating the electrostatic potential field to 2Å using the individual S proteins of SARS-CoV, MERS-CoV and SARS-CoV-2 and found that the main binding regions of the photosensitizer molecule can be found using a coarse electrostatic potential field, which requires not so many memory resources. Indeed, with a twofold increase in the grid spacing parameters, the amount of RAM occupied by the electrostatic potential field decreases by a factor of 8, which makes it possible to perform electrostatic potential and Brownian dynamics calculations on a personal computer for large molecular biological systems.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Drozdetskiy, A., Cole, C., Procter, J., Barton, G.J.: JPred4: a protein secondary structure prediction server. Nucleic acids research 43(W1), W389–W394 (2015). <https://doi.org/10.1093/nar/gkv332>
2. Fedorov, V.A., Kholina, E.G., Khruschev, S.S., et al.: What binds cationic photosensitizers better: Brownian dynamics reveals key interaction sites on spike proteins of SARS-CoV, MERS-CoV, and SARS-CoV-2. Viruses 13(8), 1615 (2021). <https://doi.org/10.3390/v13081615>
3. Fogolari, F., Brigo, A., Molinari, H.: The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology. Journal of Molecular Recognition 15(6), 377–392 (2002). <https://doi.org/10.1002/jmr.577>

4. Khruschev, S.S., Abaturova, A.M., Diakonova, A.N., *et al.*: Brownian-dynamics simulations of protein–protein interactions in the photosynthetic electron transport chain. *Biophysics* 60(2), 212–231 (2015). <https://doi.org/10.1134/S0006350915020086>
5. Kovalenko, I.B., Khruschev, S.S., Fedorov, V.A., *et al.*: The role of electrostatic interactions in the process of diffusional encounter and docking of electron transport proteins. *Doklady Biochemistry and Biophysics* 468(1), 183–186 (2016). <https://doi.org/10.1134/S1607672916030066>
6. Orekhov, P.S., Kholina, E.G., Bozdaganyan, M.E., *et al.*: Molecular mechanism of uptake of cationic photoantimicrobial phthalocyanine across bacterial membranes revealed by molecular dynamics simulations. *The Journal of Physical Chemistry B* 122(14), 3711–3722 (2018). <https://doi.org/10.1021/acs.jpcb.7b11707>
7. Schoeman, D., Fielding, B.C.: Coronavirus envelope protein: current knowledge. *Virol J.* 16(1), 1–22 (2019). <https://doi.org/10.1186/s12985-019-1182-0>
8. Webb, B., Sali, A.: Comparative protein structure modeling using modeller. *Current protocols in bioinformatics* 54(1), 5–6 (2016). <https://doi.org/10.1002/cpbi.3>
9. Woo, H., Park, S.J., Choi, Y.K., *et al.*: Developing a fully glycosylated full-length SARS-CoV-2 spike protein model in a viral membrane. *The Journal of Physical Chemistry B* 124(33), 7128–7137 (2020). <https://doi.org/10.1021/acs.jpcb.0c04553>
10. Zheng, W., Zhang, C., Li, Y., *et al.*: Folding non-homologous proteins by coupling deep-learning contact maps with I-TASSER assembly simulations. *Cell reports methods* 1(3), 100014 (2021). <https://doi.org/10.1016/j.crmeth.2021.100014>

Computational Alchemy Using Accelerated GPU Calculations: Fine Structural Tuning Inhibitors of L,D-transpeptidase 2 from *Mycobacterium tuberculosis*

Semen M. Baldin^{1,2} , Vsevolod O. Shegolev^{1,3} , Vytaus K. Švedas^{1,2,4} 

© The Authors 2022. This paper is published with open access at SuperFri.org

The computational alchemy is based on equilibrium methods of molecular dynamics (MD), including thermodynamic integration (TI) or free energy perturbation (FEP). It makes it possible to evaluate the energy characteristics of a subtle alchemical transformation of the prototype compound structures in order to modulate their functional properties, what can be used searching for new analogues of enzyme inhibitors. In recent years, the method is gaining new opportunities due to high-performance computing as different MD engines got the ability to perform TI or FEP calculations using GPU acceleration. We have tested the method on two experimentally studied enzyme-inhibitor systems: penicillin acylase from *E. coli* (PaEc) and Influenza A virus neuraminidase (NAIAv) and performed computational alchemy calculations for pairs of their known inhibitors. Thus, the validated methodology was then applied to estimate binding of L,D-transpeptidase 2 from *M. tuberculosis* (Ldt_{Mt2}) to the structural analogs of inhibitors previously discovered in our group. We have found that the performance of computational alchemy predictions is highly dependent on properly chosen simulation parameters and at optimal performance the difference between the calculated and experimentally determined relative binding energies is less than 1 kcal/mol. Thus, for optimal application to novel enzyme—inhibitor or receptor—ligand systems, we recommend prior adaptation of this method, based on validation of force field combinations as well as water models, to experimentally determined binding data of structurally related inhibitors/ligands to the molecular target of interest, if available, and then to proceed to the study of new compounds. High-performance computational alchemy can be used as a useful integrative part of the methodology searching for new enzyme inhibitors or receptor ligands and optimizing their structure at drug design.

Keywords: computational alchemy, thermodynamic integration, protein—ligand binding, binding free energy, drug design, L,D-transpeptidase.

Introduction

In classical molecular modeling, the veracity of calculation is believed to be a result of physically accurate approximations; contrariwise the molecular alchemy approach is dealing with non-physically, but mathematically correct models, which, on the one hand, can be even more precise than strict-physical models, and, on the other hand, adds new options to decrease computational efforts. The computational alchemy approach can be applied to solve some problems of computational chemistry (by so-called *alchemical transformation*, or “transmutation”) such as calculation of the energy of hydration, estimation of relative changes of binding constants K_i for substrates or inhibitors, all this by changing structures of ligands or amino acid residues even within a binding site. The purpose of this work is to show how this approach can be applied to process a structure fine-tuning of known inhibitors in case to find new analogues with lower (better) inhibition constants K_i .

¹Lomonosov Moscow State University, Belozersky Institute of Physicochemical Biology, Moscow, Russia

²Lomonosov Moscow State University, Research Computing Center, Moscow, Russia

³Lomonosov Moscow State University, Faculty of Chemistry, Moscow, Russia

⁴Lomonosov Moscow State University, Faculty of Bioengineering and Bioinformatics, Moscow, Russia

1. Theoretical Aspects

1.1. Statistical Mechanics

In canonical ensemble the number of microstates accessible to the system is described by the partition function Q . For the state \mathcal{A} of the system,

$$Q_{\mathcal{A}}(N, V, T) = \frac{1}{N!h^{3N}} \iint e^{-\beta \mathcal{H}_{\mathcal{A}}(\mathbf{p}, \mathbf{q})} d\mathbf{p} d\mathbf{q}, \quad (1)$$

where N is the number of (strictly speaking, indistinguishable) particles in the system, $\mathcal{H}_{\mathcal{A}}(p, q)$ is the Hamiltonian depending on generalized momenta $\mathbf{p} = \mathbf{p}_1, \dots, \mathbf{p}_N$ and coordinates $\mathbf{q} = \mathbf{q}_1, \dots, \mathbf{q}_N$ of all particles in the system, h is the Planck's constant, $\beta = 1/kT$ is the "thermodynamic beta", and k is the Boltzmann's constant.

Free energy A of the system in a state \mathcal{A} , and its difference between two states \mathcal{A} and \mathcal{B} of the system can be written using (1) as

$$A_{\mathcal{A}} = -\frac{1}{\beta} \ln Q_{\mathcal{A}} \quad \text{and} \quad (2)$$

$$\Delta A_{\mathcal{AB}} = A_{\mathcal{B}} - A_{\mathcal{A}} = -\frac{1}{\beta} \ln \frac{Q_{\mathcal{B}}}{Q_{\mathcal{A}}} \quad \text{accordingly.} \quad (3)$$

In molecular dynamics, the expression of the Hamiltonian \mathcal{H} of the system of N particles with masses m_i and Cartesian coordinates $\mathbf{r} = \mathbf{r}_1, \dots, \mathbf{r}_N$ [21] is

$$\mathcal{H}(\mathbf{p}, \mathbf{r}) = \sum_{i=1}^N \frac{\mathbf{p}_i^2}{2m_i} + U(\mathbf{r}_1, \dots, \mathbf{r}_N), \quad (4)$$

potential term U of the Hamiltonian is usually defined⁵ [6] as

$$U = \sum_b k_b (l_b - \bar{l}_b)^2 + \sum_a k_a (\theta_a - \bar{\theta}_a)^2 \quad (5)$$

$$+ \sum_d \sum_n k_{d,n} \{ 1 + \cos(n\phi_d - \phi'_d) \} \quad (6)$$

$$+ \sum_{i < j} \left\{ \frac{A_{i,j}}{r_{i,j}^{12}} - \frac{B_{i,j}}{r_{i,j}^6} + k_e \frac{q_i q_j}{r_{i,j}} \right\}, \quad (7)$$

where the energy of deformation from the average value is approximated by Hooke's quadratic potentials (5) for bond length l_b and valence angles θ_a , and by periodic Fourier series (6) for dihedrals ϕ_d and its harmonics n ; pairwise particles i, j interactions (7) are described by Lennard-Jones "6-12" potential and Coulomb's law and decrease in distance $r_{i,j}$; $k_b, \bar{l}_b, k_a, \bar{\theta}_a, k_{d,n}, \phi'_d, A_{i,j}, B_{i,j}, k_e, q_i$, and q_k are the relevant empirical parameters which consist the *force field*.

1.2. Free Energy Perturbation

Substituting the expression (5–7) for U in (4), then in (1), and then in (3), reducing the fraction $Q_{\mathcal{A}}/Q_{\mathcal{B}}$ in (3) by the same kinetic energy term $\exp \sum_i \frac{\mathbf{p}_i^2}{2m_i}$ leads to

$$\Delta A_{\mathcal{AB}} = -\frac{1}{\beta} \ln \frac{Z_{\mathcal{B}}}{Z_{\mathcal{A}}}, \quad (8)$$

⁵in AMBER family of force fields

where configurational partition function Z depends only on coordinates \mathbf{r} , e.g., for the \mathcal{B} state:

$$Z_{\mathcal{B}}(N, V, T) = \int e^{-\beta U_{\mathcal{B}}(\mathbf{r})} d\mathbf{r}, \quad (9)$$

this equation can be rewritten in terms of the ensemble averages as

$$Z_{\mathcal{B}} = \int e^{-\beta U_{\mathcal{B}}} d\mathbf{r} = \int e^{-\beta U_{\mathcal{B}}} e^{\beta U_{\mathcal{A}}} e^{-\beta U_{\mathcal{A}}} d\mathbf{r} = \int e^{-\beta(U_{\mathcal{B}} - U_{\mathcal{A}})} \underbrace{e^{-\beta U_{\mathcal{A}}}}_{\rho_{\mathcal{A}}} d\mathbf{r} = \langle e^{-\beta(U_{\mathcal{B}} - U_{\mathcal{A}})} \rangle_{\mathcal{A}}, \quad (10)$$

where $\rho_{\mathcal{A}}$ is a distribution function of the state \mathcal{A} , $\langle X \rangle_{\mathcal{A}}$ denotes the average of X in ensemble with the distribution $e^{-\beta U_{\mathcal{A}}}$, and finally, (8) takes the form named *free energy perturbation formula* [26]:

$$\Delta A_{\mathcal{AB}} = -\frac{1}{\beta} \ln \langle e^{-\beta(U_{\mathcal{B}} - U_{\mathcal{A}})} \rangle_{\mathcal{A}}. \quad (11)$$

Physically, (11) means that microstates of the state \mathcal{A} can be used to estimate the average of the function $e^{-\beta \Delta_{\mathcal{AB}} U}$ and then calculate transitional free energy $\Delta A_{\mathcal{AB}}$ between two states \mathcal{A} and \mathcal{B} , it means that microstates of state \mathcal{A} must also be the microstates of the state \mathcal{B} with a sufficiently high probability to get an accurate estimate of the average $\langle \dots \rangle_{\mathcal{A}}$, i.e., the state \mathcal{B} must be only the *small perturbation* of the state \mathcal{A} .

1.3. Thermodynamic Integration

Equation (11) can be generalized to estimate $\Delta A_{\mathcal{AB}}$ between arbitrary states \mathcal{A} and \mathcal{B} , close or not. It is only enough to maintain intermediate states s (with distributions $e^{-\beta U_s}$) between \mathcal{A} and \mathcal{B} , satisfying the “closeness” condition:

$$\Delta A_{\mathcal{AB}} = -\frac{1}{\beta} \sum_s \ln \langle e^{-\beta(U_{s+1} - U_s)} \rangle_s, \quad (12)$$

or, in continuous case, integration through the transformational path λ can be proceeded by defining a sufficient transformation between two states in potential term of the Hamiltonian:

$$U(\lambda) = \lambda U_{\mathcal{A}} + (1 - \lambda) U_{\mathcal{B}}, \quad (13)$$

from hence one can obtain a formula

$$\Delta A_{\mathcal{AB}} = \int_0^1 \left\langle \frac{\partial U}{\partial \lambda} \right\rangle_{\lambda} d\lambda, \quad (14)$$

by using (2) and the following:

$$\begin{aligned} \frac{\partial A}{\partial \lambda} &= -\frac{1}{\beta Q} \frac{\partial Q}{\partial \lambda} = -\frac{1}{\beta Z} \frac{\partial Z}{\partial \lambda} = -\frac{1}{\beta Z} \frac{\partial}{\partial \lambda} \int e^{-\beta U(\mathbf{q}; \lambda)} d\mathbf{q} = \\ &= -\frac{1}{\beta Z} \int \left(-\beta \frac{\partial U(\mathbf{q}; \lambda)}{\partial \lambda} \right) e^{-\beta U(\mathbf{q}; \lambda)} d\mathbf{q} = \left\langle \frac{\partial U(\lambda)}{\partial \lambda} \right\rangle_{\lambda}. \end{aligned} \quad (15)$$

The resulting relation (14) is called the *thermodynamic integration* formula. In practice, from integration again switch to summation:

$$\Delta A_{\mathcal{AB}} \approx \sum_s \left\langle \frac{\partial U}{\partial \lambda} \right\rangle_{\lambda_s} \delta \lambda_s. \quad (16)$$

1.4. Computational Alchemy

As mentioned above, thermodynamic integration (14) can be performed through a transformational path λ connecting any two states of the system. Let us suppose, just conceptually, at the start, we have a system of a ligand in water, at the state \mathcal{A} , the ligand has a chemical formula L_1 , during the all “alchemical” path λ , the ligand is continuously transforming into the chemical formula L_2 , corresponding to the state \mathcal{B} . In the other words, a chemical molecule undergoes non-physical transformation, or “transmutation” during which the whole groups of its atoms are changing their chemical nature. The described procedure can be easily implemented *in silico* in two different ways.

1.4.1. The single-topology paradigm

Empirical parameters of the force field (5–7), can be transformed during the molecular dynamics simulation, e.g., phenol molecule can be transmuted to thiophenol by changing the numerical parameters of the oxygen atom O into the sulfur S through the “fictitious” atom $X(\lambda)$ which is in the functional dependence on path λ (Fig. 1).

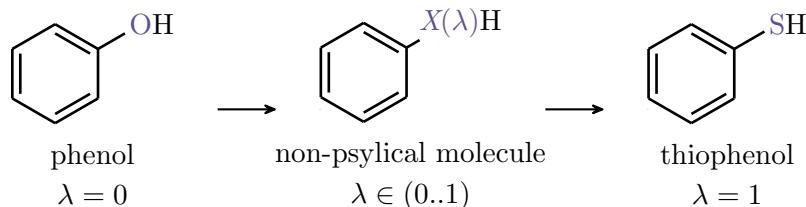


Figure 1. Scheme of alchemical transformation in single-topology paradigm

Mathematically, all force field parameters of the transforming atoms are changing along the path $\lambda = 0..1$:

$$\begin{cases} m_X(\lambda) = \lambda m_O + (1 - \lambda)m_S, \\ \vdots \\ q_X(\lambda) = \lambda q_O + (1 - \lambda)q_S. \end{cases} \quad (17)$$

Then, calculated parameters involve into forces calculation as (5–7), and integration of motion equation according to the Newton’s 2nd law. In that case, the one molecule topology is transforming.

1.4.2. The dual-topology paradigm

Alternatively, Equation (13) can be implemented more straightforwardly by combining two different topologies of the physical molecules at the start \mathcal{A} and at the end \mathcal{B} of the alchemical path λ . Pairs of atoms, common for both topologies, share their coordinates and are treated as a single atom in all interactions with the environment. In contrast, differing atoms do not undergo a transformation as in the single-topology paradigm, instead, they do not sense the existence of each other, but their interactions with the environment are scaled according to covered path λ (Fig. 2). So, from the outside, the molecule looks like a chimera of two.

Values of λ can lay anywhere on the interval $[0..1]$. Because Lennard-Jones’s and electric potential tends to $+\infty$ when $r \rightarrow 0$, the transition from $\lambda \equiv 0$ to $\lambda \sim 0$ will be unstable because a potential of appearing atoms will grow explosively in regions of $r \sim 0$, so some “soft potential”

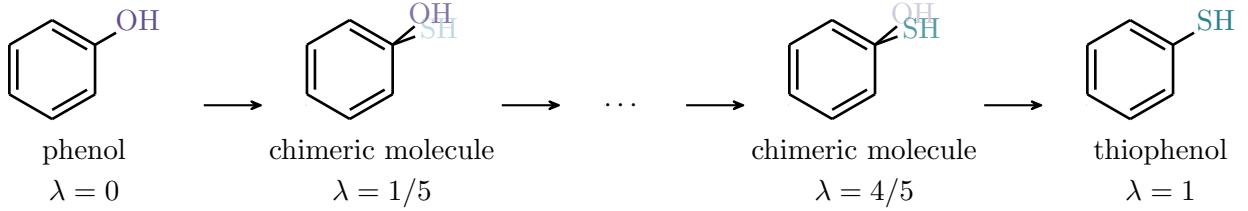


Figure 2. Scheme of alchemical transformation in dual-topology paradigm

approximation is used [2, 18]:

$$U_{i,j}^{\text{LJ}}(r_{i,j}, \lambda) = 4\epsilon_{i,j}\lambda^n \left\{ \frac{1}{[\alpha(1-\lambda)^2 + (r_{i,j}/\sigma_{i,j})^6]^2} - \frac{1}{\alpha(1-\lambda)^2 + (r_{i,j}/\sigma_{i,j})^6} \right\}, \quad (18)$$

where $\epsilon_{i,j}, \sigma_{i,j}$ are Lennard-Jones parameters, n and α are numerical constants; and similarly for electrostatics:

$$U_{i,j}^{\text{Col}}(r_{i,j}, \lambda) = (1-\lambda) \frac{q_i q_j}{4\pi \varepsilon_0 \sqrt{\beta \lambda + r_{i,j}^2}}, \quad (19)$$

where ε_0 is the permittivity, q_i and q_j are charges, β is a numeric constant.

1.5. Thermodynamic Cycle

Computational alchemy by itself is not very usable, however, the power of this technique is revealed when using a hybrid experimental-computational approach. In the previous section, we have described the transformation $L_1 \rightarrow L_2$ for a ligand in bulk. In the same way, the ligand can be transformed while being in a complex with a protein E, i.e. $EL_1 \rightarrow EL_2$. Thus, we can construct a thermodynamic cycle (Fig. 3): up and down edges of the cycle correspond to free energies $\Delta_{\text{alc}}A_{\text{lig}}$ (free ligand) and $\Delta_{\text{alc}}A_{\text{com}}$ (ligand in a complex) of alchemical transformations, left and right, to experimentally measurable ligands-protein binding energies Δ_bG .

The expression of the free energy A is defined by the thermodynamic ensemble, for canonical NVT , $A \equiv F$ (the Helmholtz free energy); in isothermal-isobaric NPT ensemble, $A \equiv G$ (the Gibbs free energy). Nevertheless, in condensed phases, $\Delta G \approx \Delta F \approx \Delta A$ in a large pressure range [12], further, we will imply that.

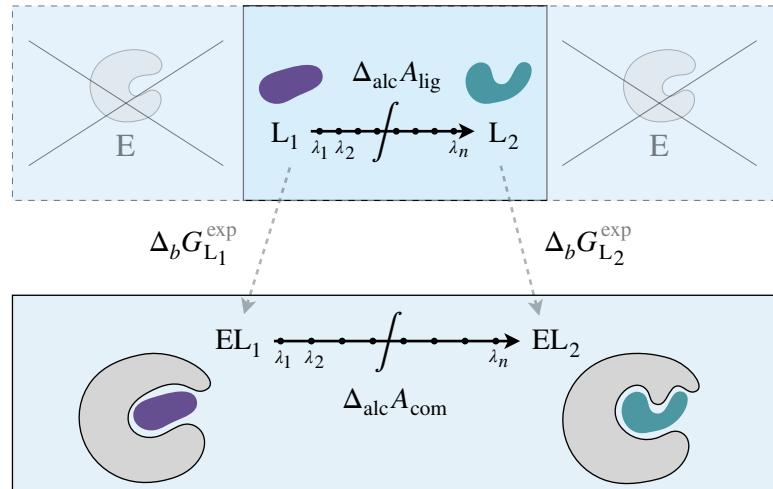


Figure 3. Thermodynamic cycle for calculation of the relative binding energy of two ligands L_1 and L_2 in the active site of the protein E

Having an experimental value $\Delta_b G_{L_1}$ for binding of L_1 one can estimate $\Delta_b G_{L_2}$ for an arbitrary ligand L_2 by using the thermodynamic cycle:

$$\begin{aligned}\Delta_b G_{L_2} &= \Delta_b G_{L_1} + \Delta_{\text{alc}} A_{\text{com}} - \Delta_{\text{alc}} A_{\text{lig}} \\ &= \Delta_b G_{L_1} + \underbrace{\Delta_{\text{alc}} G_{\text{com}} - \Delta_{\text{alc}} G_{\text{lig}}}_{\Delta \Delta_{\text{alc}} G},\end{aligned}\quad (20)$$

where $\Delta \Delta_{\text{alc}} G$ is the computed alchemical correction for the experimental $\Delta_b G_{L_1}$. Hence, the ratio for the relative binding energies is

$$\Delta \Delta_b G = \Delta \Delta_{\text{alc}} G. \quad (21)$$

2. Acceleration of TI MD Calculations Using GPU

Although the acceleration of GPU calculations for the molecular dynamics (MD) method has been implemented in many MD engines for a while, the GPU acceleration for the thermodynamic integration (TI) method has been tested relatively recently [9, 10]. It should be noted that the motivation for the implementation of GPU acceleration for TI method was to speed up calculations of relative binding energies of inhibitors with enzymes in order to adjust the inhibitor structure to the geometry of the binding site [7].

Let us discuss a typical computational task for estimation of the relative binding energy of a pair of inhibitors in a binding site of an enzyme. As it was previously revealed, to estimate the free energy of the alchemical transformation we need to do two types of calculations: transformation in the bulk (*ligands*) and in a binding site of an enzyme (*complex*).

For the first system, considering the distance from the solute to the modeling cell boundaries is from 12 to 15 Å, the size of such a box will be about 5K to 10K atoms. For the second system, the size of the solvate box will be primarily determined by the size of the enzyme model. Using the examples described later in this paper, one can find the sizes of the simulated systems: for the model of L,D-transpeptidase 2 from *M. tuberculosis* (Ldt_{Mt2}) model (373 aa, or amino acid residues) the solvate box size will be around 66 K atoms, for penicillin acylase from *E. coli* (PaEc) model (764 aa) the solvate box size will be around 111 K atoms, for 4 subunit neuraminidase from Influenza A virus subtype H₂N₂ (NaIAv) model (1876 aa) the solvate box size will be around 178 K atoms. The latter one is especially big system to run molecular dynamic without any acceleration hardware.

To estimate $\Delta \Delta_{\text{alc}} G$ for each of the described systems, it is necessary to calculate the molecular dynamic trajectory for all λ in the course of transformations. Let us take the minimum number of λ windows equal to 9, which are used in some cases in this work. In addition, to check the reproducibility of the results it is necessary to perform at least three repetitions (runs) of the same calculations. Thus, according to the most conservative estimate, in order to reliably calculate one $\Delta \Delta_{\text{alc}} G$ value, it is necessary to calculate 27 molecular dynamics trajectories for the *ligands* system and 27 trajectories for the *complex* system. In case if $\partial U / \partial \lambda$ converges on a time between 10 and 50 nanoseconds (ns) in every MD trajectory, one could see that from 270 to 1350 ns in total of MD trajectories for both *complex* and *ligands* systems are needed to calculate in acceptable period of time. The number of available to compute ns of MD trajectories per day is given in Tab. 1 for a few desktop-class GPUs in comparison to 27x Tesla P100 [22].

Obviously, only with parallel computing on multiple HPC-class GPUs it is possible to achieve the required calculation speed and obtain one $\Delta \Delta_{\text{alc}} G$ value per day, which is necessary if the

Table 1. Comparison of speed in **ns/day** of MD TI calculations for 3 systems on desktop and HPC GPU

system and size	GTX 980	RTX 2070 Super	RTX 3080	27x Tesla P100 [22]
Ldt _{Mt2} (66 K atoms)	24.3	49.1	87.4	1360.8
PaEc (111 K atoms)	12.6	26.1	45.5	—
NaIAv (178 K atoms)	—	15.7	27.9	537.8

task requires to estimate dozens of $\Delta\Delta_{\text{alc}}G$ values for potential analogues of inhibitors and their one-to-one transformations.

3. Systems Preparation and Simulation Protocol

The full atomic structure of penicillin acylase from *E. coli* (PaEc) in a complex with oxidised penicillin G (PDB 1GM9) was used as a starting structure for the construction of enzyme-inhibitor complexes of PaEc with *R*-mandelic acid, *S*-mandelic acids and phenylacetic acid.

The full atomic four subunit model of neuraminidase from Influenza A virus subtype H₂N₂ in a complex with inhibitors was built from a complex with zanamivir (PDB 3TIC).

The three-domain structure of L,D-transpeptidase 2 from *M. tuberculosis* without a signal peptide was assembled from two crystal structures: for domains A and B (PDB 4HU2), and B and C (PDB 4HUC). The resulting structure was equilibrated by classical MD followed by accelerated MD to obtain open conformations of a flexible lid covering the active site. MD frames with the open lid were used to construct enzyme-inhibitor complexes by molecular docking.

Partial charges on inhibitors atoms were obtained by geometry optimization by the B3LYP/6-31+G* method followed by calculation of the electrostatic potential by the HF/6-31G* method and charge fitting by the RESP method [1]. The types of inhibitors atoms were defined using GAFF [25] or GAFF2 force fields, enzymes were described using ff14SB [14] or ff19SB [20]. TIP3P or OPC [8] water models were used.

MD parameters of simulations were as follows: $\Delta t = 1$ fs, Langevin thermostat with collision frequency $\gamma = 2$ ps⁻¹, $T = 298$ K, Monte Carlo barostat with pressure relaxation time $\tau = 2$ ps, $P = 1.01325$ bar, SHAKE was off, NPT ensemble, nonbonded cutoff for PME was 8.0 Å, for TI calculations softcore potentials were applied for Lennard-Jones and Coulomb potentials with $\alpha = 0.5$ and $\beta = 12.0$ Å² respectively. The Amber package was used to simulate MD trajectories [4].

The stresses in the constructed systems were eliminated by minimizing the energy using the steepest descent algorithm. The systems were heated to 298 K on NVT ensemble, equilibrated on NPT ensemble until the nominal values of water density were reached and 5 ns after that. Separate trajectories were created for each value of λ using the last frame of the previous trajectory. For each value of λ the velocities obtained at the previous step were not taken into account, but the systems were heated again at a given value of λ , then equilibrated again using NPT ensemble for the first 10% of the trajectory and during the last 90% of time, values of $\partial U/\partial \lambda$ were collected. The convergence of $\partial U/\partial \lambda$ values was monitored by the values of the moving average and the incremental average. The obtained values of $\partial U/\partial \lambda$ for all λ windows were integrated by the appropriate method, depending on the partition of the transformation path (trapezoidal or Gaussian).

4. Method Validation

4.1. Inhibitors of Penicillin Acylase from *E. coli* (PaEc)

The penicillin acylase is widely used in industry to catalyze an acyl transfer reaction from the activated acyl donors to the amino group of the penicillin nucleus 6-aminopenicillanic acid. Phenylacetic acid and its derivatives are also PaEc inhibitors. The inhibition constants for phenylacetic, *R*-mandelic and *S*-mandelic acids are 0.06 mM, 31 mM and 10 mM, respectively, which means (according to $\Delta\Delta_bG = -RT \ln(K_i^1/K_i^2)$) $\Delta\Delta_bG_{\text{exp}} = -3.7 \text{ kcal/mol}$ for transformation from *R*-mandelic acid to phenylacetic acid and $\Delta\Delta_bG_{\text{exp}} = -3.0 \text{ kcal/mol}$ for transformation from *S*-mandelic acid to phenylacetic acid.

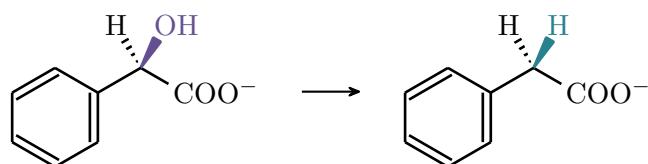


Figure 4. A scheme of alchemical transformation from *R*-mandelic acid (left) to phenylacetic acid (right). Colored atoms are those to which the soft-core potentials were applied

Firstly, for transformation from *R*-mandelic acid to phenylacetic acid (Fig. 4) we decided to use default force fields parameters for TI molecular dynamics (ff14SB, GAFF, TIP3P water) and 13 λ windows (0.0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 1.0). One could see we add 2 additional λ values on the sides of the transformation path. Calculated values of $\Delta\Delta_{\text{alc}}G$ are given in Tab. 2. After five runs of calculation, the estimated interval $\Delta\Delta_{\text{alc}}G = -1.0 \pm 0.4 \text{ kcal/mol}$ does not contain the experimental value $\Delta\Delta_bG_{\text{exp}} = -3.7 \text{ kcal/mol}$. At the same time, the computed value agrees with the experimental trend, but the absolute error is 2.7 kcal/mol.

When analyzing the $\partial U/\partial \lambda$ of λ graphs, we found that the maximum change of $\partial U/\partial \lambda$ values occurs in the interval between 0 and 0.1 in λ scale. Afterwards, to sample this interval more rigorously and improve the estimation of $\Delta\Delta_{\text{alc}}G$, we introduced 5 additional λ windows (0.01, 0.02, 0.03, 0.04 and 0.075), their total number was increased to 18. Having performed similar calculations, we obtained a new estimate of $\Delta\Delta_{\text{alc}}G = -1.3 \pm 0.2 \text{ kcal/mol}$. This estimate is better in terms of reproducibility and closer to the experimental value of $\Delta\Delta_bG_{\text{exp}}$ but still differs from the experimental value by 2.4 kcal/mol.

After several trials, we decided to choose more modern force fields: ff19SB for the enzyme and GAFF2 for the inhibitors, relatively new OPC water model that is recommended [19] to be used with ff19SB was applied. Under new conditions we estimate $\Delta\Delta_{\text{alc}}G = -2.6 \pm 0.7 \text{ kcal/mol}$ which is better than previous result and differ from the experimental value by 1.1 kcal/mol.

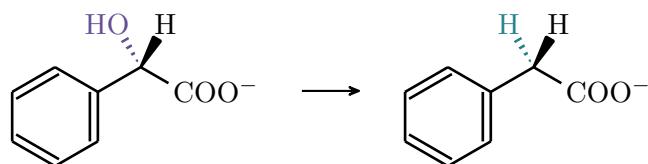


Figure 5. A scheme of alchemical transformation from *S*-mandelic acid (left) to phenylacetic acid (right). Colored atoms are those to which the soft-core potentials were applied

After applying a similar approach to modeling the conversion of *S*-mandelic acid to phenylacetic acid (Fig. 5) using only 13 λ windows we get estimate of $\Delta\Delta_{\text{alc}}G = -3.7 \pm 1.2$ kcal/mol, while experimental value is -3.0 kcal/mol. Calculated values of $\Delta\Delta_{\text{alc}}G$ are given in Tab. 3.

Table 2. Calculated thermodynamic integrals of alchemical transformation for a pair of ***R*-mandelic acid** and **phenylacetic acid** in the complex with PaEc ($\Delta_{\text{alc}}G_{\text{com}}$) and in the bulk ($\Delta_{\text{alc}}G_{\text{lig}}$) for different modeling parameters. All numerical values are given in **kcal/mol**

run	$\Delta_{\text{alc}}G_{\text{com}}$	$\Delta_{\text{alc}}G_{\text{lig}}$	$\Delta\Delta_{\text{alc}}G$	$\overline{\Delta\Delta_{\text{alc}}G} \pm \text{CI}_{95\%}$	$\Delta\Delta_bG_{\text{exp}}$
ff14SB, GAFF, TIP3P, 13 λ windows					
1	-73.71	-72.73	-0.98		
2	-73.88	-72.74	-1.14		
3	-74.10	-72.77	-1.33	-1.0 ± 0.4	-3.7
4	-73.56	-72.75	-0.81		
5	-73.32	-72.75	-0.57		
ff14SB, GAFF, TIP3P, 18 λ windows					
1	-74.21	-72.88	-1.33		
2	-74.13	-72.89	-1.24	-1.3 ± 0.2	-3.7
3	-74.30	-72.90	-1.40		
ff19SB, GAFF 2, OPC, 13 λ windows					
1	-76.13	-73.32	-2.81		
2	-76.17	-73.56	-2.61	-2.6 ± 0.7	-3.7
3	-76.93	-74.69	-2.24		

Table 3. Calculated thermodynamic integrals of alchemical transformation for a pair of ***S*-mandelic acid** and **phenylacetic acid** in the complex with PaEc ($\Delta_{\text{alc}}G_{\text{com}}$) and in the bulk ($\Delta_{\text{alc}}G_{\text{lig}}$). Parameters of modelling: ff19SB, GAFF2, OPC, 13 λ windows. All numerical values are given in **kcal/mol**

run	$\Delta_{\text{alc}}G_{\text{com}}$	$\Delta_{\text{alc}}G_{\text{lig}}$	$\Delta\Delta_{\text{alc}}G$	$\overline{\Delta\Delta_{\text{alc}}G} \pm \text{CI}_{95\%}$	$\Delta\Delta_bG_{\text{exp}}$
1	-78.56	-74.38	-4.18		
2	-78.27	-74.70	-3.57	-3.7 ± 1.2	-3.0
3	-78.42	-75.22	-3.20		

As a conclusion for penicillin acylase inhibitors we found out that ff19SB+GAFF2+OPC performs better in comparison with ff14SB+GAFF+TIP3P.

4.2. Inhibitors of Neuraminidase from Influenza A virus (NaIAv)

The neuraminidase enzyme of influenza virus cleaves of sialic acid from human receptors to facilitate the release of newly formed viral particles from the cell surface and is a molecular target for the most effective anti-influenza drugs zanamivir and oseltamivir [11, 23, 24]. Several sialic acid analogue inhibitors of the enzyme have been synthesized and characterized experimentally [24].

Let us look at the pair 4-amino-Neu5Ac2en and 4-guanidino-Neu5Ac2en (zanamivir) (Fig. 6), the inhibition constants K_i are 50.0 nM and 0.2 nM respectively [24], which means, $\Delta\Delta_bG_{\text{exp}} = -3.3$ kcal/mol for the transformation of 4-amino-Neu5Ac2en to 4-guanidino-Neu5Ac2en.

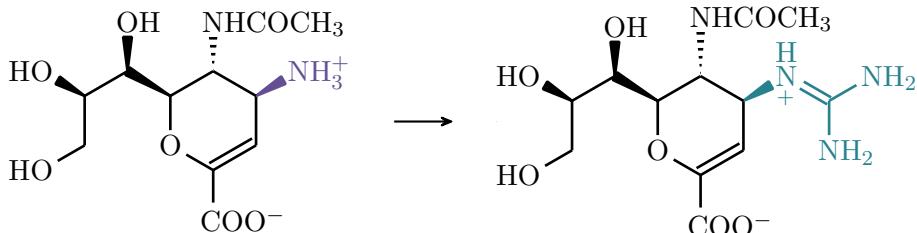


Figure 6. A scheme of alchemical transformation from **4-amino-Neu5Ac2en** (left) to **4-guanidino-Neu5Ac2en**, or **zanamivir**, (right). Colored atoms are those to which the soft-core potentials were applied

In this case, we used ff14SB+GAFF+TIP3P to describe the system. We split the transformation path into 9 λ windows (0.01592, 0.08198, 0.19331, 0.33787, 0.5, 0.66213, 0.80669, 0.91802, 0.98408) according to the n -point Gaussian quadrature ($n = 9$), then Gauss integration method was applied to estimate $\Delta\Delta_{\text{alc}}G$. As can be seen in Tab. 4, after three runs of the calculations, $\Delta\Delta_{\text{alc}}G = -3.3 \pm 0.7$ kcal/mol which is very close to the experimental (-3.3 kcal/mol) value calculated from known K_i values.

Table 4. Calculated thermodynamic integrals of alchemical transformation for a pair of **4-amino-Neu5Ac2en** and **4-guanidino-Neu5Ac2en** (zanamivir) in the complex with NaIAv ($\Delta_{\text{alc}}G_{\text{com}}$) and in the bulk ($\Delta_{\text{alc}}G_{\text{lig}}$). Parameters of modelling: ff14SB, GAFF, TIP3P, 9 λ windows. All numerical values are given in **kcal/mol**

run	$\Delta_{\text{alc}}G_{\text{com}}$	$\Delta_{\text{alc}}G_{\text{lig}}$	$\Delta\Delta_{\text{alc}}G$	$\overline{\Delta\Delta_{\text{alc}}G} \pm \text{CI}_{95\%}$	$\Delta\Delta_bG_{\text{exp}}$
1	-8.35	-4.94	-3.41		
2	-8.07	-4.98	-3.09	-3.3 ± 0.7	-3.3
3	-8.65	-5.06	-3.59		

4.3. Inhibitors of L,D-transpeptidase from *M. tuberculosis* (Ldt_{Mt2})

The L,D-transpeptidase 2 from *M. tuberculosis* is an enzyme catalyzing cross-linking of peptidoglycan chains in the pathogenic cell wall [5]. This protein is a potential target for the new class of antitubercular medications (anti-TB) due to high importance for the growth of the pathogenic bacteria. Well-known β -lactams do not provide efficient anti-TB effect [16, 17], so an active search for new treatments is underway: new combinations of known anti-TB drugs are being studied [15] and new molecules are being sought [13].

Competitive inhibitors of L,D-transpeptidase 2 from *M. tuberculosis* are considered as perspective anti-TB preparations [3]. We have found a new representative by *in silico* screening of commercial ligand libraries. The selected compound, inhA (Fig. 7) was then experimentally studied, characterized as Ldt_{Mt2} inhibitor and used as a molecule at computational alchemy optimization. The resulting improved structure inhB of the inhibitor is shown in Fig. 7 and the data

of the corresponding energy calculations are presented in Tab. 5. The computed value of relative binding energy ($\Delta\Delta_{\text{alc}}G = -1.6 \pm 0.6$ kcal/mol) shows an appropriate fit with experimental data, which has confirmed more effective inhibition of Ldt_{Mt2} by the designed compound inhB ($K_i = 5$ μM) compared to the parent structure inhA ($K_i = 14$ μM); the absolute error between calculated and experimental relative free energy of binding ($\Delta\Delta_bG = -RT \ln(K_i^{\text{inhA}}/K_i^{\text{inhB}})$ = -0.6 kcal/mol) is in order of 1 kcal/mol.

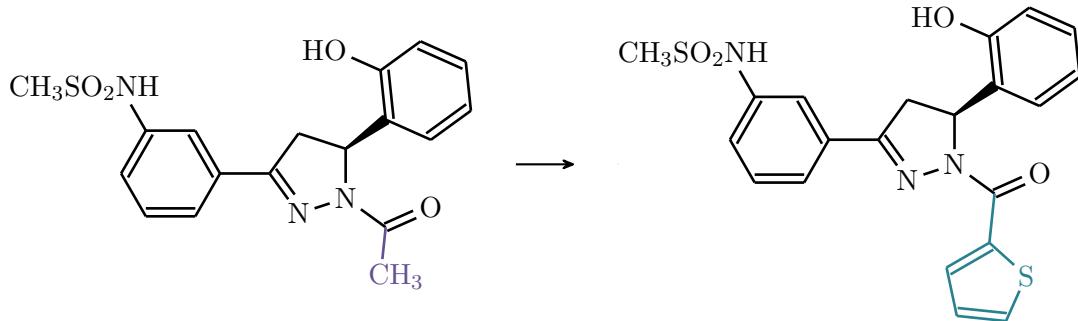


Figure 7. A scheme of alchemical transformation from **inhA** (left) to **inhB** (right). Colored atoms are those to which the soft-core potentials were applied

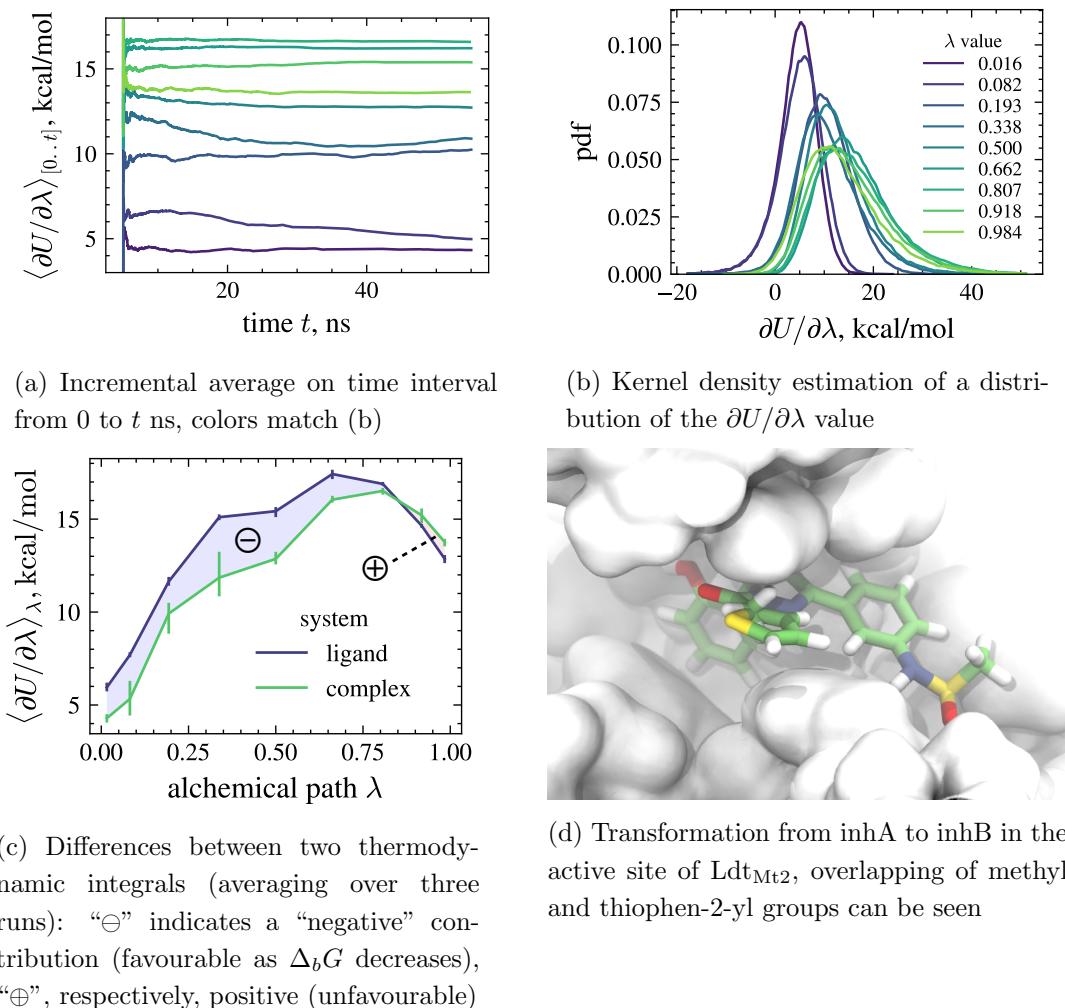


Figure 8. Visualization of $\partial U / \partial \lambda$ during the alchemical transformation of found and tested competitive inhibitors inhA and inhB of Ldt_{Mt2}

Table 5. Calculated thermodynamic integrals of alchemical transformation for a pair **inhA**, **inhB** in the complex with Ldt_{Mt2} ($\Delta_{\text{alc}}G_{\text{com}}$) and in the bulk ($\Delta_{\text{alc}}G_{\text{lig}}$). Parameters of modelling: ff14SB, GAFF, TIP3P, 9 λ windows. All numerical values are given in kcal/mol

run	$\Delta_{\text{alc}}G_{\text{com}}$	$\Delta_{\text{alc}}G_{\text{lig}}$	$\Delta\Delta_{\text{alc}}G$	$\overline{\Delta\Delta_{\text{alc}}G} \pm \text{CI}_{95\%}$	$\Delta\Delta_bG_{\text{exp}}$
1	12.69	14.06	-1.37		
2	12.36	14.18	-1.81	-1.6 ± 0.6	-0.6
3	12.48	14.15	-1.67		

The parameters of computational experiment were as follows: nine λ values were chosen in an interval from 0 to 1 (according to 9-point Gaussian quadrature integration), at least 50 ns of MD trajectories were recorded for all 9 λ on three runs, i.e. [50 ns × 9 λ MDs] × 3 runs.

Let us take a closer look at how we processed the raw calculated data to obtain estimates of the relative binding free energy. Values of $\partial U / \partial \lambda$ obtained during the simulations were treated to compute two TIs (Gaussian quadrature integration was used) and then $\Delta\Delta_{\text{alc}}G$. Using an example of one run (9 MDs for each λ value) we can briefly describe the results (Fig. 8). Estimates of the averages $\langle \partial U / \partial \lambda \rangle_{[0..t]}$ reach a constant value in the first 20–30 ns of MD trajectories (Fig. 8a), all observations after 20 ns were used to get the estimation (for all runs separately). Values of $\partial U / \partial \lambda$ are distributed around their means, forming symmetric Gaussian-like “bell” curves (Fig. 8b). Since the difference of two TIs is required to calculate relative energies $\Delta\Delta_{\text{alc}}G$ the “negative” and “positive” areas are shown in Fig. 8c, it can be seen that the larger in absolute value term corresponds to a decrease in the free energy of binding.

The computational alchemy allows to estimate relative binding energies (21) which makes it useful for structure optimization. It should be noted, that a straightforward transformation with interpretable results can only be performed between close structural analogues (see subsection 1.2) which were found by methods of chemoinformatics (fingerprints, maximum common subgraph). Thus, we can conclude that the method can be used to optimize the structure of inhibitors.

Conclusion

The computational alchemy is based on equilibrium methods of molecular dynamics (MD), including thermodynamic integration (TI) or free energy perturbation (FEP). It makes it possible to evaluate the energy characteristics of a subtle alchemical transformation of the prototype compound structures in order to modulate their functional properties, what can be used searching for new analogues of enzyme inhibitors. In recent years, the method is gaining new opportunities due to high-performance computing as different MD engines got the ability to perform TI or FEP calculations using GPU acceleration. We have tested the method on two experimentally studied enzyme-inhibitor systems: penicillin acylase from *E. coli* (PaEc) and Influenza A virus neuraminidase (NaIAv) and performed computational alchemy calculations for pairs of their known inhibitors. Thus, the validated methodology was then applied to estimate binding of L,D-transpeptidase 2 from *M. tuberculosis* (Ldt_{Mt2}) to the structural analogs of inhibitors previously discovered in our group. We have found that the performance of computational alchemy predictions is highly dependent on properly chosen simulation parameters.

In our case, various combinations of force fields and water models (ff14SB+GAFF+TIP3P and ff19SB+GAFF2+OPC) were used, and at the optimal performance, the difference between the calculated and experimentally determined relative binding energies was less than 1 kcal/mol. Thus, for optimal application to novel enzyme—inhibitor or receptor—ligand systems, we recommend prior adaptation of this method, based on validation of force field combinations, to experimentally determined binding data of structurally related inhibitors/ligands to the molecular target of interest, if available, and then to proceed to the search for new compounds. High-performance computational alchemy can be used as a useful integrative part of the methodology searching for new enzyme inhibitors or receptor ligands and optimizing their structure at drug design.

Acknowledgements

The work was supported by the Russian Science Foundation (grant no. 21-71-30003).

The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Bayly, C., Cieplak, P., Cornell, W., Kollman, P.: A well-behaved electrostatic potential based method using charge restraints for deriving atomic charges: the RESP model. *J Phys Chem* 97, 10269–10280 (1993). <https://doi.org/10.1021/j100142a004>
2. Beutler, T.C., Mark, A.E., van Schaik, R.C., *et al.*: Avoiding singularities and numerical instabilities in free energy calculations based on molecular simulations. *Chemical Physics Letters* 222(6), 529–539 (Jun 1994). [https://doi.org/10.1016/0009-2614\(94\)00397-1](https://doi.org/10.1016/0009-2614(94)00397-1)
3. Billones, J.B., Carrillo, M.C.O., Organo, V.G., *et al.*: Toward antituberculosis drugs: In silico screening of synthetic compounds against *Mycobacterium tuberculosis* l,d-transpeptidase 2. Drug design, development and therapy 10, 1147–1157 (2016). <https://doi.org/10.2147/DDDT.S97043>
4. Case, D.A., Cheatham III, T.E., Darden, T., *et al.*: The amber biomolecular simulation programs. *Journal of computational chemistry* 26(16), 1668–1688 (2005). <https://doi.org/10.1002/jcc.20290>
5. Cordillot, M., Dubée, V., Triboulet, S., *et al.*: In vitro cross-linking of *Mycobacterium tuberculosis* peptidoglycan by L,D-transpeptidases and inactivation of these enzymes by carbapenems. *Antimicrobial agents and chemotherapy* 57(12), 5940–5945 (Dec 2013). <https://doi.org/10.1128/AAC.01663-13>
6. Cornell, W.D., Cieplak, P., Bayly, C.I., *et al.*: A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *Journal of the American Chemical Society* 117(19), 5179–5197 (May 1995). <https://doi.org/10.1021/ja00124a002>

7. He, X., Liu, S., Lee, T.S., *et al.*: Fast, accurate, and reliable protocols for routine calculations of protein–ligand binding affinities in drug design projects using AMBER GPU-TI with ff14SB/GAFF. *ACS omega* 5(9), 4611–4619 (2020). <https://doi.org/10.1021/acsomega.9b04233>
8. Izadi, S., Anandakrishnan, R., Onufriev, A.V.: Building water models: a different approach. *The journal of physical chemistry letters* 5(21), 3863–3871 (2014). <https://doi.org/10.1021/jz501780a>
9. Lee, T.S., Cerutti, D.S., Mermelstein, D., *et al.*: GPU-accelerated molecular dynamics and free energy methods in Amber18: performance enhancements and new features. *Journal of chemical information and modeling* 58(10), 2043–2050 (2018). <https://doi.org/10.1021/acs.jcim.8b00462>
10. Lee, T.S., Hu, Y., Sherborne, B., *et al.*: Toward fast and accurate binding affinity prediction with pmemdGTI: an efficient implementation of GPU-accelerated thermodynamic integration. *Journal of chemical theory and computation* 13(7), 3077–3084 (2017). <https://doi.org/10.1021/acs.jctc.7b00102>
11. Lehnert, R., Pletz, M., Reuss, A., Schaberg, T.: Antiviral medications in seasonal and pandemic influenza: A systematic review. *Deutsches Ärzteblatt International* 113(47), 799–807 (2016). <https://doi.org/10.3238/arztebl.2016.0799>
12. Lin, Y.H., Wessén, J., Pal, T., *et al.*: Numerical Techniques for Applications of Analytical Theories to Sequence-Dependent Phase Separations of Intrinsically Disordered Proteins. In: *Phase-Separated Biomolecular Condensates: Methods and Protocols*, pp. 51–94. Springer US, New York, NY (2023). https://doi.org/10.1007/978-1-0716-2663-4_3
13. Ma, Z., Lienhardt, C., McIllemon, H., *et al.*: Global tuberculosis drug development pipeline: The need and the reality. *The lancet* 375(9731), 2100–2109 (2010). [https://doi.org/10.1016/S0140-6736\(10\)60359-9](https://doi.org/10.1016/S0140-6736(10)60359-9)
14. Maier, J.A., Martinez, C., Kasavajhala, K., *et al.*: ff14SB: improving the accuracy of protein side chain and backbone parameters from ff99SB. *Journal of chemical theory and computation* 11(8), 3696–3713 (2015). <https://doi.org/10.1021/acs.jctc.5b00255>
15. Martelli, G., Pessatti, T.B., Steiner, E.M., *et al.*: N-Thio- β -lactams targeting L,D-transpeptidase-2, with activity against drug-resistant strains of *Mycobacterium tuberculosis*. *Cell Chemical Biology* 28(9), 1321–1332.e5 (2021). <https://doi.org/10.1016/j.chembiol.2021.03.008>
16. Rullas, J., Dhar, N., McKinney, J.D., *et al.*: Combinations of β -Lactam Antibiotics Currently in Clinical Trials Are Efficacious in a DHP-I-Deficient Mouse Model of Tuberculosis Infection. *Antimicrobial agents and chemotherapy* 59(8), 4997–4999 (Aug 2015). <https://doi.org/10.1128/AAC.01063-15>
17. Solapure, S., Dinesh, N., Shandil, R., *et al.*: In vitro and in vivo efficacy of β -lactams against replicating and slowly growing/nonreplicating *Mycobacterium tuberculosis*. *Antimicrobial agents and chemotherapy* 57(6), 2506–2510 (Jun 2013). <https://doi.org/10.1128/AAC.00023-13>

18. Steinbrecher, T., Mobley, D.L., Case, D.A.: Nonlinear scaling schemes for Lennard-Jones interactions in free energy calculations. *The Journal of chemical physics* 127(21), 214108 (2007). <https://doi.org/10.1063/1.2799191>
19. Tian, C., Kasavajhala, K., Belfon, K.A.A., *et al.*: ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. *Journal of Chemical Theory and Computation* 16(1), 528–552 (Jan 2020). <https://doi.org/10.1021/acs.jctc.9b00591>
20. Tian, C., Kasavajhala, K., Belfon, K.A., *et al.*: ff19SB: Amino-acid-specific protein backbone parameters trained against quantum mechanics energy surfaces in solution. *Journal of chemical theory and computation* 16(1), 528–552 (2019). <https://doi.org/10.1021/acs.jctc.9b00591>
21. Tuckerman, M.E., Martyna, G.J.: Understanding Modern Molecular Dynamics: Techniques and Applications. *The Journal of Physical Chemistry B* 104(2), 159–178 (Jan 2000). <https://doi.org/10.1021/jp992433y>
22. 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>
23. Von Itzstein, M.: The war against influenza: discovery and development of sialidase inhibitors. *Nature reviews Drug discovery* 6(12), 967–974 (2007). <https://doi.org/10.1038/nrd2400>
24. Von Itzstein, M., Wu, W.Y., Kok, G.B., *et al.*: Rational design of potent sialidase-based inhibitors of influenza virus replication. *Nature* 363(6428), 418–423 (1993). <https://doi.org/10.1038/363418a0>
25. Wang, J., Wolf, R.M., Caldwell, J.W., *et al.*: Development and testing of a general amber force field. *Journal of Computational Chemistry* 25(9), 1157–1174 (Jul 2004). <https://doi.org/10.1002/jcc.20035>
26. Zwanzig, R.W.: High-Temperature Equation of State by a Perturbation Method. I. Nonpolar Gases. *The Journal of Chemical Physics* 22(8), 1420–1426 (Aug 1954). <https://doi.org/10.1063/1.1740409>