

Supercomputing Frontiers and Innovations

2022, Vol. 9, No. 1

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

4D Technology of Variational Data Assimilation for Sea Dynamics Problems V.P. Shutyaev, V.I. Agoshkov, V.B. Zalesny, E.I. Parmuzin, N.B. Zakharova	4
A Supercomputer-Based Modeling System for Short-Term Prediction of Urban Surface Air Quality A.V. Starchenko, E.A. Danilkin, S.A. Prokhanov, L.I. Kizhner, E.A. Shelmina	17
River Routing in the INM RAS-MSU Land Surface Model: Numerical Scheme and Parallel Implementation on Hybrid Supercomputers V.M. Stepanenko	32
Machine Learning Approaches to Extreme Weather Events Forecast in Urban Areas: Challenges and Initial Results F. Porto, M. Ferro, E. Ogasawara, T. Moeda, C.D.T. Barros, A. Chaves Silva, R. Zorrilla, R.S. Pereira, R. Castro, J.V. Silva, R. Salles, A. Fonseca, J. Hermsdorff, M. Magalhães, V. Sã, A.A. Simões, C. Cardoso, E. Bezerra	49
Data Assimilation by Neural Network for Ocean Circulation: Parallel Implementation H.F. Campos Velho, H.C.M. Furtado, S.B.M. Sambatti, C.O.F. Barros, M.E.S. Welter, R.P. Souto, D. Carvalho, D.O. Cardoso	74
Multistage Iterative Method to Tackle Inverse Problems of Wave Tomography A.V. Goncharsky, S.Y. Romanov, S.Y. Seryozhnikov	87



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.

4D Technology of Variational Data Assimilation for Sea Dynamics Problems

Victor P. Shutyaev^{1,3} , **Valery I. Agoshkov^{1,2}** , **Vladimir B. Zalesny¹** ,
Eugene I. Parmuzin^{1,3} , **Natalia B. Zakharova¹** 

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

The technology aimed at high-performance computing is presented for modeling the sea dynamics problems based on 4D variational data assimilation technique developed at the Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences (INM RAS). The technology is based on the multicomponent splitting method for the mathematical model of sea dynamics and the minimization of cost functionals related to the observation data by solving an optimality system that involves the adjoint equations with observation data and observation error covariances. Efficient algorithms for solving the variational data assimilation problems are presented based on modern iterative processes with a special choice of iterative parameters. The technology is illustrated for the Baltic Sea dynamics model with variational data assimilation to restore the initial states and the heat fluxes on the sea surface.

Keywords: *sea dynamics modeling, variational data assimilation, observations, sea surface temperature.*

Introduction

In recent years, there has been an increasing interest in research methods and numerical solution of inverse and data assimilation problems, which play a fundamental role in the theoretical understanding and mathematical modeling of processes and phenomena from various fields of knowledge. The data assimilation technique is widely used in geosciences to develop high-performance computational technologies that combine the flows of real data and hydrodynamic forecasts using mathematical models. It received the greatest applications in meteorology and oceanography, where observations of the atmosphere and ocean are assimilated into atmospheric and oceanic models in order to obtain the initial and/or boundary conditions (and other model parameters) for further modeling and forecasting [1, 7, 8, 12, 15, 16, 18, 22, 25].

A significant progress in solving data assimilation problems has been the use of variational methods and, in particular, optimal control methods. The development of this direction at the INM RAS was initiated by Academician Guriy I. Marchuk [18]. These approaches were the main content of research of G.I. Marchuk and his scientific school in various fields of mathematics and applications [1, 4, 5, 18, 25].

The variational data assimilation allows, on a unified methodological basis, to solve the problems of initializing hydrophysical fields, assessing the sensitivity of a model solution, identifying model parameters, etc. The main idea of the method is to minimize some functional that describes the deviation of the model solution from the observational data, and the minimum of this functional is sought on the model trajectories, in other words, in the subspace of model solutions. The problem is formulated in a four-dimensional space-time domain and requires the solution of a coupled system of direct and adjoint equations in forward and backward time, respectively, which is very complicated from the computational point of view. The problem adjoint

¹Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow, Russian Federation

²Lomonosov Moscow State University, Moscow, Russian Federation

³Moscow Institute of Physics and Technology, Dolgoprudny, Russian Federation

to the original nonlinear problem has a more complex form, and for solving the adjoint problem, it is required to store 4D arrays of the solutions of the direct problem in machine memory.

Ocean general circulation models are very complex systems, which are based on nonlinear differential equations describing the evolution of three-dimensional fields of currents, temperature and salinity, as well as pressure and density [9, 11, 13, 21], and require the development of efficient numerical methods for a long-time integration. This underlines the importance of high-performance computing for such problems. The ocean hydrodynamics INMOM model (INM ocean model) is described by primitive equations in the sigma-coordinate system, which is solved by finite-difference methods [11, 24, 27]. Its numerical implementation is based on the method of splitting according to physical processes and spatial coordinates [17, 27], which allows us to split the complex problem into a number of simpler ones and solve it in time using explicit or implicit schemes.

This paper is based on the talk given at the Lomonosov Moscow State University Seminar “Supercomputer Modelling of the Earth System” (headed by V.A. Sadovnichy) and presents some approaches to solving the problems of variational data assimilation, developed at the INM RAS last year. As an application, a mathematical model of sea dynamics is considered with a block of variational assimilation of data on sea surface temperature taking into account the covariance matrices of observation errors. On the basis of variational assimilation of observational data, algorithms are proposed for solving inverse problems of restoring initial conditions and heat fluxes on the sea surface. This is the novelty of this paper compared to the previous studies [3–6, 25]. The results of numerical experiments for the Baltic Sea area are discussed.

The article is organized as follows. Section 1 is devoted to the the mathematical model of sea dynamics using the splitting method. In Section 2 we give the statement of the variational data assimilation problem and formulate the algorithms for its solution. Section 3 contains the results of numerical experiments for the Baltic Sea water area. The main results are discussed in the Conclusions.

1. Mathematical Model of Sea Dynamics

We consider the system of equations of sea hydrothermodynamics in geographical coordinates under hydrostatics and Boussinesq approximations [2, 19], in the domain D of variables (x, y, z) for $t \in (0, \bar{t})$:

$$\left\{ \begin{array}{l} \frac{d\vec{u}}{dt} + \begin{bmatrix} 0 & -f \\ f & 0 \end{bmatrix} \vec{u} - g\mathbf{grad}\zeta + A_u \vec{u} + (A_k)^2 \vec{u} = \vec{F} - \frac{1}{\rho_0} \mathbf{grad}P_a - \\ \quad - \frac{g}{\rho_0} \mathbf{grad} \int_0^z \rho_1(T, S) dz', \\ \frac{\partial \zeta}{\partial t} - m \frac{\partial}{\partial x} \left(\int_0^H \Theta(z) u dz \right) - m \frac{\partial}{\partial y} \left(\int_0^H \Theta(z) \frac{n}{m} v dz \right) = f_3, \\ \frac{dT}{dt} + (\bar{U}, \mathbf{Grad}) T + A_T T = f_T, \frac{dS}{dt} + (\bar{U}, \mathbf{Grad}) S + A_S S = f_S, \end{array} \right. \quad (1)$$

where $\bar{U} = (u, v, w)$ is the velocity vector, ζ is the sea surface level function, T is the temperature, S is the salinity, $\vec{u} = (u, v)$, $\rho_1(T, S) = \rho_0 \beta_T (T - T^{(0)}) + \rho_0 \beta_S (S - S^{(0)}) + \gamma \rho_0 \beta_{TS} (T, S)$ is the water density, P_a is the atmospheric pressure, $\vec{F} = (F_1, F_2)$ is the forcing, f_T, f_S are the functions of the ‘internal’ sources, $\rho_0 = \text{const} \approx 1$ is the mean density, $T^{(0)}, S^{(0)}$ are the reference

values of temperature and salinity, $\beta_{TS}(T, S)$ is the sum of all other terms of the expansion of the function of state $\rho = \rho(T, S)$, $f_3 \equiv f_3(x, y, t)$ is the function related to the tide-generating forces, $\beta_T, \beta_S, \gamma, g = const$, $A_\varphi \varphi \equiv -\mathbf{Div}(\hat{a}_\varphi \mathbf{Grad} \varphi)$, $m = 1/(r \cos y)$, $n = 1/r$, $r = R - z \approx R$, $\Theta(z) \equiv (R - z)/R \approx 1$, R is the Earth radius.

The operators $A_\varphi \varphi \equiv -\mathbf{Div}(\hat{a}_\varphi \mathbf{Grad} \varphi)$ involve $\hat{a}_\varphi = diag((a_\varphi)_{ii})$, where $(a_\varphi)_{11} = (a_\varphi)_{22} \equiv \mu_\varphi$, $(a_\varphi)_{33} \equiv \nu_\varphi$, and φ may take the values u, v, T, S . We assume that $\mu_u = \mu_v \equiv \mu$, $\nu_u = \nu_v \equiv \nu$, and $\mu, \nu, \mu_T, \mu_S, \nu_T, \nu_S$ are diffusion coefficients that are supposed to be positive bounded functions. The fourth order operator $(A_k)^2$, with A_k taken for $A_\varphi = A_k$, is defined by the matrix $\hat{k} = diag\{k_{ii}\}$ with nonnegative diagonal elements k_{ii} that are viscosity coefficients in respective directions. We consider $f = f(u) = l + mu \sin y \equiv l + f_1(u)$, where $l = l(y)$ is the Coriolis parameter $l = 2\omega \sin y$, and ω is the Earth angular rotation speed.

The boundary $\Gamma \equiv \partial D$ of the domain D is represented as a union of four disjoint parts Γ_S , $\Gamma_{w,op}$, $\Gamma_{w,c}$, and Γ_H , where $\Gamma_S \equiv \Omega$ is the “unperturbed” sea surface, $\Gamma_{w,op}$ is the liquid (open) part of the vertical lateral boundary, $\Gamma_{w,c}$ is the solid part of the vertical lateral boundary, and Γ_H is the sea bottom. The characteristic functions (indicator functions) of the parts Γ_S , $\Gamma_{w,op}$, $\Gamma_{w,c}$, and Γ_H of the boundary Γ are denoted by m_S , $m_{w,op}$, $m_{w,c}$, and m_H , respectively. These functions are equal to 1 on the corresponding parts, otherwise they are equal to zero.

The unit outer normal vector to Γ is denoted by $\vec{N} \equiv (N_1, N_2, N_3)$, with $\vec{N} = (0, 0, -1)$ on Γ_S and $\vec{N} = (N_1, N_2, 0)$ on $\Gamma_w = \Gamma_{w,op} \cup \Gamma_{w,c}$, and $\vec{n} \equiv (n_1, n_2) \equiv (n_1, n_2)$ is the unit outer normal vector to $\partial\Omega$. We assume also that $|N_3| > 0$ on Γ_H . The components N_1, N_2, N_3 are defined by the chosen parametric representation of the corresponding part of the boundary. For the velocity vector $\bar{U} = (u, v, w)$ on the boundary Γ , the normal components are denoted by $U_n : U_n = \bar{U} \cdot \vec{N} = uN_1 + vN_2 + wN_3$. Below we put $U_n^{(+)} \equiv (|U_n| + U_n)/2$, $U_n^{(-)} \equiv (|U_n| - U_n)/2$, with $U_n = U_n^{(+)} - U_n^{(-)}$ on Γ .

The hydrostatics approximation means that $\frac{\partial P}{\partial z} = g\rho$, where P is the pressure, $\rho = \rho_0 + \rho_1$. This equation is used to find P after solving the system (1). Due to this relation, the pressure gradient in (1) is divided into three terms: the gradients of the atmospheric pressure, sea surface elevation, and water column pressure deviation.

We consider the equations (1) in $D \times (0, \bar{t})$ with the following boundary and initial conditions [2].

Boundary conditions on Γ_S :

$$\left\{ \begin{array}{l} \left(\int_0^H \Theta \vec{u} dz \right) \vec{n} = 0 \text{ on } \partial\Omega, \\ U_n^{(-)} u - \nu \frac{\partial u}{\partial z} - k_{33} \frac{\partial}{\partial z} A_k u = \tau_x^{(a)} / \rho_0, \quad U_n^{(-)} v - \nu \frac{\partial v}{\partial z} - k_{33} \frac{\partial}{\partial z} A_k v = \tau_y^{(a)} / \rho_0, \\ A_k u = 0, \quad A_k v = 0, \\ U_n^{(-)} T - \nu_T \frac{\partial T}{\partial z} + \gamma_T (T - T_a) = Q_T, \\ U_n^{(-)} S - \nu_S \frac{\partial S}{\partial z} + \gamma_S (S - S_a) = Q_S, \end{array} \right. \quad (2)$$

where $\tau_x^{(a)}, \tau_y^{(a)}$ are the tangent wind stress components along the axes Ox and Oy , respectively, on the sea surface $z = 0$, γ_T, γ_S are the coefficients of relaxation to the specified values of temperature T_a and salinity S_a , respectively, k_{33} is the vertical viscosity coefficient, ν is the turbulent exchange coefficient, and Q_T, Q_S are the surface heat and salinity fluxes, respectively.

We have also $U_n|_{z=0} = -w|_{z=0}$, where $w = w(u, v)$ is defined by the formula

$$w(x, y, z, t) = \frac{1}{r} \left(m \frac{\partial}{\partial x} \left(\int_z^H r u dz' \right) + m \frac{\partial}{\partial y} \left(\frac{n}{m} \int_z^H r v dz' \right) \right), \quad (x, y, t) \in \Omega \times (0, \bar{t}). \quad (3)$$

Boundary conditions on $\Gamma_{w,c}$ (on the “solid” part lateral wall):

$$U_n = 0, \quad A_k \tilde{U} = 0, \quad \frac{\partial \tilde{U}}{\partial N_u} \cdot \tau_w + \left(\frac{\partial}{\partial N_u} A_k \tilde{U} \right) \cdot \tau_w = 0, \quad \frac{\partial T}{\partial N_T} = 0, \quad \frac{\partial S}{\partial N_S} = 0, \quad (4)$$

where $\tau_w = (-N_2, N_1, 0)$, $\tilde{U} \equiv (u, v, 0) \equiv (\vec{u}, 0)$, $\partial \varphi / \partial N_\varphi \equiv \vec{N} \cdot \hat{a}_\varphi \cdot \mathbf{Grad} \varphi$, $\varphi = u, T, S$.

Boundary conditions on $\Gamma_{w,op}$ (on the “liquid” part lateral wall):

$$\begin{cases} U_n^{(-)}(\tilde{U} \cdot \vec{N}) + \frac{\partial \tilde{U}}{\partial N_u} \cdot \vec{N} = 0, \quad A_k \tilde{U} = 0 \\ U_n^{(-)}(\tilde{U} \cdot \tau_w) + \frac{\partial \tilde{U}}{\partial N_u} \cdot \tau_w + \left(\frac{\partial}{\partial N_u} A_k \tilde{U} \right) \cdot \tau_w = 0, \\ U_n^{(-)}T + \frac{\partial T}{\partial N_T} = Q_T, \quad U_n^{(-)}S + \frac{\partial S}{\partial N_S} = Q_S, \end{cases} \quad (5)$$

where Q_T, Q_S are the heat and salinity fluxes, respectively.

Boundary conditions on Γ_H (on the bottom):

$$\begin{cases} w = um \frac{\partial H}{\partial x} + vn \frac{\partial H}{\partial y}, \quad A_k \tilde{U} = 0, \quad \frac{\partial T}{\partial N_T} = 0, \quad \frac{\partial S}{\partial N_S} = 0, \\ \frac{\partial \tilde{U}}{\partial N_u} \cdot \tau_x + \left(\frac{\partial}{\partial N_u} A_k \tilde{U} \right) \cdot \tau_x = \tau_x^{(b)} / \rho_0, \quad \frac{\partial \tilde{U}}{\partial N_u} \cdot \tau_y + \left(\frac{\partial}{\partial N_u} A_k \tilde{U} \right) \cdot \tau_y = \tau_y^{(b)} / \rho_0, \end{cases} \quad (6)$$

where τ_x, τ_y is the system of unit orthogonal vectors of the coordinate system corresponding to x and y directions; $\tau_x^{(b)}, \tau_y^{(b)}$ are the projections of the bottom friction vector on the axes Ox, Oy , respectively.

Initial conditions for u, v, T, S, ζ :

$$u = u^0, v = v^0, T = T^0, S = S^0, \zeta = \zeta^0, \text{ for } t = 0, \quad (7)$$

where $u^0, v^0, T^0, S^0, \zeta^0$ are the given functions.

The problem of large-scale sea dynamics in terms of the functions u, v, w, ζ, T, S consists in solving the system (1)–(7). If the functions u, v, ζ, T, S are found, then the function w is determined by formula (3).

The main features of the numerical model of sea dynamics INMOM are the simultaneous use of the splitting method [17, 27] and the σ -coordinate system [24, 27] for (1)–(7). These two components are used in tandem to build efficient computer technology for 4DVAR ocean data assimilation.

The transition to the σ -system can be carried out at the stage of considering the original problem (1)–(7) before applying suitable splitting schemes and other numerical procedures [20].

In order to approximate the model (1)–(7) in time, we use the splitting method that allows us to represent the solution of the original nonlinear system by subsequent solutions of simpler problems (steps of the splitting method). Let us introduce the grid on $[0; \bar{t}]$: $0 = t_0 < t_1 < \dots < t_{J-1} < t_J = \bar{t}$, $\Delta t_j = t_j - t_{j-1}$ and consider problem (1)–(7) on (t_{j-1}, t_j) , assuming that the vector of the approximate solution $\phi_k \equiv (u_k, v_k, \xi_k, T_k, S_k)$, $k = 1, 2, \dots, j-1$ at the previous

intervals, is already defined. To approximate the problem, we use one of the schemes of the total approximation method [17], which consists in the implementation of the following steps.

Step 1. Consider the problem

$$T_t + (\bar{U}, \mathbf{Grad}) T - \mathbf{Div}(\hat{a}_T \cdot \mathbf{Grad} T) = f_T \text{ in } D \times (t_{j-1}, t_j) \quad (8)$$

under corresponding boundary and initial conditions.

Step 2. Solve the problem

$$S_t + (\bar{U}, \mathbf{Grad}) S - \mathbf{Div}(\hat{a}_S \cdot \mathbf{Grad} S) = f_S \text{ in } D \times (t_{j-1}, t_j) \quad (9)$$

under appropriate boundary and initial conditions.

Step 3. The system

$$\begin{cases} \underline{u}_t^{(1)} + \begin{bmatrix} 0 & -l \\ l & 0 \end{bmatrix} \underline{u}^{(1)} - g\mathbf{grad}\zeta = \vec{F} - \frac{1}{\rho_0} \mathbf{grad} \left(P_a + g \int_0^z \rho_1(\bar{T}, \bar{S}) dz' \right) \\ \qquad \qquad \qquad \text{in } D \times (t_{j-1}, t_j), \\ \zeta_t - \mathbf{div} \left(\int_0^H \Theta \underline{u}^{(1)} dz \right) = f_3 \text{ in } \Omega \times (t_{j-1}, t_j), \\ \underline{u}^{(1)} = \underline{u}_{j-1}, \zeta = \zeta_{j-1} \text{ for } t = t_{j-1}, \underline{u}_j^{(1)} \equiv \underline{u}^{(1)}(t_j) \text{ in } D \end{cases} \quad (10)$$

is solved under corresponding boundary conditions, and the function $\zeta_j \equiv \zeta^{(1)}$ is taken as an approximation to ζ on (t_{j-1}, t_j) . Then the following problems are solved:

$$\begin{cases} \underline{u}_t^{(2)} + \begin{bmatrix} 0 & -f_1(\bar{u}) \\ f_1(\bar{u}) & 0 \end{bmatrix} \underline{u}^{(2)} = 0 \text{ in } D \times (t_{j-1}, t_j), \\ \underline{u}^{(2)} = \underline{u}_j^{(1)} \text{ for } t = t_{j-1}, \underline{u}_j^{(2)} \equiv \underline{u}^{(2)}(t_j) \text{ in } D, \end{cases} \quad (11)$$

$$\begin{cases} \underline{u}_t^{(3)} + (\bar{U}, \mathbf{Grad}) \underline{u}^{(3)} - \mathbf{Div}(\hat{a}_u \cdot \mathbf{Grad}) \underline{u}^{(3)} + (A_k)^2 \underline{u}^{(3)} = 0 \text{ in } D \times (t_{j-1}, t_j), \\ \underline{u}^{(3)} = \underline{u}^{(2)} \text{ for } t = t_{j-1} \text{ in } D, \end{cases} \quad (12)$$

where $\underline{u}^{(3)} = (u^{(3)}, v^{(3)})$, $\bar{U}^{(3)} = (\underline{u}^{(3)}, w^{(3)}(u^{(3)}, v^{(3)}))$. After solving (12), the vector $\underline{u}^{(3)} \equiv \vec{u}_j \equiv (u_j, v_j)$ is taken as an approximation to the exact vector \vec{u} on $D \times (t_{j-1}, t_j)$, and the approximation $w_j \equiv w(u_j, v_j)$ to the vertical component of the velocity vector is calculated by (3).

It is seen that step 3 consists of 3 substeps, and by the superscripts in parentheses we denote the value of the solution at the corresponding substeps. The underline stands for 2D vectors, and the overline stands for 3D vectors.

Steps 1 and 2 may be also subsplitted, each into several substeps, based on the method of splitting according to spatial coordinates [20, 27]. The differential operator of the three-dimensional transport-diffusion heat and salt problems (8) and (9) is represented as a sum of simpler non-negative operators, which allows to split the problems into a number of simpler ones and solve them in time using explicit or implicit schemes.

When steps 1–3 are implemented, after the first step we get an approximation to T , after the second an approximation to S , and after the third step we get an approximation to $\vec{u} = (u, v)$ and ζ . Therefore, the subproblems at these steps are independent of each other and may be solved in parallel. This is very important for high-performance computing.

2. Variational Data Assimilation Technology

Comprehensive monitoring of the main characteristics of natural environment and climate, which is important both for everyday life and for reducing the consequences of natural and man-made disasters, requires the new effective methods and algorithms for the variational assimilation of remote sensing data in atmospheric, ocean and climate models to be developed for high-performance computing. The purpose is to estimate the unknown model inputs: the initial state of the system, the boundary conditions, the source terms, distributed coefficients, etc. The problems are formulated as optimal control problems (deterministic or stochastic) involving cost functions associated with observations, and the minimization is considered on the trajectories (solutions) of the model under consideration [1, 4, 5, 15, 16, 18, 25].

We will demonstrate the data assimilation technology for the case when in problem (1)–(7) the initial state T^0 and the total heat flux function $Q = -\nu_T \frac{\partial T}{\partial z}$ on Γ_S are unknown and treated as additional “controls”. The cost function is related to observations and has the form:

$$J(T^0, Q) = \frac{\alpha}{2} \int_0^{\bar{t}} \int_{\Omega} |Q - Q^{(0)}|^2 d\Omega dt + \frac{\beta}{2} \int_D |T^0 - T^{(0)}|^2 dD + \frac{1}{2} \sum_{j=1}^J J_{0,j}, \quad (13)$$

$$J_{0,j} \equiv \int_{t_{j-1}}^{t_j} \int_{\Omega} (T|_{z=0} - T_{\text{obs}}) \mathcal{R}^{-1} (T|_{z=0} - T_{\text{obs}}) d\Omega dt,$$

where $Q^{(0)} = Q^{(0)}(x, y, t)$, $T^{(0)} = T^{(0)}(x, y, z)$ are the given functions, T_{obs} is the function of observations on the sea surface Ω , \mathcal{R} is the observation error covariance operator, and $\alpha, \beta = \text{const} > 0$ are the regularization parameters. The functions $Q^{(0)}, T^{(0)}$ are usually chosen as first approximations (so-called “background”) for the unknown Q and T^0 . The aim of variational data assimilation is, using $Q^{(0)}$ and $T^{(0)}$, to find better estimates for Q and T^0 , consistent with the model solution and observations, for further modelling and forecast.

We consider the following variational data assimilation problem: *find a solution to (1)–(7) and functions T^0, Q , such that functional (13) takes the minimum value:*

$$J(T^0, Q) = \inf_{T^0, Q} J(T^0, Q).$$

The gradient of the functional $J(T^0, Q)$ with respect to T^0, Q is defined by the adjoint state T^* as follows:

$$\begin{aligned} J'_Q &= \alpha (Q - Q^{(0)}) + T^* \text{ on } \Omega, \\ J'_{T^0} &= \beta (T^0 - T^{(0)}) + T^*|_{t=0} \text{ in } D. \end{aligned} \quad (14)$$

The necessary optimality condition $J'_Q = J'_{T^0} = 0$ leads to the optimality system, which determines the solution of the formulated problem of variational data assimilation. The optimality system includes the direct problem (1)–(7), the adjoint problem, and the optimality conditions in the form:

$$\begin{aligned} \alpha (Q - Q^{(0)}) + T^* &= 0 \text{ on } \Omega, \\ \beta (T^0 - T^{(0)}) + T^*|_{t=0} &= 0 \text{ in } D. \end{aligned} \quad (15)$$

Equations (14) are obtained by differentiating the cost function (13) with respect to T^0 and Q and using the classical representation of the result through the adjoint problem [18]. The adjoint state T^* is the solution of the adjoint problem, which in the case of applying the splitting method

is determined at Step 1 in the form:

$$\begin{aligned}
 -T^*_{\cdot t} - \mathbf{Div}(\bar{U}T^*) - \mathbf{Div}(\hat{a}_T \cdot \mathbf{Grad} T^*) &= 0 \quad \text{in } D \times (t_{j-1}, t_j), \\
 T^* &= 0 \quad \text{for } t = t_j, \\
 -\nu_T \frac{\partial T^*}{\partial z} &= \mathcal{R}^{-1}(T|_{z=0} - T_{\text{obs}}) \quad \text{on } \Omega.
 \end{aligned} \tag{16}$$

The adjoint problem (16) involves the observation data T_{obs} and the observation error covariance operator \mathcal{R} in the boundary condition on the sea surface.

The optimality system that determines the solution of the formulated problem of variational data assimilation reduces to the sequential solution of the subproblems on $t \in (t_{j-1}, t_j)$, $j = 1, 2, \dots, J$.

To find an approximate solution of the optimality system, with the simultaneous determination of T^0, Q by variational assimilation of T_{obs} we can use the following iterative algorithm. If Q_k is the already constructed approximation to Q on (t_{j-1}, t_j) , and T_k^0 is the approximation to T^0 , then after solving the forward and adjoint problems with $Q \equiv Q_k, T^0 = T_k^0$, the next approximations Q_{k+1}, T_{k+1}^0 are computed by:

$$Q_{k+1} = Q_k - \gamma_k(\alpha(Q_k - Q^{(0)}) + T^*) \quad \text{on } \Omega \times (t_{j-1}, t_j), \tag{17}$$

$$T_{k+1}^0 = T_k^0 - \gamma_k(\beta(T_k^0 - T^{(0)}) + T^*|_{t=0}) \quad \text{on } D \tag{18}$$

with the parameters γ_k chosen so that the iterative process (17)–(18) is convergent [3]. After computing Q_{k+1}, T_{k+1}^0 , the solution of the direct and adjoint problems is repeated with the new approximations Q_{k+1}, T_{k+1}^0 , and then Q_{k+2}, T_{k+2}^0 are calculated, and so on. Iterations are repeated until a suitable convergence criterion is met.

The convergence properties of similar iterative algorithms were studied in previous works of the authors. For example, in the case $\beta = 0$, for γ_k one can take the parameters

$$\gamma_k = \frac{1}{2} \int_{t_{j-1}}^{t_j} \int_{\Omega} (T|_{z=0} - T_{\text{obs}}) \mathcal{R}^{-1} (T|_{z=0} - T_{\text{obs}}) d\Omega dt / \int_{t_{j-1}}^{t_j} \int_{\Omega} (T^*)^2|_{z=0} d\Omega dt$$

which may significantly accelerate the convergence of the iterative process [3].

The formulated algorithm allows us to solve the considered four-dimensional variational data assimilation problem. Each step of the assimilation procedure according to (17)–(18) requires solving the forward and adjoint problems. With the use of the σ -coordinate system, the model solution domain does not depend on time: its horizontal boundaries do not change, and the vertical coordinate changes from zero to unity. This allows using the uniform grid in the vertical direction, which is convenient for numerical implementation. The use of the method of splitting with respect to geometric coordinates makes it possible to numerically solve the subproblems independently of each other. These calculations can be done in parallel, which is important for high-performance computing.

3. Numerical Experiments for the Baltic Sea Water Area

To carry out numerical experiments on the assimilation of satellite observation data on the sea surface temperature, the water area of the Baltic Sea was selected. In all experiments,

the problem of recovering the initial and boundary conditions was considered in one iterative procedure of the form (17)–(18). The model of the dynamics of the Baltic Sea was chosen as the main model [26]. This model uses the method of splitting both in spatial variables and in physical processes, which greatly simplifies the application of the theory of adjoint equations for the formulation and solution of the data assimilation problems. It also allows the use of OpenMP technology for those processes that can be calculated independently of each other.

The second distinctive feature of the model considered is the use of the sigma-coordinate system. The approximation of the model on the “grid C” [26] was used. This model was supplemented by the variational data assimilation procedure described in the previous section. The model was started running with zero initial conditions and run with atmospheric forcing obtained from reanalysis, of about 20 years, and after that the result of calculation was taken as an initial condition for further running of the model. The assimilation procedure worked only during some time windows.

Meteorological characteristics were used to calculate the atmospheric impact in the INMOM model [10], including using the bulk formulas for calculating turbulent flows on the sea surface. The values of the mean climatic heat flow $Q^{(0)}$ calculated in this way were used in the data assimilation procedure as a background. To start the assimilation procedure, the function $T^{(0)}$ was taken as a model forecast for the previous time interval. For other functions in the boundary conditions their climatic values were taken.

The daily mean observations T_{obs} for the experiments were obtained from the Copernicus Marine Service (<https://www.marine.copernicus.eu>). Numerical calculations used the DMI Sea Surface Temperature reprocessed analysis aimed at providing daily gap-free maps of sea surface temperature, at $0.02^\circ \times 0.02^\circ$ horizontal resolution, using satellite data from infra-red radiometers [14]. The data obtained were verified and interpolated on the computational grid of the numerical model [23]. Based on the observational data on the surface temperature, the covariance matrices of data errors \mathcal{R} [6] were constructed, which are used to calculate the cost functional (13) and its gradient in the course of the numerical solution of the problem.

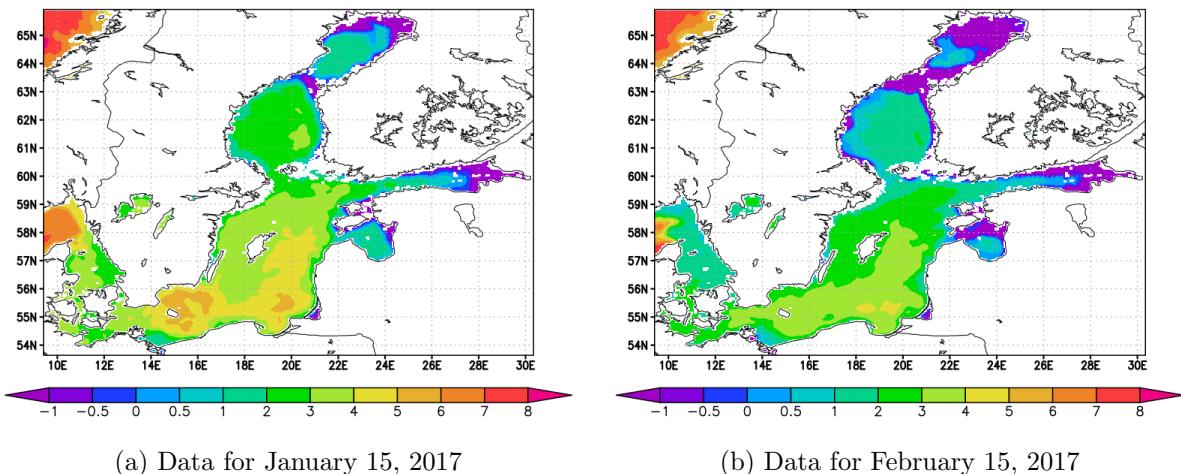


Figure 1. Daily mean SST observation data, $^{\circ}\text{C}$

A number of calculations were made between January and March 2017. The grid step in the model was 3.5 km in space, with 27 vertical levels. The time step in the experiments was

5 minutes. In all experiments, the regularization parameters were chosen the same and equal to $\alpha = \beta = 10^{-5}$.

Let us consider some of the calculation results. Figure 1 shows the daily mean sea surface temperature (SST) fields for January 15 (Fig. 1a) and February 15 (Fig. 1b) obtained from Copernicus Marine Service and used as observational data in numerical experiments. In the model with the assimilation procedure, these data are used 2 times a day to adjust the initial and boundary conditions, i.e. the functions T^0 and Q .

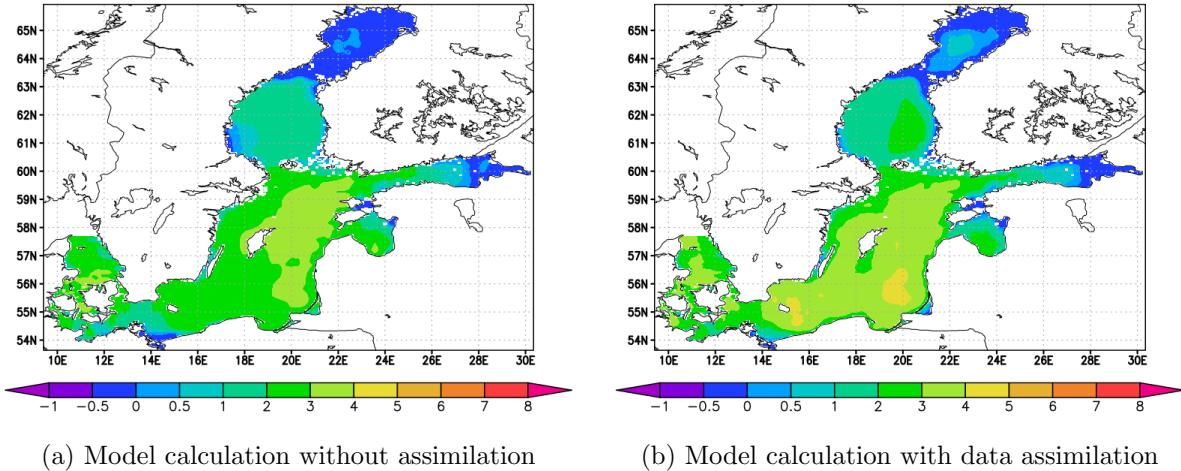


Figure 2. Daily mean SST, January 15, 2017, °C

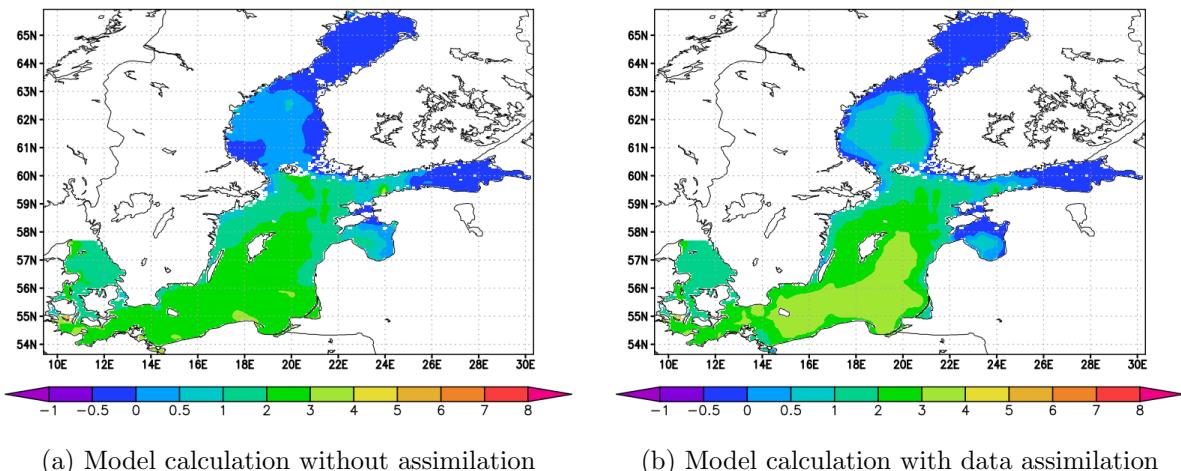


Figure 3. Daily mean SST, February 15, 2017, °C

Figures 2 and 3 show the results of calculations using the model without variational data assimilation (Fig. 2a and 3a) and with temperature assimilation of sea surface data (Fig. 2b and 3b). It can be seen from the presented calculations that the use of the assimilation procedure makes it possible to correct the calculations of the model and bring them closer to the actually observed data. In wintertime, the sea ice block is used in the model [26], however, the assimilation does not use the points in the regions with ice, because we have no observation data at these points. Note that the model without assimilation in the southern part of the Baltic Sea and in the Gulf of Bothnia gives somewhat underestimated SST values, and the deviation from the

observed values may reach 2.5°C . Application of the assimilation procedure allows bringing this deviation to $1\text{--}1.5^{\circ}\text{C}$. It is not possible to completely remove this deviation using only the daily averaged observational data, and it is necessary to use additional data sources for a more reliable correction of the model.

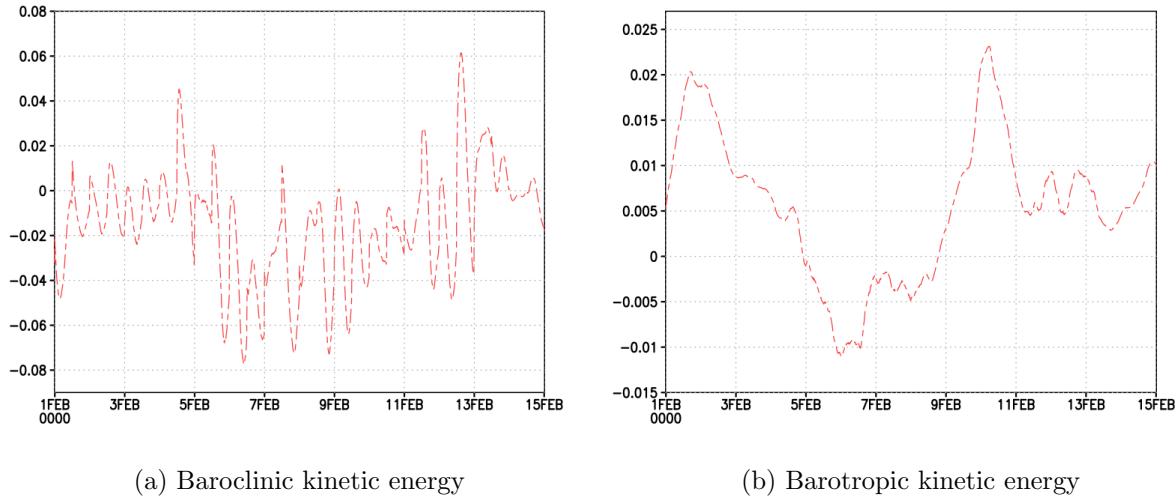


Figure 4. Difference in energies calculated without assimilation and using variational assimilation, cm^2/sec^2

Figure 4 shows the differences in the values of the baroclinic and barotropic kinetic energies of the system as a function of time, obtained from model calculations without using the data assimilation procedure and with using this procedure.

Numerical experiments show that the influence of assimilation on the value of the baroclinic and barotropic energies of the system is insignificant. According to the calculations, the difference in energies when calculated by the model without assimilation and using the assimilation procedure does not exceed 1%. So, the values of the baroclinic kinetic energy vary in the range from $4 \text{ cm}^2/\text{sec}^2$ to $28 \text{ cm}^2/\text{sec}^2$, while the difference in values at calculation according to the model with and without data assimilation lies in the range from -0.08 to $0.06 \text{ cm}^2/\text{sec}^2$ (see Fig. 4a). Similar results are obtained for barotropic kinetic energy. With values from 3 to $17 \text{ cm}^2/\text{sec}^2$, the difference in calculations with and without assimilation varies from -0.012 to $0.023 \text{ cm}^2/\text{sec}^2$ (see Fig. 4a).

From these and many other our numerical experiments it follows that when only the sea surface temperature is assimilated, the values of the velocities change faintly. Nevertheless, all hydrophysical fields obtained in the course of computations using the variational assimilation of observational data remain consistent and physical.

The iterative procedures used for the four-dimensional variational assimilation of the sea surface temperature in the Baltic Sea showed good convergence, and no more than 10 iterations were required to obtain the optimal heat flux Q and the initial state T^0 . In some experiments, the parameters of the iterative process can be calculated based on the features of the system itself, and in this case it is possible to achieve convergence of the process in 3–5 iterations.

Numerical experiments has shown that the inclusion of the data assimilation procedure increases the calculation time by about 10%, which can be reduced by using parallelization. Due to the fact that some procedures of the numerical model use implicit schemes, it is quite difficult to build a full parallel model for the version used for experiments in this work. Nevertheless, where

possible, the procedures were parallelized using the OpenMP methods. A series of calculations has been run to evaluate performance and acceleration when using the OpenMP technology. The Tab. 1 shows some test results calculated for 144 steps of the model.

Table 1. Test results

Number of threads	Computation time, s	Speed-up
1	178.37	1.00
2	118.93	1.49
4	77.31	2.31

When using the OpenMP methods, it was possible to speed up the model calculations by 2.3 times. The assimilation code has also been accelerated using the OpenMP technology. In the assimilation procedure, for all grid nodes in which there are observation data, the same operations are performed, therefore, such nodes were grouped into sets of 32, and the assimilation procedure was rewritten in such a way that each mathematical operation necessary for assimilation was performed in the most nested loop of 32 elements. On a 4-core Intel Core i7-3770K processor with 8 threads, the program code was accelerated by about 4 times due to parallel computations using the OpenMP technology.

Numerical experiments for the Baltic Sea dynamics model confirmed the feasibility of the presented computational technology and demonstrated that the assimilation can improve the predictive properties of the model.

Conclusions

The paper presents the results obtained by the INM RAS researchers on the 4D technology of variational data assimilation for sea dynamics problems, which is based on the development of efficient numerical algorithms for problems of variational assimilation of observation data for a model of marine hydrothermodynamics. Based on the variational assimilation of the observation data, we propose the algorithms for solving inverse problems to restore the heat fluxes on the sea surface and the initial states of the model under consideration. These algorithms have shown their efficiency for the models based on the use of the method of splitting with respect to physical processes and geometric coordinates, which made considered problems easier at each implementation step. The numerical experiments for the Baltic Sea dynamics model have shown the ability to apply the proposed variational assimilation algorithms to modelling hydrothermodynamics problems of marine areas and demonstrated a good proximity of the obtained solutions to real observation data. The reported technology belongs to a class of computational technologies that combine the flows of real data and hydrodynamic forecasts using mathematical models.

Acknowledgements

The work was supported by the Russian Science Foundation (project 20–11–20057, in the part of research of Sections 2–3) and the Moscow Center of Fundamental and Applied Mathematics (agreement with the Ministry of Education and Science of the Russian Federation, No. 075–15–2019–1624).

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. Agoshkov, V.I.: Methods of Optimal Control and Adjoint Equations in Problems of Mathematical Physics. INM RAS, Moscow (2003)
2. Agoshkov, V.I., Gusev, A.V., Diansky, N.A., Oleinikov, R.V.: An algorithm for the solution of the ocean hydrothermodynamics problem with variational assimilation of the sea level function data. Russ. J. Numer. Anal. Math. Model 22(2), 133–161 (2007). <https://doi.org/10.1515/RJNAMM.2007.007>
3. Agoshkov, V.I., Parmuzin, E.I., Shutyaev, V.P.: Numerical algorithm for variational assimilation of sea surface temperature data. Comp. Math. and Math. Physics 48(8), 1293–1312 (2008). <https://doi.org/10.1134/S0965542508080046>
4. Agoshkov, V.I., Parmuzin, E.I., Zalesny, V.B., et al.: Variational assimilation of observation data in the mathematical model of the Baltic Sea dynamics. Russ. J. Numer. Anal. Math. Model 30(4), 203–212 (2015). <https://doi.org/10.1515/rnam-2015-0018>
5. Agoshkov, V.I., Zalesny, V.B., Parmuzin, E.I., et al.: Problems of variational assimilation of observational data for ocean general circulation models and methods for their solution. Izv. Atmos. Ocean. Phys. 46, 677–712 (2010). <https://doi.org/10.1134/S0001433810060034>
6. Agoshkov, V.I., Parmuzin, E.I., Zakharova, N.B., Shutyaev, V.P.: Variational assimilation with covariance matrices of observation data errors for the model of the Baltic Sea dynamics. Russ. J. Numer. Anal. Math. Model 33(3), 149–160 (2018). <https://doi.org/10.1515/rnam-2018-0013>
7. Asch, M., Bocquet, M., Nodet, M.: Data Assimilation: Methods, Algorithms, and Applications. SIAM, Philadelphia (2016). <https://doi.org/10.1137/1.9781611974546>
8. Carrassini, A., Bocquet, M., Bertino, L., Evensen, G.: Data assimilation in the geosciences: an overview of methods, issues, and perspectives. WIREs Clim. Change 9, 1–80 (2018). <https://doi.org/10.1002/wcc.535>
9. Chassignet, E.P., Verron, J.: Ocean Weather Forecasting: An Integrated View of Oceanography. Springer, Heidelberg (2006). <https://doi.org/10.1007/1-4020-4028-8>
10. Diansky, N.A., Fomin, V.V., Zhokhova, N.V., Korshenko, A.N.: Simulations of currents and pollution transport in the coastal waters of Big Sochi. Izv. Atmos. Ocean. Phys. 49(6), 611–621 (2013). <https://doi.org/10.1134/S0001433813060042>
11. Dymnikov, V.P., Zalesny, V.B.: Fundamentals of Computational Geophysical Fluid Dynamics. GEOS, Moscow (2019)
12. Fletcher, S.J.: Data Assimilation for the Geosciences: From Theory to Application. Elsevier, Amsterdam (2017)
13. Griffies, S.M., Boening, C., Bryan, F.O., et al.: Developments in ocean climate modelling. Ocean Model 2, 123–192 (2000). [https://doi.org/10.1016/S1463-5003\(00\)00014-7](https://doi.org/10.1016/S1463-5003(00)00014-7)

14. Hyer, J.L., Karagali, I.: Sea surface temperature climate data record for the North Sea and Baltic Sea. *Journal of Climate* 29(7), 2529–2541 (2016). <https://doi.org/10.1175/JCLI-D-15-0663.1>
15. Le Dimet, F., Talagrand, O.: Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus A* 38, 97–110 (1986). <https://doi.org/10.3402/tellusa.v38i2.11706>
16. Lions, J.L.: Contrôle Optimal de Systèmes Gouvernés par des Équations aux Dérivées Partielles. Dunod, Paris (1968). [https://doi.org/10.1016/0041-5553\(71\)90092-9](https://doi.org/10.1016/0041-5553(71)90092-9)
17. Marchuk, G.I.: Splitting and alternating direction methods. In: Ciarlet, P.G., Lions, J.L. (eds.) *Handbook of Numerical Analysis*, pp. 197–462. North-Holland, Amsterdam (1990). [https://doi.org/10.1016/S1570-8659\(05\)80035-3](https://doi.org/10.1016/S1570-8659(05)80035-3)
18. Marchuk, G.I.: *Adjoint Equations and Analysis of Complex Systems*. Kluwer, Dordrecht (1995). <https://doi.org/10.1007/978-94-017-0621-6>
19. Marchuk, G.I., Dymnikov, V.P., Zalesny, V.B.: Mathematical Models in Geophysical Hydrodynamics and Numerical Methods for their Implementation. Hydrometeoizdat, Leningrad (1987)
20. Marchuk, G.I., Rusakov, A.S., Zalesny, V.B., Diansky, N.A.: Splitting numerical technique with application to the high resolution simulation of the Indian Ocean circulation. *Pure Appl. Geophys.* 162, 1407–1429 (2005). <https://doi.org/10.1007/s00024-005-2677-8>
21. Sarkisyan, A., Sündermann, J.: *Modelling Ocean Climate Variability*. Springer, Heidelberg (2009). <https://doi.org/10.1007/978-1-4020-9208-4>
22. Shutyaev, V.P.: Methods for observation data assimilation in problems of physics of atmosphere and ocean. *Izv. Atmos. Ocean. Phys.* 55, 17–31 (2019). <https://doi.org/10.1134/S0001433819010080>
23. Zakharova, N.B.: Verification of the sea surface temperature observation data. *Sovremennye Problemy Distantionnogo Zondirovaniya Zemli iz Kosmosa* 13(3), 106–113 (2016). <https://doi.org/10.21046/2070-7401-2016-13-3-106-113>
24. Zalesny, V., Agoshkov, V., Aps, R., et al.: Numerical modeling of marine circulation, pollution assessment and optimal ship routes. *J. Mar. Sci. Eng.* 5, 1–20 (2017). <https://doi.org/10.3390/jmse5030027>
25. Zalesny, V.B., Agoshkov, V.I., Shutyaev, V.P., et al.: Numerical modeling of ocean hydrodynamics with variational assimilation of observational data. *Izv. Atmos. Ocean. Phys.* 52, 431–442 (2016). <https://doi.org/10.1134/S0001433816040137>
26. Zalesny, V.B., Gusev, A.V., Ivchenko, V.O., et al.: Numerical model of the Baltic Sea circulation. *Russ. J. Numer. Anal. Math. Model.* 28(1), 85–99 (2013). <https://doi.org/10.1515/rnam-2013-0006>
27. Zalesny, V.B., Marchuk, G.I., Agoshkov, V.I., et al.: Numerical simulation of large-scale ocean circulation based on the multicomponent splitting method. *Russ. J. Numer. Anal. Math. Model.* 25(6), 581–609 (2010). <https://doi.org/10.1515/rjnamm.2010.036>

A Supercomputer-Based Modeling System for Short-Term Prediction of Urban Surface Air Quality

Alexander V. Starchenko¹ , *Evgeniy A. Danilkin¹* ,

Sergei A. Prokhanov¹ , *Lubov I. Kizhner¹*, *Elena A. Shelmina^{1,2}* 

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

This paper proposes a mathematical model and an effective supercomputer-based numerical method for short-term prediction of extreme meteorological conditions and atmospheric air quality over limited stretches of land encompassing large population centers. The mathematical model includes a pollutant transport model with a reduced chemical mechanism and a non-hydrostatic mesoscale meteorological model with a modern moisture microphysics parametrization scheme. The numerical method relies on the use of the finite volume method and semi-implicit difference schemes of the second order of approximation, which are solved using the TDMA method with a linear dependence of the number of arithmetic operations on the size of the grid. This property of the numerical method ensures high efficiency when parallelized: not less than 70% when using up to 256 computing cores with a horizontal grid size of 0.5–1.0 km. Development of parallel programs was carried out using the Message Passing Interface parallel programming technology, two-dimensional decomposition of the grid area along horizontal (west to east and south to north) directions, and introduction of additional fictitious grid nodes along the perimeter of the decomposition subdomains.

Keywords: parallel computations, numerical weather prediction, mesoscale models, urban air quality, MPI.

Introduction

Currently, protection of the environment is becoming one of the most important tasks of science, interest in which is stimulated by the ever-increasing pace of technological progress around the world. Intensive industrial development and the resulting increase in industrial emissions of air pollutants are already starting to have a noticeable effect on the preservation of ecological balance in many regions of our planet. One of the most pressing issues is that of atmospheric air quality deterioration. The significance of this problem is primarily due to the fact that the atmosphere is one of the main vital elements of the environment. Moreover, the quality of the air in the lower part of the atmosphere, the part that is in contact with the Earth's surface, is of particular importance, since it is where the bulk of plant and animal life, including humans, reside. Presently, the deteriorating quality of the air is of particular concern due to changes in its chemical and aerosol composition caused by anthropogenic impact – emissions from industrial enterprises and transport [1].

Recently, mathematical modeling methods have been successfully used to monitor and forecast the ecological state of the urban atmosphere along with instrumental surveys. However, the complexity and interconnectedness of the processes of propagation, dispersion and chemical transformation of pollutant components occurring in the turbulent atmospheric boundary layer make air quality forecasting models cumbersome in mathematical notation and very demanding on computational resources [2]. In addition, reliable calculation of the vertical transport of pollutants, especially within the conditions of the convective boundary layer, requires the use of modern algebraic turbulence models to determine turbulent flows of momentum, mass and heat, which in itself is not an easy task. Thus, the problem of human interaction with the environment

¹National Research Tomsk State University, Tomsk, Russian Federation

²Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russian Federation

currently represents a new and actively developing field of application of mathematical modeling methods.

Currently, several modeling systems have been created and are functioning in the world, operating in real time, which rely on models of the atmospheric boundary layer and pollutant transport. The Enviro-HIRLAM [3] is designed as a fully integrated online system for numerical weather prediction and pollutant transport modeling for study and prediction of meteorological, chemical and biological weather. The integrated modeling system was developed by the Danish Meteorological Institute (DMI) in collaboration with other researchers and is used as the base system of the HIRLAM consortium. Development of the model was started at DMI more than ten years ago and it is now used by several countries. The current version of Enviro-HIRLAM [4] is based on the master version of HIRLAM 7.2 with a more complex and efficient chemistry scheme and aerosol dynamics modules based on a multimode aerosol-radiation and aerosol-cloud microphysics interaction approach. An immediate emergency report model has been developed. The system is optimized for use on vector supercomputers. For example, at the Finnish Meteorological Institute, calculations are carried out on the FMI Cray XC30 supercomputer, which includes 3420 computing units with peak performance of 70 Tflops.

ADMS-Urban is a modeling system developed by Cambridge Environmental Research Consultants with support from the UK Meteorological Office [5]. The system is used to monitor air quality and study complex situations in cities on highways, and in both non-metropolitan areas and large industrial centers. Currently, this model is used in cities in Europe and Asia, including China, as well as in the United States to monitor air pollution levels and ensure their compliance with modern standards. In the UK, over 70 local governments, including London, use ADMS-Urban for monitoring and evaluation, as well as for modeling possible consequences of industrial developments, such as the expansion of airports or construction of highways. The ADMS-Urban Regional Model system runs using the ARCHER UK National Supercomputing Service.

The EURAD-IM (EUROpean Air pollution Dispersion-Inverse Model) system [6], created at the University of Cologne Institute for Geophysics and Meteorology, simulates the physical, chemical and dynamic processes that affect the emission, formation, transport and deposition of atmospheric pollutants, and determines the distribution of concentrations in the troposphere over Europe, as well as the dry and wet deposition of air pollutants. This system consists of five submodules. Meteorological data calculations are carried out using the Weather Research and Forecast model [7], operations related to chemical transformations and pollutant transport are carried out using the EURAD-IM model, including the Regional Atmospheric Chemistry Mechanism (RACM) [8], and emission-related operations are carried out using the EURAD Emission Model (EEM). This system is used to monitor the formation of ozone and other photooxidants. The system uses the JURECA supercomputer at the Jülich Supercomputing Centre for Atmospheric air quality calculations.

The aim of the study is to develop efficient parallel numerical methods for a short-term high-resolution ($\sim 1 \text{ km}$) modeling system capable of forecasting extreme meteorological conditions and determining urban air quality, aimed at the use of a distributed memory supercomputer.

The article is organized as follows. Section 1 is devoted to mathematical state of the problem. In Section 2 we present a description of numerical method and its parallelization. Conclusion summarizes the study.

1. Description of the Mathematical State of the Problem

To calculate the concentration of pollutant components with regard to the chemical interactions between them, the proposed atmospheric air quality modeling system uses an Eulerian turbulent diffusion model, which includes non-stationary equations describing advection, turbulent diffusion and chemical reactions:

$$\begin{aligned} \frac{\partial \rho C_i}{\partial t} + \frac{\partial \rho u C_i}{\partial x} + \frac{\partial \rho v C_i}{\partial y} + \frac{\partial \rho w C_i}{\partial z} = \\ = \frac{\partial}{\partial x} \left(K_{xy} \frac{\partial C_i}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_{xy} \frac{\partial C_i}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_Z \frac{\partial C_i}{\partial z} \right) - \sigma_i C_i + S_i + R_i, \quad (1) \\ i = 1, \dots, n_s, \end{aligned}$$

$$-L_x/2 \leq x \leq L_x/2, -L_y/2 \leq y \leq L_y/2, h(x, y) \leq z \leq H.$$

Here, C_i is the dimensionless concentration of the i^{th} pollutant component; u and v are horizontal wind speed vector components; w is the vertical velocity component of the pollutant; ρ is the air density; K_{xy} and K_Z are diffusion coefficients; S_i is the intensity of pollutant component emission into the atmosphere from both elevated point and linear land-based sources; R_i describes the formation and transformation of substances through chemical and photochemical reactions involving pollutant components; σ_i is the rate of wet deposition of pollutants through precipitation; n_s is the number of chemical components of the pollutant, the concentration of which to be determined; x and y are horizontal coordinates, the Ox axis is directed to the east, Oy – to the north; z is the vertical coordinate; t is time. The computational domain is parallelepiped-shaped, L_x and L_y are the horizontal dimensions, H is its height, and $h(x, y)$ is the elevation above sea level. The transport model (1) predicts the chemical composition of surface air in an area of $50 \times 50 \text{ km}$ with a resolution of 500 m .

Background component concentration values are used as initial conditions; transport process and chemical reactions are spun up through a 24-hour period before the start of numerical surveys. The upper boundary of the examined area is set at 1 km , and the horizontal dimensions $L_x = L_y = 50 \text{ km}$. At the upper boundary, simple gradient conditions for concentrations are applied. At lateral boundaries, computational conditions of the so-called “radiation” type are used [9], providing computational stability of calculations in an open-border area.

At the lower boundary of the examined area, in the center of which the city is located, the conditions of dry deposition of the pollutant due to the resistance of the viscous sublayer, aerodynamic resistance and resistance caused by the distribution of vegetation are set [10].

Primary air pollutant emission sources taken into account are ground sources – motor vehicles – and elevated sources – factory chimneys. Emission parameters for elevated sources were assumed to be constant. For linear land-based sources, the standardized emission rate was set according to the following law:

$$I_{Vehicle}(t) = \begin{cases} 0.05 + 0.95 \sin(\pi(t - 6)/18), & t \in [6, 24], \\ 0.05, & t \notin [6, 24]. \end{cases} \quad (2)$$

Here t is the local time in hours. According to this formula, the maximum intensity of vehicle emissions is observed at about 15:00, local time. From 00:00 to 06:00, the intensity of vehicle emissions is minimal. The integral emission intensity of linear land-based sources and elevated

sources per day was calculated based on annual emissions of main pollutants in the Tomsk region for 2019 [11].

In this paper, R_i in (1) is estimated using a kinetic scheme based on two well-tested reduced chemical reaction mechanisms [12, 13]. The General Reaction Set semi-empirical chemical reaction mechanism [12] provides a compact description of the formation of secondary air pollutants ($PM_{2.5}$, PM_{10} , RP , H_2O_2 , etc). A more detailed representation of tropospheric ozone generation through photochemical reactions was taken from the reduced mechanism [13].

To calculate the speed vector components (u, v, w) and the turbulent diffusion coefficients, (1) uses a non-hydrostatic mesoscale meteorological model [14, 15]. This model uses a system of equations of hydrodynamics and heat and mass transfer in the troposphere, and an upper soil layer heat equation. The meteorological model predicts the wind speed components and temperature and humidity characteristics in the boundary layer of the atmosphere at 50 vertical levels (up to 10 000 m) above a territory of 150×150 km and an embedded area with a base of 50×50 km (the grid step is 1 km, the grid is centered on the city) for 24 hours. The basic equations of the mathematical representation of the model are presented below [16]:

$$\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0, \quad (3)$$

$$\begin{aligned} \rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + w \frac{\partial u}{\partial z} \right) = \\ - \frac{\partial p}{\partial x} + \rho f v + \frac{\partial}{\partial x} \left(K_{xy} \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_{xy} \frac{\partial u}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_Z^m \frac{\partial u}{\partial z} \right), \end{aligned} \quad (4)$$

$$\begin{aligned} \rho \left(\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + w \frac{\partial v}{\partial z} \right) = \\ - \frac{\partial p}{\partial y} - \rho f u + \frac{\partial}{\partial x} \left(K_{xy} \frac{\partial v}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_{xy} \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_Z^m \frac{\partial v}{\partial z} \right), \end{aligned} \quad (5)$$

$$\begin{aligned} \rho \left(\frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial y} + w \frac{\partial w}{\partial z} \right) = \\ - \frac{\partial p}{\partial z} - \rho g + \frac{\partial}{\partial x} \left(K_{xy} \frac{\partial w}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_{xy} \frac{\partial w}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_Z^m \frac{\partial w}{\partial z} \right). \end{aligned} \quad (6)$$

Here, t is time, u , v and w are longitudinal, transverse and vertical components of the average wind speed vector in the direction of the Cartesian coordinates x , y , z , respectively, f is the Coriolis parameter, K_{xy} is the horizontal diffusion coefficient, K_Z^m is the momentum vertical diffusion coefficient, g is gravitational acceleration, and p is pressure.

$$\begin{aligned} \rho \left(\frac{\partial \theta}{\partial t} + u \frac{\partial \theta}{\partial x} + v \frac{\partial \theta}{\partial y} + w \frac{\partial \theta}{\partial z} \right) = \\ \frac{\partial}{\partial x} \left(K_{xy} \frac{\partial \theta}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_{xy} \frac{\partial \theta}{\partial y} \right) + \frac{\partial}{\partial z} \left(K_Z^h \frac{\partial \theta}{\partial z} \right) + \frac{\theta}{c_p T} (Q_{rad} - \rho L_w \Phi). \end{aligned} \quad (7)$$

Here, T is the absolute temperature, θ is the potential temperature, $\theta = T(p_0/p)^{R_0/c_p}$, c_p is the air heat capacity at constant pressure, $p_0 = 101300$ N/m², R_0 – the gas constant, Q_{rad} is heating (or cooling) of the atmosphere due to long- and shortwave radiation heat fluxes propagating in a humid atmosphere, $\rho L_w \Phi$ is the temperature change caused by phase transitions

of moisture in the atmosphere, K_z^h is the heat and moisture vertical diffusion coefficient, L_w is vaporization heat, and q_V is the relative density of atmospheric steam.

$$p = \rho RT, R = R_0 \left[\frac{1 - q_V}{M_{air}} + \frac{q_V}{M_{H_2O}} \right]. \quad (8)$$

Simulation of water moisture phase transformation processes in the atmosphere in this study is carried out using the WSM6 6-class moisture microphysics scheme [17]. This scheme simulates the processes that occur between the six states of atmospheric moisture (water vapor, cloud water content, rain, ice particles, snow and graupel (hail)). For each of the parameters characterizing the state of moisture in the atmosphere, a transport equation is used, which includes, along with advective transport, various parameterizations of physical processes leading to changes in the phase state of the aforementioned forms of moisture.

To close the system of equations (1), (3)–(8), a turbulence model is used, including an equation for kinetic energy [18], as well as algebraic relations for determining turbulent diffusion coefficients [19]. The horizontal diffusion coefficient is calculated using the Smagorinsky formula [20].

In addition, the model takes into account the following: heat transfer through shortwave and longwave radiation in the examined layer of the atmosphere with regard to clear-sky scattering and attenuation, water vapor absorption, absorption and reflection by clouds [21, 22]; turbulent momentum, heat and moisture exchange with the underlying surface [23, 24]; temperature change in the upper layer of the surface and humidity at the “air-underlying surface” boundary.

The initialization of the mesoscale numerical weather prediction model and its provision with lateral boundary conditions is carried out based on the results of numerical weather forecasting by the Hydrometeorological Centre of Russia's SL-AV operational global model [25].

2. Numerical Method and its Parallelization

Thus, the basis of the mathematical formulation of the problem in question is a generalized inhomogeneous differential equation of convective diffusion transport:

$$\begin{aligned} \frac{\partial \rho \Phi}{\partial t} + \frac{\partial \rho u \Phi}{\partial x} + \frac{\partial \rho v \Phi}{\partial y} + \frac{\partial \rho w \Phi}{\partial z} &= \frac{\partial}{\partial x} \left(K_{xy} \frac{\partial \Phi}{\partial x} \right) + \frac{\partial}{\partial y} \left(K_{xy} \frac{\partial \Phi}{\partial y} \right) + \\ &+ \frac{\partial}{\partial z} \left(K_z^{\Phi} \frac{\partial \Phi}{\partial z} \right) + S_{\Phi}. \end{aligned} \quad (9)$$

Here, Φ represents different sought-for functions in different equations (1), (3)–(7). To account for in relief of the underlying surface $h(x, y)$, equation (9) is brought to the following form:

$$x' = x; y' = y; z' = \frac{z - h(x, y)}{H - h(x, y)} H. \quad (10)$$

As a result of this transformation, mixed derivatives appear. Here, H is the height of the computational domain. Equations such as (9) are solved using the finite volume method using grids with uniform spacings along the horizontal x and y axes and spacings along the vertical z axis getting progressively finer towards the Earth's surface. The lower boundary is 10 m above the surface. Differential equation (9) is approximated using the finite volume method with second-

order spatial variable approximation and explicit-implicit time approximations [26], which also provide second-order accuracy in time.

$$\begin{aligned}\Phi_h^{n+1} = & \Phi_h^n + \frac{\Delta t_n}{2} (3L_h(\Phi_h^n) - L_h(\Phi_h^{n-1})) + \\ & + \frac{\Delta t_n}{2} (\Lambda_h(\Phi_h^{n+1}) + \Lambda_h(\Phi_h^n)) + \frac{\Delta t_n}{2} (3S_\Phi(\Phi_h^n) - S_\Phi(\Phi_h^{n-1})),\end{aligned}\quad (11)$$

where $\Phi_h^n = \{\Phi_{i,j,k}^n\}$ is the grid function of the scalar Φ for which the differential equation (9) is given; n is the time layer number; $i = 1, \dots, Nx$ is the number of grid nodes along the x axis; $j = 1, \dots, Ny$ is the number of grid nodes along the y axis; $k = 1, \dots, Nz$ is the number of grid nodes along the z axis; L_h is the finite-volume analogue of the convective-diffusive operator in equations (9) except for the vertical diffusion along the z axis, Λ_h is the difference analogue of the differential operator of vertical diffusion $\frac{\partial}{\partial z} (K_z \frac{\partial \Phi}{\partial z})$, $S_\Phi(\Phi)$ is the source terms of equation (9). Implicit approximation for vertical diffusion transport, which is important in the boundary layer of the atmosphere, allows to avoid a more rigid time restriction on the integration step. When approximating the convective terms of equation (9), Van Leer's monotonized linear upstream schemes are used [27]. For diffusion terms, ordinary approximations of second-order accuracy are used. These approximations result in a formation of a difference scheme, in which values of the grid function $\{\Phi_{i,j,k}^{n+1}\}$ on a new time layer ($n+1$) can be calculated using the tridiagonal matrix algorithm [28] independently along vertical grid lines. As a result of using this method of forming the difference scheme (11), the number of arithmetic operations depends linearly on the grid size and conditions are created for perfectly parallel processing.

To match the finite-difference values of the speed and pressure vector field at each time step, the predictor-corrector scheme is used. Its main idea is that, firstly, the grid values of the speed vector components are predicted using difference schemes of the form (11) for known grid pressure function p_h^n values at the n^{th} time layer. Then an elliptic difference equation is solved for pressure correction $p'_h = p_h^{n+1} - p_h^n$ and the intermediate speed and pressure fields are corrected. This operation is carried out with the requirement that the corrected values of the speed components exactly satisfy the difference analogue of the continuity equation (3). The described scheme includes the following sequence of operations at each n^{th} time step:

1. Approximate speed values $\tilde{u}_h^{n+1}, \tilde{v}_h^{n+1}, \tilde{w}_h^{n+1}$ are calculated using difference formulas of the type (11) for a known pressure field p_h^n .
2. An equation of the following form for pressure correction p'_h is solved

$$\begin{aligned}ap_{i,j,k}(p')_{i,j,k} - ab_{i,j,k}(p')_{i,j,k-1} - at_{i,j,k}(p')_{i,j,k+1}^{l+1} - ae_{i,j,k}(p')_{i+1,j,k}^l - \\ - an_{i,j,k}(p')_{i,j+1,k}^l - aw_{i,j,k}(p')_{i-1,j,k}^{l+1} - as_{i,j,k}(p')_{i,j-1,k}^{l+1} = b_{i,j,k}(p'_h), \\ i = \overline{1, Nx}; j = \overline{1, Ny}; k = \overline{1, Nz}.\end{aligned}\quad (12)$$

3. A new pressure field $p_h^{n+1} = p_h^n + p'_h$ is determined.
4. Speed components $u_h^{n+1}, v_h^{n+1}, w_h^{n+1}$ are calculated using speed values $\tilde{u}_h^{n+1}, \tilde{v}_h^{n+1}, \tilde{w}_h^{n+1}$ and pressure correction value p'_h .

In view of this and the results of conducted computational experiments, a decision was made to calculate pressure correction p'_h using the Gauss-Seidel line by line iteration method [29] with red and black grid node arrangement for each horizontal level k and implicit representation of

sought-for values of the grid function p'_h in the nodes $(i, j, k + 1)$, (i, j, k) , $(i, j, k - 1)$, i.e., along the vertical grid lines:

$$\begin{aligned}
& -ab_{i,j,k} (p')_{i,j,k-1}^{l+1} + ap_{i,j,k} (p')_{i,j,k}^{l+1} - at_{i,j,k} (p')_{i,j,k+1}^{l+1} = \\
& = ae_{i,j,k} (p')_{i+1,j,k}^l + an_{i,j,k} (p')_{i,j+1,k}^l + \\
& + aw_{i,j,k} (p')_{i-1,j,k}^{l+1} + as_{i,j,k} (p')_{i,j-1,k}^{l+1} + b_{i,j,k} ((p')_h^l), \\
& i = \overline{1, Nx}; j = \overline{1, Ny}; k = \overline{1, Nz},
\end{aligned} \tag{13}$$

l is the number of iteration.

Thus, the main complexity of the program for calculating pollutant concentrations and meteorological fields lies in solving convective-diffusion equations of the form (11) and elliptic equations for pressure field correction (13).

2.1. The Choice of a Parallel Programming Technology

To find the most suitable parallel programming technology for solving the numerical prediction problems in question and implement the chosen numerical method, test calculations were carried out. As one of the subtasks for testing, a mathematical statement with mixed boundary conditions was chosen for one non-stationary inhomogeneous convection diffusion equation (9), the numerical solution of which used difference scheme (11). Open MultiProcessing (OpenMP), Message Passing Interface (MPI), Open Accelerators (OpenACC) and Compute Unified Device Architecture (CUDA) parallel programming technologies [30] were used for test calculations. The calculations used a $256 \times 256 \times 32$ grid and 5000 time steps. Calculations were carried out using Tomsk State University's Cyberia cluster. Each node of the cluster has the following specifications: 48Gb RAM, 2×Intel Xeon X5670 (2.93GHz), 1×GPU NVIDIA GeForce RTX 2080 Ti. Average sequential program runtime for one node is 1733.0 seconds.

When using OpenMP, OpenACC and CUDA technologies, only one node of the cluster with shared RAM for two CPUs and one GPU can be used. For the test case in question and the chosen numerical method, OpenMP programming technology was applied by distributing a set of $Nx \times Ny$ independent subtasks of solving tridiagonal systems between independent parallel threads. To do this, the OpenMP parallel directives were used. As a result of the tests, a speedup of 10.5 was achieved (Fig. 1). It was also found that for the case in question, parallelization of two external loops (i and j) yields the same speedup results as parallelization of just the i loop. Using static, dynamic and guided schedules also does not yield a significant speedup compared to the default iteration distribution.

Attempts were also made to use hybrid CPU/GPU technologies – OpenACC and CUDA – to solve the test problem (9) using a graphics card. When using OpenACC, similarly to OpenMP, compiler directives were used to assign independent subtasks i and j , consisting of solving tridiagonal systems of linear equations (11), to GPU cores. This approach yielded a runtime of 20.4 seconds.

When using CUDA technology, an algorithm for solving a tridiagonal system of linear equations dependent on the indices i and j , was also chosen as the compute kernel. These systems were distributed among GPU cores and solved for each time step with subsequent synchronization of computing processes. The use of CUDA made it possible to obtain a solution to the problem in 16.6 seconds.

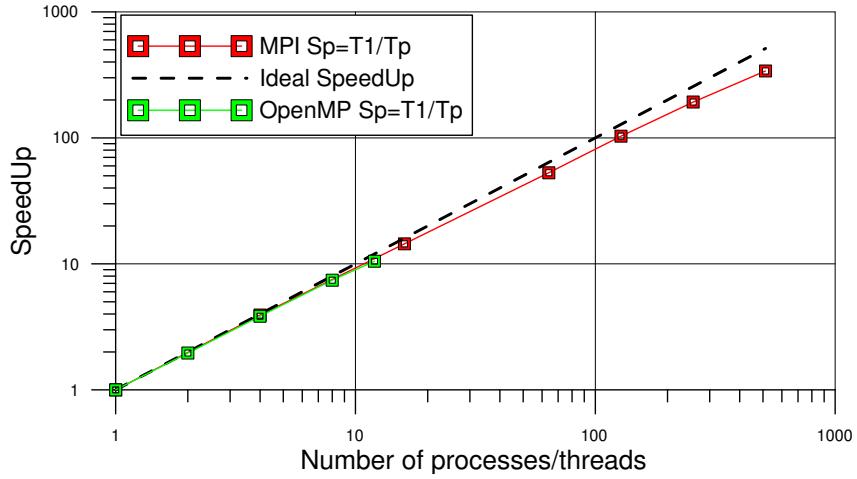


Figure 1. Comparison of execution speedup of parallel programs developed using OpenMP and MPI, tested using TSU's Cyberia cluster

Thus, when the amount of data transmitted between software modules or the number of equations to be solved (9) is small, the use of GPUs for implementation of parallel programming technologies shows more promise.

MPI technology is also often considered for development of parallel programs on multiprocessor distributed-memory systems. For the problem in question (9) and the chosen numerical method (11), this technology was used in combination with a two-dimensional decomposition of the grid area along the grid lines i and j (in the directions of the x and y axes). Blocking communication functions MPI_Sendrecv were used to transmit messages between processes. Replacing these functions with non-blocking communication functions MPI_Isend and MPI_Irecv yielded no more than a 5% decrease in the total execution time of the test task, therefore it was not considered further. To ensure uniformity of parallel computations, two rows of dummy cells were used along the perimeter of each subdomain of the two-dimensional decomposition, due to the use of Van Leer's monotonized upstream scheme [27]. Tridiagonal systems of linear equations (11) were solved simultaneously in each subdomain. With this method of organizing parallel computing using technology on one node of the Cyberia cluster, a speedup of 11.4 was achieved when solving the test problem (9). Note that when performing parallel calculations using the same program on 128 processes, a speedup was obtained by 103 times (average calculation time 16.8 sec), and by 512 – 337 times (average time 5.1 sec).

When developing a parallel method for solving equations (13), it should be noted that the iterative scheme corresponds well to the chosen parallel algorithm for solving transport equations (11) and allows the use of two-dimensional decomposition and asynchronous relaxation [29] with a minimum amount of data exchange to synchronize parallel computations in decomposition subdomains. Unfortunately, at each iteration, such data exchanges must be performed, which makes it relevant to use rapidly converging iterative methods with a minimum number of iterations when they are implemented in parallel.

Thus, if the program code contains several software modules with a large amount of information transmitted between them during computations, which is a property of numerical implementation of mathematical models with a large number of equations (more than ten) of the form (9) and additional algebraic relations, then it is advisable to use the Message Passing Interface technology.

2.2. Parallel Implementation of the Numerical Method for TSUNM3

In view of the results of the computational experiments presented above, the working version of the TSUNM3 model of numerical weather prediction (3)–(8) was parallelized using 2D decomposition and MPI technology. Table 1 shows the resulting speedup values (calculated as the ratio of the execution time of the serial version of the code to the execution time of the parallel program) for the parallel program when performing weather forecast computations for the local research area on two grids for two days: $96 \times 96 \times 50$ nodes (Grid1) and $192 \times 192 \times 50$ nodes (Grid2). Since the plan is to use the TSUNM3 mesoscale meteorological model with a horizontal resolution of 1–2 km, Grid1 corresponds to an industrial area with a large number of enterprises, transport hubs and a large (\sim million residents) population center at the center of a 100–200 km – wide research area. Grid2 can cover an area of 200–400 km and can be used for short-term weather prediction in an administrative area with several large population centers and transport hubs. For the selected grid sizes, mesoscale model calculations were performed on the TSU Cyberia cluster.

Table 1. TSUNM3 parallel program speedup values for various grids at different process quantities

Processes	4	9	16	36	64	144	256
Grid1	3.9	8.6	15.2	33.8	59.0	119.1	184.7
Grid2	3.9	8.5	14.8	32.1	58.1	129.2	218.8

The speedup values for parallel computing programs with an increasing number of processes presented in Tab. 1 show good supercomputer resource utilization efficiency. With the results obtained, it is possible to carry out predictive calculations for the next day in 12 minutes of the program's runtime for Grid1 and 39 minutes for Grid2 using 256 cores of the TSU Cyberia supercomputer. Note that in the calculations, we used a version of the program in which calculations are performed with single precision, which makes it possible to reduce the computation time and the amount of data transmitted between computing nodes.

To clarify the results presented in Tab. 1, diagrams of the relative (in %) time costs of the work of each of the main blocks were constructed and the speedup values of the execution of each of these blocks for the considered number of processes were calculated. The main blocks of the program are: block for calculating the interaction with the underlying surface (friction, heat and mass transfer) and the exchange of values – WF&Exchange; velocity component calculation block – Velocities; block for calculating the hydrostatic and non-hydrostatic parts of pressure – Pressure; block for calculating turbulent kinetic energy and turbulent diffusion – TKE; moisture microphysics calculation block – MicroPhysics; block for solving the equation of heat transfer in the atmosphere – Heat; block for assembling distributed data and writing it to disk – Record. In these blocks, except for the Record block, only the computation time was measured. We also measured the total time required for interprocessor data transfer in 2D decomposition to perform parallel computing (MPI_Send).

Figure 2 shows that a large share in the total amount of computational costs is the execution of blocks for calculating the velocity components and moisture microphysics parameters. This is understandable, because three equations of the form (9) are solved in the first block, and four in the second. That is, on average, the numerical solution of one equation (9) by method (11) takes

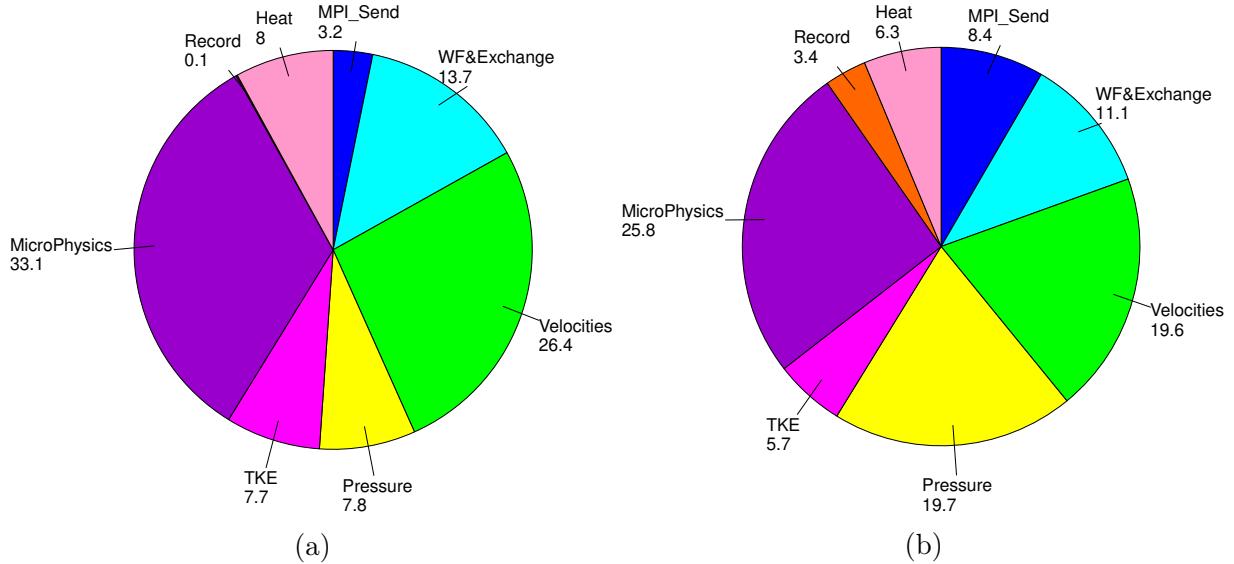


Figure 2. Diagrams of the relative time spent on the execution of various blocks of the program. (a) is the launch of the program on 36 processes, (b) – on 256. The calculations were performed on the grid Grid1

about 8–9% of the total computational time when using up to 36 processes. This is confirmed by the time spent on the implementation of blocks for the numerical solution of the heat transfer equation and the calculation of turbulent characteristics (Fig. 2), where one equation of the form (9) is solved. Note that when solving finite-difference equations of the form (11) numerically, due to the nonlinearity (advection/turbulent diffusion) of equation (9) and the presence of time-consuming source terms (especially for the MicroPhysics, Velocities, Heat blocks), it is required to perform quite a lot of arithmetic operations to calculate the coefficients of equations (11) for each dependent variable. Note also that the WF&Exchange block does not require data exchange and runs with perfect parallelism. Its share in the total cost of the program is more than 10%. The costs of implementing the Pressure block with a relatively small number of processes are commensurate with the costs of numerically solving one transport equation (9). However, with an increase in the degree of decomposition of the grid domain, the rate of convergence of the iterative Gauss-Seidel method slows down and more iterations are required to achieve a given accuracy. This entails an increase in the parallel execution time of the Pressure block (Figs. 2, 3).

After calculating the grid functions of the desired dependent variables for each equation of the form (11), in accordance with the selected grid pattern, the boundary values of the grid subdomain in the xz and yz planes are exchanged. Naturally, with an increase in the number of processes used, the ratio of the computation time for each subdomain to the time of exchange of boundary grid values to ensure the uniformity of parallel computations decreases (Fig. 2).

Figure 3 shows that, due to a fairly large number of auxiliary calculations in calculating the coefficients of finite-difference equations (11), due to its nonlinearity and time-consuming source terms, the speedup of all blocks of the parallel code, except for the pressure calculation block, is close to ideal even with an increase in the fraction time spent on exchange operations at each time step. This character of the speedup of calculations is preserved even with a relatively small amount of grid nodes in the 2D decomposition region. This is confirmed by the above results of speedup in the numerical solution of one convective-diffusion equation (Fig. 1).

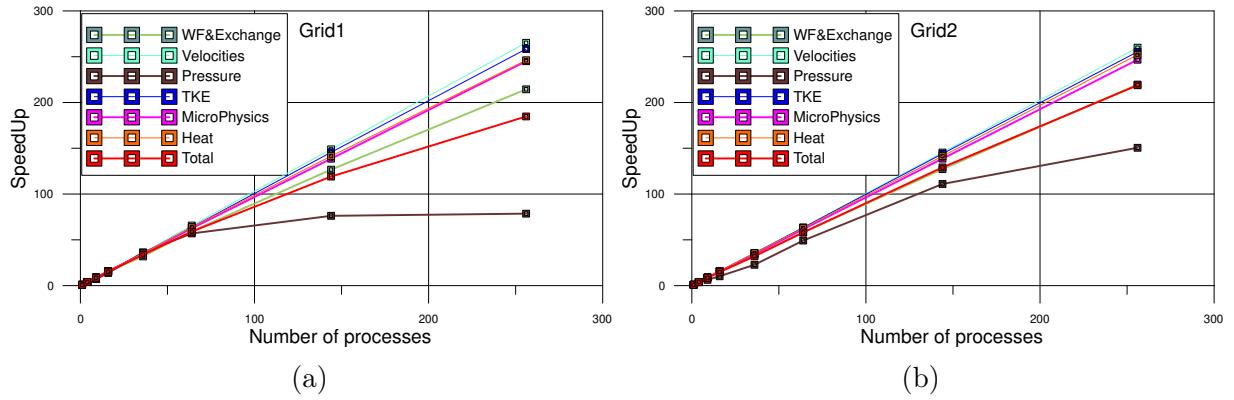


Figure 3. The speedup of the main blocks of the TSUNM3 program depending on the number of processes used. The calculations use Grid1 (a) and Grid2 (b)

When solving the finite-difference equation (13) by the iterative Gauss-Seidel method at each time step, exchange operations must be performed at each iteration. Therefore, for the considered grids Grid1 and Grid2 and the parallel programming technology, already at 144 processes, speedup saturation for the pressure calculation block is observed and the time spent on its numerical implementation begins to occupy an increasing share in the total amount of calculations. Nevertheless, since the rest of the code blocks demonstrate a high level of speedup, in general, the high efficiency of using a multiprocessor computing system remains for the entire parallel program (Fig. 3, Tab. 1).

To prove the good strong, relative and weak scalability [31] of the developed parallel program, we also present Fig. 4, which shows the graphs of absolute efficiency E , relative efficiency rE and “weak” efficiency wE from the number of processes used, which are calculated by the following formulas:

$$E(p, \text{Grid}) = \frac{T(1, \text{Grid})}{pT(p, \text{Grid})}; rE(p, \text{Grid}) = \frac{E(p, \text{Grid})}{E(p_{\text{prev}}, \text{Grid})}; wE(p) = \frac{T(p, \text{Grid1})}{T(4p, \text{Grid2})}. \quad (14)$$

Here p is the number of processes used, Grid is the size of the grid area, $T(p, \text{Grid})$ is the program computation time on the Grid on p processes, p_{prev} is the number of processes in the sequence of test runs preceding p (see Tab. 1, in accordance with which, for example, at $p = 36$, $p_{\text{prev}} = 16$), Grid2 is a grid with 4 times more nodes than Grid1.

Figure 4 shows that the developed computational code has good scalability, since its absolute efficiency is not less than 70% when using up to 256 processes. In addition, the ratio of neighboring values of absolute efficiency starts to decrease below 0.9 only when more than 100 processes are used. The values of weak scalability wE also do not fall below 0.9, which shows a small change in the computation time with a synchronous increase in the grid size and the number of processes used, while maintaining the number of processed grid values in each subdomain of the 2D decomposition.

Thus, the reasons for the high scalability of the developed TSUNM3 program compared to parallel programs for the numerical solution of traditional Navier-Stokes equations with constant coefficients, which consider three equations for the velocity components and an equation for finding pressure (see, for example, [32]), in our opinion, are the following:

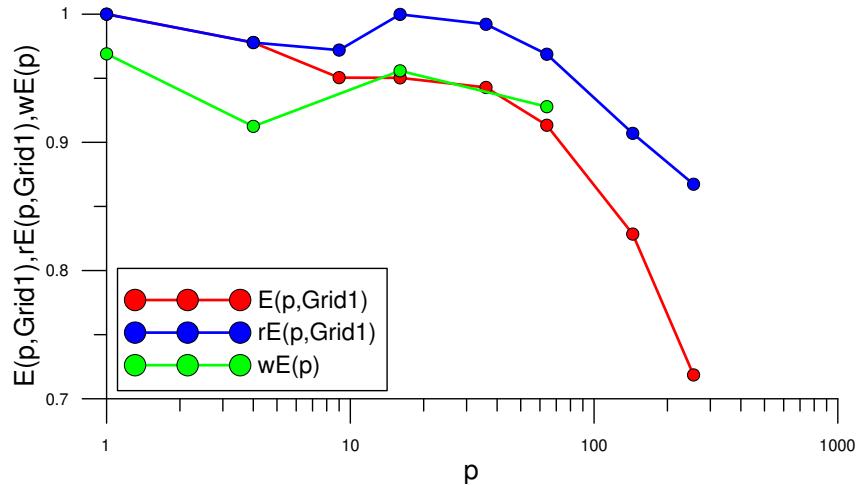


Figure 4. The scalability of the developed program

- in this program, six more three-dimensional non-stationary convective-diffusion equations of the form (9) are additionally solved numerically;
- the developed parallel numerical method for solving the transport equation (9), as shown by the above results, has good scalability up to the use of a relatively small number of nodes in the grid subdomains obtained after applying the two-dimensional decomposition; the latter is associated with a large volume of calculations of the coefficients of difference equations (11) and source terms of equations (9), especially for blocks of microphysics of moisture, turbulence and heat transfer;
- apparently, with more intensive use of fast cache memory with an increase in the degree of decomposition of the finite difference problem with an increase in the number of processes used.

The numerical method (11) and MPI parallel programming technology with two-dimensional decomposition of the grid area along horizontal directions were also used for the pollutant transport model (1), which is applied in off-line mode after calculations using the TSUNM3 numerical weather prediction model. The computations in question include 12 equations of the form (1), which are solved on a $100 \times 100 \times 50$ grid with a horizontal step of 500 m and a time step of ~ 1 sec for a 50×50 km area. Table 2 shows the speedup values for the pollutant transport model (1) parallel program.

Table 2. Pollutant transport parallel program speedup values at different process quantities

Process	4	16	25	100
SpeedUp	3.9	13.6	19.4	70.8

Thus, the table shows that the efficiency of parallel computing for the model of air quality (1) in question does not fall below 70%.

Conclusion

A mathematical model and an effective numerical method for the short-term prediction of dangerous meteorological conditions and atmospheric air quality over limited stretches of

land encompassing large population centers designed to be used on multiprocessor distributed-memory systems were developed in the course of this study. The mathematical model includes a pollutant transport model with a reduced chemical mechanism and a non-hydrostatic mesoscale meteorological model with a modern atmospheric moisture microphysics parametrization scheme. The numerical solution of the non-stationary inhomogeneous convection diffusion equations of the modeling system is carried out by means of the transformation of coordinates, the finite volume method and semi-implicit difference schemes of the second order of approximation, which are solved using the TDMA method with a linear dependence of the number of arithmetic operations on the size of the grid. Difference equations used for pressure field computations that ensure correct satisfaction of continuity equations in the difference form are solved using the Gauss-Seidel iteration method and asynchronous relaxation.

Parallel implementation of the chosen numerical scheme for solving the equations of the mathematical model was based on the use of MPI parallel programming technology, two-dimensional decomposition of the grid area along horizontal directions (x and y axes), and introduction of additional fictitious grid nodes along the perimeter of the decomposition subdomains. The results of computational experiments carried out using the TSU Cyberia cluster showed good software scalability and computer resource utilization efficiency were not less than 70% when using up to 256 of the cluster's computing cores. In this case, numerical prediction of local weather conditions for the next 24 hours was carried out in 12 minutes on a $96 \times 96 \times 50$ grid and in 39 minutes on a $192 \times 192 \times 50$ grid with a horizontal resolution of 1 km. Numerical prediction of surface air quality for the next 24 hours was carried out in 23 minutes on a $100 \times 100 \times 50$ grid with a horizontal resolution of 500 m.

Acknowledgements

The work was supported by the Russian Science Foundation (project no. 19-71-20042).

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. Sokhi, R., Baklanov, A., Schlünzen, H.: Mesoscale Modelling for Meteorological and Air Pollution Applications. Anthem Press, New York (2018)
2. Dabdub, D., Seinfeld, J.H.: Parallel Computation in Atmospheric Chemical Modeling. Parallel Computing 22, 111–130 (1996). [https://doi.org/10.1016/0167-8191\(95\)00063-1](https://doi.org/10.1016/0167-8191(95)00063-1)
3. Baklanov, A.A., Korsholm, U.S., Mahura, A.G., et al.: Physic a land chemical weather forecasting as a joint problem: two-way interacting integrated modelling. American Meteorological Society, Seattle (2011)
4. Nuterman, R., Korsholm, U., Zakey, A., et al.: New developments in Enviro-HIRLAM online integrated modelling system. Geophysical Research Abstracts 15 (2013)
5. Hood, C., MacKenzie, I., Stocker, J., et al.: Air quality simulations for London using a coupled regional-to-local modelling system. Atmospheric Chemistry and Physics 18, 11221–

- 11245 (2018). <https://doi.org/10.5194/acp-18-11221-2018>
6. Strunk, A., Ebel, A., Elbern, H.: A nested application of four-dimensional variational assimilation of tropospheric chemical data. International Journal of Environment and Pollution 46(1–2), 43–60 (2011). <https://doi.org/10.1504/IJEP.2011.042607>
 7. Skamarock, W.C., Klemp, J.B., Dudhia, J., et al.: A Description of the Advanced Research WRF Model Version 4. National Center for Atmospheric Research, Colorado (2021). <https://doi.org/10.5065/1dfh-6p97>
 8. Stockwell, W.R., Kirchner, F., Kuhn, M., Seefeld, S.: A new mechanism for regional atmospheric chemistry modeling. Journal of Geophysical Research Atmospheres 102(22), 25847–25879 (1997). <https://doi.org/10.1029/97jd00849>
 9. Carpenter, K.: Note on the paper ‘Radiation conditions for the lateral boundaries of limited-area numerical models’. Quarterly Journal of the Royal Meteorological Society 108(457), 717–719 (1982). <https://doi.org/10.1002/qj.49710845714>
 10. Wesley, M.L.: Parameterisation of surface resistances to gaseous dry deposition in regional-scale numerical models. Atmospheric Environment 23(6), 1293–1304 (1989). [https://doi.org/10.1016/0004-6981\(89\)90153-4](https://doi.org/10.1016/0004-6981(89)90153-4)
 11. Department of Natural Resources and Environmental Protection of the Tomsk Region. <https://depnature.tomsk.gov.ru/2019-god>, accessed: 2021-10-29
 12. Hurley, P.J.: TAPM V4. Part 1: Technical Description. CSIRO Marine and Atmospheric Research, Aspendale (2008). <https://doi.org/10.4225/08/585c175bc5884>
 13. Stockwell, W.R., Goliff, W.S.: Comment on “Simulation of a reacting pollutant puff, using an adaptive grid algorithm” by R.K. Srivastava et al. Journal of Geophysical Research Atmospheres 107(22), 4643–4650 (2002). <https://doi.org/10.1029/2002JD002164>
 14. Starchenko, A.V., Bart, A.A., Kizhner, L.I., Danilkin, E.A.: Mesoscale meteorological model TSUNM3 for the study and forecast of meteorological parameters of the atmospheric surface layer over a major population center. Tomsk State University Journal of Mathematics and Mechanics 66, 35–55 (2020). <https://doi.org/10.17223/19988621/66/2>
 15. Starchenko, A., Prokhanov, S., Danilkin, E., Lechinsky, D.: Numerical Forecast of Local Meteorological Conditions on a Supercomputer. In: Voevodin, V., Sobolev, S. (eds.) Supercomputing. RuSCDays 2020. Communications in Computer and Information Science, vol. 1331, pp. 273–284. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64616-5_24
 16. Pielke, R.A.: Mesoscale Meteorological modelling. Academic Press, Orlando (1984)
 17. Hong, S., Lim, J.: The WRF single-moment 6-class microphysics scheme (WSM6). Journal of the Korean Meteorological Society 42(2), 129–151 (2006)
 18. Andrén, A.: Evaluation of a Turbulence Closure Scheme Suitable for Air-Pollution Applications. Journal of Applied Meteorology and Climatology 29(3), 224–239(1990). [https://doi.org/10.1175/1520-0450\(1990\)029<0224:EOATCS>2.0.CO;2](https://doi.org/10.1175/1520-0450(1990)029<0224:EOATCS>2.0.CO;2)

19. Yamada, T.: Simulations of Nocturnal Drainage Flows by a q^2l Turbulence Closure Model. *Journal of Atmospheric Sciences* 40(1), 91–106 (1983). [https://doi.org/10.1175/1520-0469\(1983\)040<0091:SONDFB>2.0.CO;2](https://doi.org/10.1175/1520-0469(1983)040<0091:SONDFB>2.0.CO;2)
20. Smagorinsky, J.: General Circulation Experiments With the Primitive Equations: Part I. The Basic Experiment. *Monthly Weather Review* 91(2), 99–164 (1963). [https://doi.org/10.1175/1520-0493\(1963\)091<0099:GCEWTP>2.3.CO;2](https://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2)
21. Mahrer, Y., Pielke, R.A.: A numerical study of the airflow over irregular terrain. *Beitr. Phys. Atmosph.* 50, 98–113 (1977)
22. Stephens, G.: Radiation Profiles in Extended Water Clouds. Part II: Parameterization Schemes. *Journal of the Atmospheric Sciences* 35(11), 2123–2132 (1978). [https://doi.org/10.1175/1520-0469\(1978\)035<2123:RPIEWC>2.0.CO;2](https://doi.org/10.1175/1520-0469(1978)035<2123:RPIEWC>2.0.CO;2)
23. Monin, A.S., Obukhov, A.M.: Basic laws of turbulent mixing in the surface layer of the atmosphere. *Tr. Akad. Nauk SSSR Geofiz.* 24, 163–187 (1954)
24. Dyer, A.J., Hicks, B.B.: Flux-gradient relationships in the constant flux layer. *Quarterly Journal of the Royal Meteorological Society* 96(410), 715–721 (1970). <https://doi.org/10.1002/qj.49709641012>
25. Tolstykh, M., Goyman, G., Fadeev, R., Shashkin, V.: Structure and Algorithms of SL-AV Atmosphere Model Parallel Program Complex. *Lobachevskii Journal of Mathematics* 39(4), 587–595 (2018). <https://doi.org/10.1134/S1995080218040145>
26. Mazumder, S.: Numerical Methods for Partial Differential Equations. Finite Difference and Finite Volume Methods. Academic Press (2016)
27. Van Leer, B.: Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second order scheme. *Journal of Computational Physics* 14(4), 361–370 (1974). [https://doi.org/10.1016/0021-9991\(74\)90019-9](https://doi.org/10.1016/0021-9991(74)90019-9)
28. Patankar, S.: Numerical Heat Transfer and Flow. CRC Press (2009). <https://doi.org/10.1201/9781482234213>
29. Ortega, J.O.: Introduction to Parallel and Vector Solution of Linear System. Plenum Press New York, Charlottesville (1988)
30. Starchenko, A.V., Danilkin, E.A., Prohanov, S.A., Leshchinskiy, D.V.: Parallel implementation of a numerical method for solving transport equations for the mesoscale meteorological model TSUNM3. *Journal of Physics: Conference Series* 1715(1), 012073 (2020). <https://doi.org/10.1088/1742-6596/1715/1/012073>
31. Bruneau, C.-H., Khadra, K.: Highly parallel computing of a multigrid solver for 3D Navier-Stokes equations. *Journal of Computational Science* 17(1), 35–46 (2016). <https://doi.org/10.1016/j.jocs.2016.09.005>
32. Wang, Y., Baboulin, M., Dongarra, J., et al.: A Parallel Solver for Incompressible Fluid Flows. *Procedia Computer Science* 18, 439–448 (2013). <https://doi.org/10.1016/j.procs.2013.05.207>

River Routing in the INM RAS-MSU Land Surface Model: Numerical Scheme and Parallel Implementation on Hybrid Supercomputers

Victor M. Stepanenko^{1,2} 

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

The land surface model (LSM) is a necessary compartment of any numerical weather forecast system or the Earth system model. This paper presents a new version of the INM RAS-MSU land surface model where the river hydrodynamic and thermodynamic scheme is embedded into the parallel execution framework using MPI and OpenMP. Numerical experiments have been performed for the East European domain with resolution $0.5^\circ \times 0.5^\circ$. The soil model parallel efficiency at 1–144 MPI cores was 0.52–0.79 and limited by the presence of ocean area, and by imbalance of computational load between soil columns. The acceleration of the river model at MPI level was defined by the size of the largest river basin in the domain. At the OpenMP level, the potential for acceleration of large river basin simulation is shown to be close to number of threads used, based on fractal properties of the river networks. This acceleration was hindered in our numerical experiments by the reduced river orders at the coarse land surface model resolution, so that the optimal speedup for the Volga river basin was 2.5–3 times attained at 4–6 threads. This performance is projected to improve with refinement of the LSM spatial resolution.

Keywords: *land surface model, soil, river network, MPI, OpenMP.*

Introduction

The land surface scheme (or land surface model, LSM) is a necessary compartment of any numerical weather forecast system or the Earth system model. It reproduces the thermodynamic, hydrophysical and ecological state of land active layer as well as momentum, mass and energy fluxes between Earth surface and the atmosphere. These fluxes are important to reproduce well for reliable weather forecasts at a broad range of time scales: from subdiurnal to seasonal. Especially, the soil moisture is widely recognized as a critical parameter for seasonal weather dynamics and prediction [6, 19, 32], as it defines the structure of the soil surface heat balance.

Rivers are an integrating component of the land water cycle, as they gather soil moisture from large terrestrial areas and transport it to ocean. The significance of rivers in the Earth system is caused by their notable contribution to the ocean freshwater budget [14, 23], methane, carbon dioxide [17, 21, 29] and dissolved organic carbon [2, 3] transfer from land to ocean and emission of greenhouse gases to the atmosphere. In addition, the river discharge at large timescales is close to accumulated “precipitation minus evaporation” over the river basin, and thus serves as a useful proxy for this difference during the land surface model validation. Those considerations led to introduction to the land surface models the so-called river routing schemes representing river flows in simplified manner [1, 11, 15, 22, 31].

The land surface models for long have been consisting of a large number of independent vertical 1D soil problems, enabling straightforward parallel implementation using MPI and the longitude-latitude domain decomposition technique. However, embedding river module introduces horizontal dependency of river variables and thus existing longitude-latitude decomposition is not optimal for LSM river module. This calls for development of new parallel implementation

¹Lomonosov Moscow State University, Moscow, Russian Federation

²Moscow Center of Fundamental and Applied Mathematics, Moscow, Russian Federation

approach to LSMs with advanced river simulators. Our paper addresses this objective for the INM RAS-MSU land surface scheme.

The paper is organized as follows. Section 1 presents general information on the INM RAS-MSU model basic equations and physical parameterizations, with detailed description of the soil module. Section 1.2 is devoted to the recently developed thermo- and hydrodynamic model of the river network inside INM RAS-MSU LSM. Information structure of the INM RAS-MSU simulator with river block is elaborated in Section 2; the corresponding levels of parallelism are analyzed in Section 3. Configuration of numerical experiments with parallelized LSM is described in Section 4. Results of experiments are demonstrated and discussed in Section 5. Conclusions summarize key results of the study, and sketch future directions for the optimization of LSM parallel implementation.

1. The INM RAS-MSU Land Surface Scheme

The land surface scheme used in this study is jointly developed by Institute of Numerical Mathematics (INM), Russian Academy of Sciences, and Lomonosov Moscow State University (MSU) (Fig. 1). It is a part of the INM-CM5 Earth system model [27] and of the SL-AV numerical weather forecast system [10]. Here, we consider it in a standalone version, where the atmospheric forcing is prescribed. It reproduces the thermodynamic and hydrophysical state of soil, lakes and rivers as well as momentum, mass and energy fluxes between Earth surface and the atmosphere. The effects of surface vegetation on surface-atmosphere exchanges are represented via modifications of soil-air exchange laws (based mostly on Monin-Obukhov similarity) and introduction of liquid water sink in soil due to roots suction. This means, no vegetation layer storage for heat or mass is considered. Subgrid-scale variability of the land surface is simulated by tile approach, where contribution of each surface type in a cell to cell-averaged fluxes of heat, radiation, water vapor and carbon is proportional to its areal fraction. Carbon cycle processes accounted for into the model include photosynthesis, organic matter decay in soils with CO₂ release and organics degradation in wetlands with CH₄ formation. The most computationally expensive parts of INM RAS-MSU land model are soil, lake and river modules. The soil and river modules are described in more detail below. The lake model is similar in its information structure to the soil model, as both of them comprise a number of independent 1D problems.

1.1. Soil Model

The basic numerical kernel of any land surface scheme is a solver for an equation system describing heat and water transport in soil and snow, including phase transitions. In INM RAS-MSU model, this system includes equations as follows. The heat equation is:

$$\rho_d c \frac{\partial T}{\partial t} = \frac{\partial}{\partial z} \lambda_T \frac{\partial T}{\partial z} + \rho_d (L_i F_i - L_v F_v), \quad (1)$$

where $\rho_d c$ is a volumetric heat capacity of soil, T is temperature, z is a coordinate directed along acceleration due to gravity, t is time, λ_T is coefficient of heat conductivity, ρ_d is dry soil bulk density, L_k and F_k are the specific heat and rate of water freezing/melting ($k = i$) and evaporation/condensation ($k = v$), respectively. The heat conductivity λ_T is a function of liquid water content W and dry soil conductivity. An equation for liquid water content W describes vertical transport (diffusion due to capillary and sorption forces and gravitational infiltration),

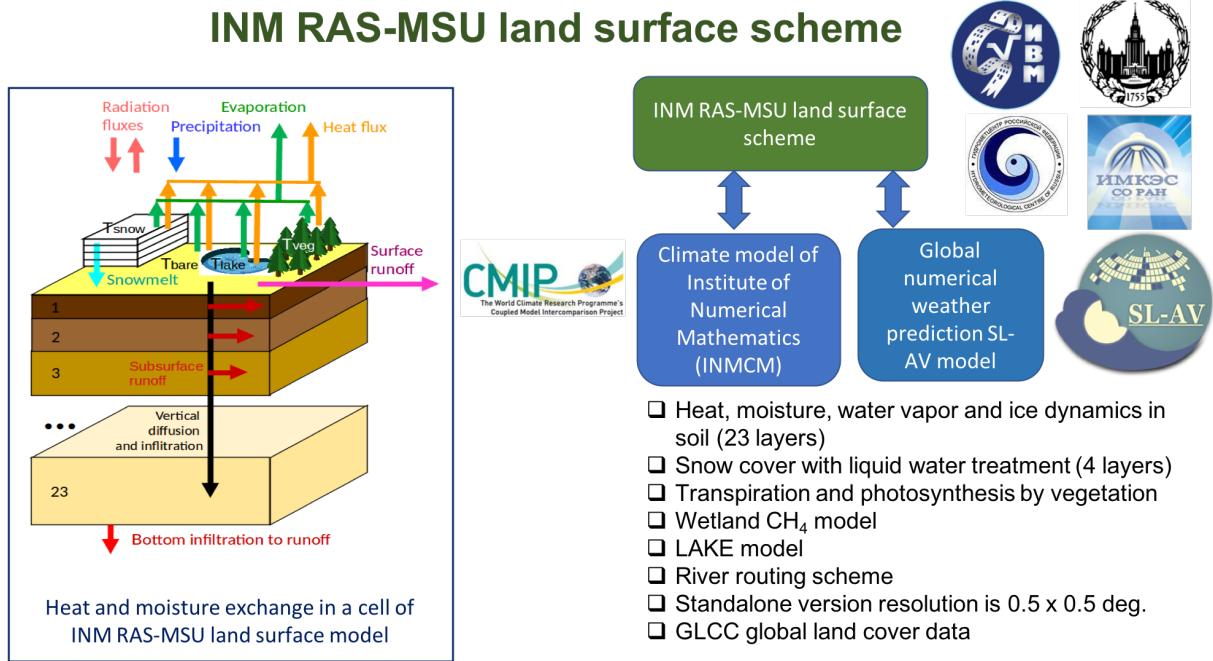


Figure 1. INM RAS-MSU land surface model

freezing/melting and evaporation/sublimation, root uptake and horizontal discharge [16]:

$$\frac{\partial W}{\partial t} = \frac{\partial}{\partial z} \left(\lambda_W \frac{\partial W}{\partial z} + \lambda_I \frac{\partial I}{\partial z} \right) + \frac{\partial \gamma}{\partial z} - F_i - F_v - S_r - S_l. \quad (2)$$

Here, W is a ratio of liquid water mass to solid soil matrix mass, S_r is root suction, and S_l is a sink due to lateral water flow. A dependence of soil moisture potential Ψ (or water retention curve, WTC) and hydraulic conductivity γ on moisture W and ice content I are important features of the system, defining coefficients $\lambda_W, \lambda_I, \gamma$ as functions of the solution (λ_I is neglected in the current version of the model). The liquid water diffusivity $\lambda_W(W)$ and hydraulic conductivity $\gamma(W)$ are related functions:

$$\lambda_W(W) = \gamma(W) \frac{\partial \Psi(W)}{\partial W}. \quad (3)$$

At least 22 semi-empirical forms are proposed for WTC [9], fitting different sets of empirical data with different performance. The WTC function explicitly enters the hydraulic conductivity function γ . For instance, choosing Mualem approach for hydraulic conductivity quantification [20], one gets:

$$\gamma = \gamma_{max} \widetilde{W}^{1/2} \left[\int_0^{\widetilde{W}} \frac{d\widetilde{W}'}{\Psi(\widetilde{W}')} \left(\int_0^1 \frac{d\widetilde{W}'}{\Psi(\widetilde{W}')} \right)^{-1} \right]^2, \quad (4)$$

where $\widetilde{W} \doteq (W - W_{min}) / (W_{max} - W_{min})$ is a degree of soil moisture saturation. Thus, introducing in (4) and (3) different forms of WTC yields corresponding pairs of functions (λ_W, γ) , based on Mualem equation. In INM RAS-MSU model, the two most widespread sets of (λ_W, γ) are implemented, namely those by Brooks-Corey [4, 5] and Mualem-van Genuchten [12, 20].

The content of water vapor (V , expressed as a mass ratio, similar to W) is governed by diffusion equation and phase transitions:

$$\frac{\partial V}{\partial t} = \frac{\partial}{\partial z} \lambda_V \frac{\partial V}{\partial z} + F_v, \quad (5)$$

while the dynamics of ice content I is defined by phase transitions only:

$$\frac{\partial I}{\partial t} = F_i. \quad (6)$$

The system of four equations above is supplemented by boundary conditions, representing heat and water mass balance at the soil-air interface $z = 0$ and the bottom of soil column $z = H$. At the surface, heat and water balance equations include net radiation and modification of fluxes by vegetation canopy, while at bottom zero diffusive flux condition is imposed. In cold season, snow depth dynamics is simulated as well as temperature and snow moisture vertical distribution [24, 28]:

$$\rho_{sn}c_{sn}\frac{\partial T}{\partial t} = \frac{\partial}{\partial z}\lambda_T\frac{\partial T}{\partial z} + \rho_{sn}L_iF_i, \quad (7)$$

$$\frac{\partial W}{\partial t} = \frac{\partial\gamma}{\partial z} - F_i, \quad (8)$$

where the subscript “sn” denotes thermodynamic properties of snow. This system is coupled to the soil equations set via continuity of fluxes and temperature at the soil-snow interface.

The system is solved by implicit in time and central-differences in space numerical scheme with 23 levels in soil down to 10 m depth, 4 levels in snow (if present) and 1 hour time step for every cell of a regular latitude-longitude grid ($0.5^\circ \times 0.5^\circ$ in this study). In order to ensure the heat balance equation at the soil surface, iteration procedure in respect to surface temperature is implemented.

The code is written in Fortran and uses the external libraries for I/O in netcdf format, MPI exchanges and OpenMP threading.

1.2. The Model for River Hydrodynamics and Thermodynamics

The model for river hydrodynamics and thermodynamics is based on diffusive wave approximation for 1D (i.e. averaged over the vertical cross-section of a stream) Saint-Venant system [24]. Under this approach, the pressure gradient force is balanced by quadratic bottom friction, which delivers a closed set of equations for vertical cross-section area S and temperature T :

$$\frac{\partial S}{\partial t} + \frac{\partial([U_0 + U_*]S)}{\partial x} = E_r + \frac{\partial}{\partial x}k_S\frac{\partial S}{\partial x}, \quad (9)$$

$$\frac{\partial(ST)}{\partial t} + \frac{\partial([U_0 + U_*]ST)}{\partial x} = u_{tr}T_{tr}h_{tr} + b_sF + \frac{\partial}{\partial x}k_{ST}\frac{\partial S}{\partial x}, \quad (10)$$

supplemented with boundary conditions:

$$S|_{x=0} = \left.\frac{\partial S}{\partial x}\right|_{x=L} = 0, \quad (11)$$

$$ST|_{x=0} = 0, \quad (12)$$

where the along-river coordinate $x = 0$ corresponds to the river origin and $x = L$ locates at the river mouth, E_r is a water volume inflow rate from soil and tributaries per unit of river length, $u_{tr}T_{tr}h_{tr}$ is heat inflow per river unit length from soil and inlets, b_s is a width of a river surface, F is net kinematic heat and radiation flux at the water surface, U_* , k_S and k_{ST} are functions of channel slope and geometry. The net radiation includes shortwave and longwave components, net heat flux includes sensible and latent heat flux calculated by the same Monin-Obukhov similarity as for the soil model (with water-specific roughness lengths). The ice cover

is not explicitly simulated in the current version, and water temperature is allowed to decrease below 0°C.

The system is solved for each river by Preisman scheme semi-implicit in time with 10 mesh nodes per the model cell (Fig. 2) and Vreman conservative filtering [30] to suppress two-step oscillations.

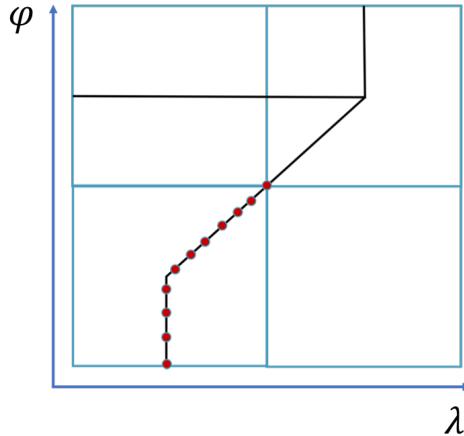


Figure 2. The nodes of the river numerical scheme (red circles on a black line, the black line representing river course) on a grid of land surface model (blue rectangles), λ is longitude, ϕ is latitude (adopted from [24])

2. Information Structure of the Land Surface Model with River Routing

The soil and river models are one-way coupled. It means that the solution of the soil problem affects the solution of the river model (via groundwater runoff), but not vice versa. The total water volume source in the stream continuity equation (9) is $E_r = E_{r,s} + E_{r,r}$, where $E_{r,s}$ is the soil runoff, and $E_{r,r}$ is the water input from tributaries. The soil water runoff to rivers in each model cell is:

$$E_{r,s} = \frac{A}{\rho_{w0} L_A} \left[(\rho_d \gamma) |_{z=H} + \int_0^H \rho_d S_l dz \right], \quad (13)$$

where A is a cell area, L_A is a total river length in a cell, and ρ_{w0} is the reference water density. Thus, an update of soil variables in a cell during timestep provides $E_{r,s}$ over this time step to be used in a river model as a longitude-latitude field. The water input from tributaries $E_{r,r}$ is:

$$E_{r,r} = [(U_0 + U_*)S]_{tr}/L_A, \quad (14)$$

with subscript “tr” denoting the tributary(ies) of the river, if any in the cell. The heat source in (10) is:

$$u_{tr} T_{tr} h_{tr} = E_{r,s} H_a^{-1} \int_0^{H_a} T dz + [(U_0 + U_*)ST]_{tr}/L_A, \quad (15)$$

with H_a standing for active soil layer depth, assumed 1 m in current model version. The above formulas mean that in order to advance river variables at each time step the soil runoff and discharge of tributaries are necessary. Thus the sequence of model execution at each time step is as follows (omitting other parts of algorithm such as lakes, I/O, etc.):

- soil model (time step i),
- river model (time step i):
 - rivers of the 1st order,
 - rivers of the 2nd order,
 - ...
 - rivers of the maximal order.
- soil model (time step $i + 1$),
- river model (time step $i + 1$):
 - rivers of the 1st order,
 - ...

Here, we introduced the Strahler orders of rivers, where by definition, rivers of order n have tributaries of the order not larger than $n - 1$, and the streams having no inlets are of $n = 1$. This is the strict sequence that cannot be changed under selected numerical scheme. The independent in terms of information exchange parts of the algorithm are:

- time step updates of different soil columns,
- time step updates of different river basins,
- time step updates of different rivers of the same order.

Here, we used a term “basin” to denote a river network with highest-order river flowing into the ocean or a lake with no outlet. The serial and parallel parts of the algorithm indicated above cause the levels of parallel implementation described in the following section.

The INM RAS-MSU land surface model with river routine scheme described above was shown to successfully reproduce the annual cycle of runoff of two North Eurasian rivers: Severnaya Dvina and Kolyma [18, 24].

3. Levels of Parallelism and Analytical Estimates for Model Speedup

The soil model is parallelized using standard decomposition of longitude-latitude domain in $M_\lambda M_\phi$ subdomains of the same dimensions, where M_λ and M_ϕ are the numbers of MPI processes along those two coordinates, respectively. Given no data exchanges are needed between any two MPI processes for soil simulations, this part of the model should speedup with efficiency ≈ 1 under even computational load of processes.

The river model parallel implementation consists of two levels. The top level comprises the distribution of river basins between MPI processes. This distribution is realized as follows. Let us assume, there are N_b basins in the model domain, and n_i , $i = 1, \dots, N_b$ is a decreasing sequence of numbers of land model cells constituting these basins. Then, the MPI process of rank 0 simulates the largest basin ($i = 1$), whereas k -th process computes dynamics of rivers in basins labeled $i = i_{1,k}, \dots, i_{m_k,k}$, so that the total number of cells in those basins $\sum_{i=i_{1,k}, \dots, i_{m_k,k}} n_i \leq n_1$, and closest possible to n_1 . Such a scheme ensures distribution of basins between a small number of MPI processes which is even in terms of total number of land model cells, but not necessarily in number of basins.

To supply the river model with input data (meteorological variables, inflow of water volume and heat from soil to streams), each MPI process should have access to the 2D (lan-lot) arrays of those variables covering all river basins, which are associated to this process. However, these 2D arrays are originally split between MPI-processes according to 2D domain decomposition of

the soil model (see the first paragraph of this Section, and Fig. 5 in Section 4). This issue is solved by allocating auxiliary 2D array covering the whole simulation domain which is identical on each MPI process and which is filled in with the above mentioned variables by ALLREDUCE operation. This operation is performed one time each time step after the call of soil update (PBLFLX subroutine) and before the river update (RIVWAT subroutine).

The second level of parallelizm of the river model is realized for each MPI process, and uses the informational independency of river equations solution for rivers of the same order. This level is implemented using OpenMP instructions. Below we estimate the maximal speedup for a single river basin simulation following this approach, which is attainable under no overhead costs.

Let ω be the Strahler order of a river. We can now introduce the values n_ω (total number of watercourses of order ω in a given basin) and L_ω (average length of watercourses of order ω in the basin). According to Horton's laws, which can be derived from an approximation that the river network is a fractal [8, 25], the following relations are statistically valid:

$$\frac{n_{\omega-1}}{n_\omega} = C_n > 1, \quad (16)$$

$$\frac{L_{\omega-1}}{L_\omega} = C_L < 1, \quad (17)$$

where C_n and C_L are constants of the river basin. The wall-clock time of solving a one-dimensional problem for one river at one time step can be expressed as $O(L_\omega^k)$, where k is defined by numerical scheme, e.g., $k = 1$ for explicit schemes and implicit schemes, where linear systems are solved by direct factorization method (the case of our model). Now estimate the time of sequential processing T_s , i.e., time needed for sequential solution of one-dimensional problems for all rivers of the river basin. For the number of rivers of order ω and their length we have:

$$n_\omega = n_{\omega-1}C_n^{-1} = \dots = n_1C_n^{-(\omega-1)}, \quad (18)$$

$$L_\omega = L_{\omega-1}C_L^{-1} = \dots = L_1C_L^{-(\omega-1)}. \quad (19)$$

Let ω_{max} be a maximal order of rivers in the basin. Then

$$T_s = \sum_{\omega=1}^{\omega_{max}} n_\omega O(L_\omega^k) = n_1 O(L_1^k) \sum_{\omega=1}^{\omega_{max}} C_L^{-k(\omega-1)} C_n^{-(\omega-1)}. \quad (20)$$

Since $1 = n_{\omega_{max}} = n_1 C_n^{-(\omega_{max}-1)}$, and introducing $C_L^* \doteq C_L^{-1} > 1$, the expression for T_s reads:

$$T_s = C_n^{\omega_{max}-1} O(L_1^k) \sum_{\omega=0}^{\omega_{max}-1} C_n^{-\omega} C_L^{*\omega}. \quad (21)$$

If all rivers of the same order are processed simultaneously by different cores, then the update of all rivers of order ω takes the time $O(L_\omega^k)$ rather than $n_\omega O(L_\omega^k)$ as in the sequential version, so that the total processing time of the whole river network T_p is estimated as:

$$T_p = O(L_1^k) \sum_{\omega=0}^{\omega_{max}-1} C_L^{*\omega}, \quad (22)$$

and the speedup of the whole river basin simulation:

$$\frac{T_s}{T_p} = \frac{C_n^{\omega_{max}-1} \sum_{\omega=0}^{\omega_{max}-1} C_n^{-\omega} C_L^{*\omega}}{\sum_{\omega=0}^{\omega_{max}-1} C_L^{*\omega}}. \quad (23)$$

If for the river basin processing only m cores are available, the above estimate transforms to:

$$\frac{T_s}{T_p} = \frac{\sum_{\omega=0}^{\omega_{max}-1} C_n^{\omega_{max}-\omega-1} C_L^{*k\omega}}{\sum_{\omega=0}^{\omega_{max}-1} \max [C_n^{\omega_{max}-\omega-1}/m, 1] C_L^{*k\omega}}. \quad (24)$$

An example of river model speedup for single basin processing according to (24) is shown in Fig. 3. One can see that for largest river basins (with maximal river order exceeding 6–8) the speedup approaches the number of cores used. This is important, as the largest basins cause a bottleneck at the MPI level of river model implementation, and their effective speedup at the OpenMP level may significantly reduce the execution time of the whole river module.

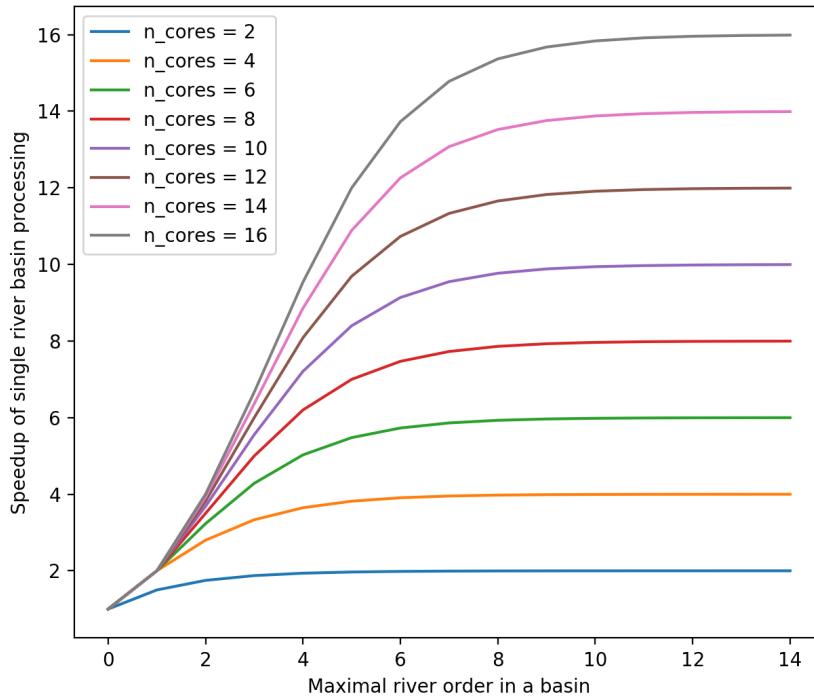


Figure 3. Theoretical speedup of a single river basin processing with no overhead costs, as estimated from fractal theory ($C_n = 4$, $C_L^* = 2$)

4. Configuration of Numerical Experiments

Numerical experiments with INM RAS-MSU land surface model have been conducted for domain 40° – 80° N, 20° – 59.5° E with horizontal resolution $0.5^\circ \times 0.5^\circ$ and for the period of 31 days. Atmospheric forcing (surface air temperature, humidity, wind speed, downwelling longwave and shortwave radiation, pressure, precipitation) was read from GFDL-ESM2M piControl simulation data, 1–31 Jan 1661, bias-corrected EWEMBI dataset of the ISIMIP2b project (<https://www.isimip.org>). Atmospheric data has the same grid with the model, temporal resolution is daily, the model time step is 1 hour. The data for river flow directions and riverbed slopes are taken from the database of GWSP-WATCH project based on DDM30 data [7] made accessible through ISIMIP project as well. The resulting river network of the Volga basin is shown in Fig. 5. On

the model mesh, Volga river has the order 6, which is much less than that defined for the river network obtained at much finer resolution (the Volga order is 9 at 200–500 m resolution, [13]).

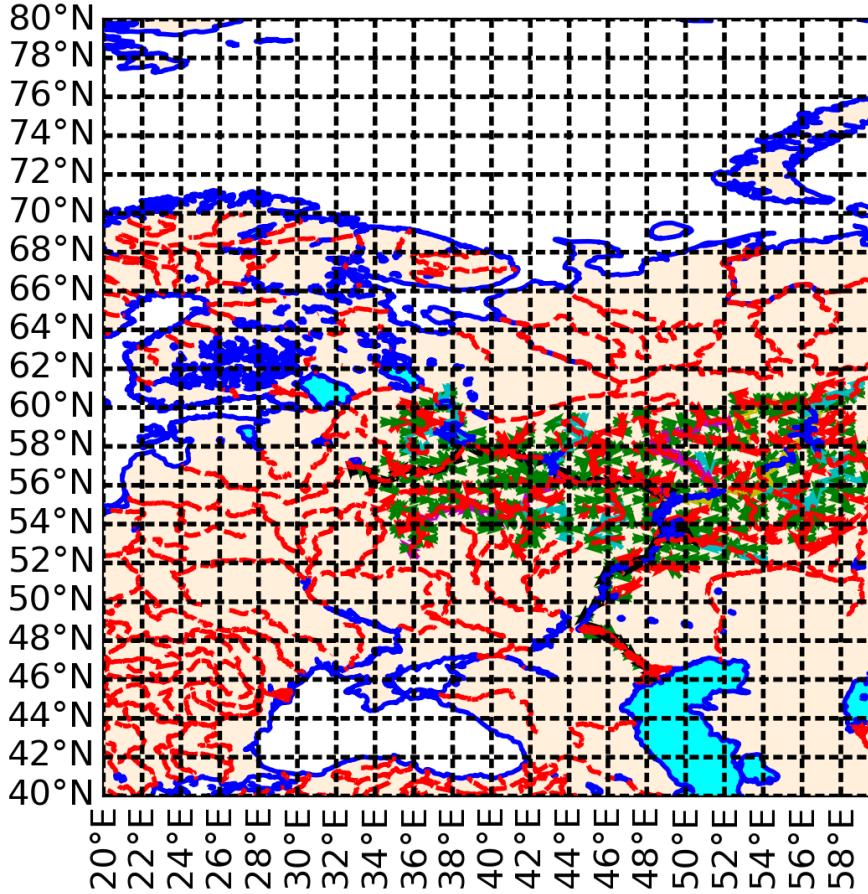


Figure 4. The full domain of the INM RAS-MSU land surface model simulations. The small green arrows depict the Volga river basin, shown in detail in Fig. 5

Two series of experiments have been performed to measure the model speedup:

- MPI series: 1 OpenMP thread, number of MPI processes: 1, 4 ($M_\lambda = M_\phi = 2$), 16 ($M_\lambda = M_\phi = 4$), 36 ($M_\lambda = M_\phi = 6$), 64 ($M_\lambda = M_\phi = 8$), 100 ($M_\lambda = M_\phi = 10$), 144 ($M_\lambda = M_\phi = 12$),
- OpenMP series: 4 MPI processes ($M_\lambda = M_\phi = 2$), number of OpenMP threads: 1, 2, 4, 6, 8, 10, 12, 14.

The performance of the model at OpenMP level is analyzed for the largest basin in the simulation domain (Volga basin).

Model simulations have been conducted using Lomonosov-2 supercomputer [26], using nodes with Intel Haswell-EP E5-2697v3, 2.6 GHz, 14 cores and Infiniband FDR interconnect. The executable of the land surface model was assembled with ifort compiler using -O3 optimization. In MPI series of runs, the model was launched as:

```
sbatch --ntasks=<number of MPI processes> --bind-to none <executable>
```

whereas for OpenMP series:

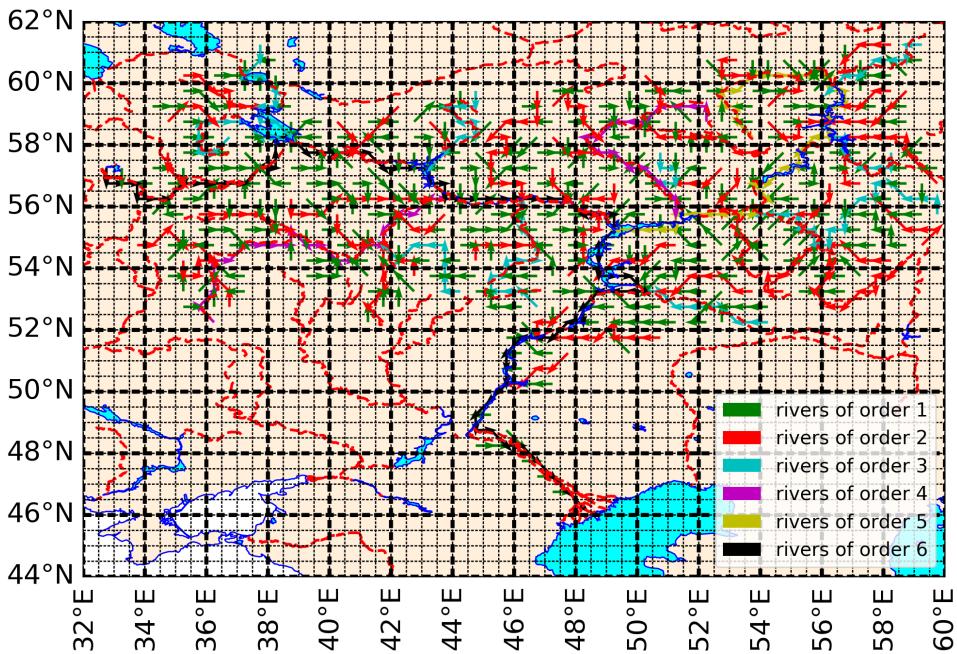


Figure 5. Volga river basin on the INM RAS-MSU model mesh (a subdomain of the full model domain shown in Fig. 4), each arrow denotes the river course segment containing mesh nodes depicted in Fig. 2, black thin dotted lines are used to show model cells, thick dashed black lines delineate MPI subdomains, color is used to denote the river Strahler order

```
sbatch --ntasks=4 --ntasks-per-node=1 --bind-to none <executable>
```

The latter ensures that each MPI process runs at a separate node, and all OpenMP threads of a given MPI process locate at the same node.

5. Results and Discussion

The wall-clock time and parallel efficiency (defined as the speedup of parallel version in respect to serial code version divided by number of MPI processes) for experiments of MPI series are shown in Fig. 6a and 6b, respectively.

The soil model accelerates with gradual decrease of efficiency from 0.79 at 4 cores to 0.52 at 144 cores, despite the absence of MPI exchanges in this part of the code. The reason is the load imbalance between MPI processes, taking into account that the wall-clock time of the soil model at any timestep is defined be the maximal wall-clock time among processes. At 4 cores, all MPI subdomains include ocean cells and different number of land cells which define the distribution of workload between processes. At large numbers of $M_\phi M_\lambda$ there are subdomains with all cells residing over ocean, which means corresponding MPI processes are idle. The fraction of such processes is estimated as a fraction of ocean cells which is 30%. The efficiency may be corrected for this factor by taking into account only number of MPI processes, processing land cells, e.g. for 144 cores we have the corrected efficiency $0.52/(1 - 0.3) = 0.74$. The remaining loss of efficiency $1 - 0.74 = 0.26$ is mostly explained by the increase of the time for one-timestep processing of a single soil column, averaged over a subdomain of the most loaded MPI-process, with rise of the cores number (from 3.9×10^{-5} s at 1 MPI process to 4.9×10^{-5} s at 144 MPI processes).

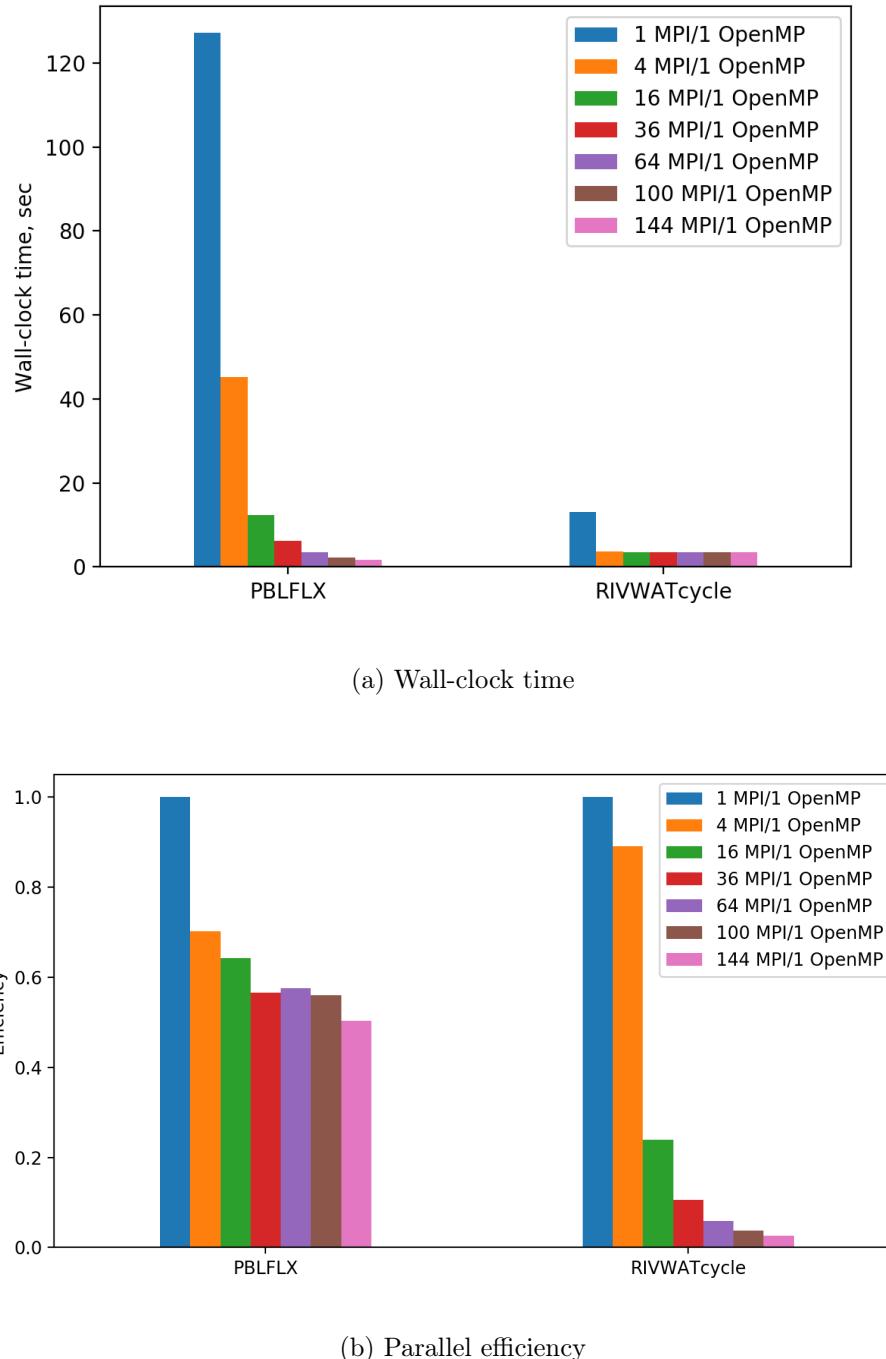


Figure 6. Wall-clock time and parallel efficiency of the soil model (subroutine PBLFLX) and the river model (subroutine RIVWATcycle) using 1–144 MPI processes on Lomonosov-2 supercomputer (MPI series)

This means that even fully terrestrial MPI subdomains have different workload depending on the presence of snow cover and different number of iterations for surface temperature to perform a timestep depending on the local meteorological conditions.

The regular longitude-latitude decomposition of the LSM computational domain between MPI processes is inherited from the fully coupled ESM, where this decomposition matches the decomposition in the atmospheric model. However, as shown above, in standalone mode it be-

comes non-optimal if rectangular domain in (λ, ϕ) contains ocean cells. In this case, the current MPI decomposition scheme can be optimized, in order the MPI subdomains to contain only land cells or a minimal number of ocean cells. This model improvement is left for the future.

Remarkably, the river model is accelerated almost 4 times when using 4 MPI processes, whereas under larger number of cores the wall-clock time remains almost constant. This is due to fact that at the MPI level, the largest river basin in the domain (Volga basin) is processed by single process, and the wall-clock time of this process is a minimal possible for the river model in general. As a result, the river model contribution to the whole land surface model runtime, being negligible in serial mode starts to dominate over the soil model above ~ 100 MPI processes (3.49 s for river model vs. 1.75 s for soil model).

The time for ALLREDUCE operation gathering the data for the river model input gradually increases from 0.22 s at $M_\phi M_\lambda = 4$ to 0.66 s at $M_\phi M_\lambda = 144$ indicating increasing overhead costs of this collective operation, still remaining much less than river model computational time (3.49 s).

The wall-clock time and parallel efficiency of Volga basin processing in experiments of OpenMP series are shown in Fig. 7a and 7b, respectively. The efficiency is defined here in respect to the model configuration with 4 MPI processes and 1 OpenMP thread.

Simulation of rivers with smallest order demonstrated the highest speedup and efficiency, whereas the rivers of order 5 and 6 are simulated with no acceleration. The overall decrease of Volga simulation time when using 14 threads is 4.11 times which is much smaller compared to estimated ≈ 12 times from theoretical formula (24). To address this difference, consider the Volga basin scale-similarity parameters (Horton metrics) computed on the model mesh (Fig. 5), presented in Tab. 1.

Table 1. The Horton metrics for Volga basin at $0.5^\circ \times 0.5^\circ$

Horton parameter / Strahler order	$\omega = 2$	$\omega = 3$	$\omega = 4$	$\omega = 5$	$\omega = 6$
$C_{n,\omega}$	5.911	4.5	5	2	1
$C_{L,\omega}^*$	9.098	2.293	2.188	1.943	1.971

The river length parameter $C_{L,\omega}^*$ is close to typical value of 2, excluding the case of $\omega = 2$, because the lengths of low-order rivers are not explicitly reproduced on the model mesh (e.g., the majority of first-order rivers have length of the one model cell). On the contrary, the river number parameter $C_{n,\omega}$ is biased from typical value of ~ 4 for the largest order 5 and 6. The value $C_{n,6} = 1$ means that there are one river of order 5 and one river of order 6, which are both simulated sequentially by one OpenMP thread (see no speedup for those rivers in Fig. 7a). There are two rivers of order 4, so that the acceleration of their simulation ceases with the number of threads > 2 . Gradual decrease of parallel efficiency of simulation of the rivers with orders 1 and 2 while increasing number of threads is caused by increasing the contribution of overheads of initializing the OpenMP PARALLEL section and DO loop each timestep to the total wall-clock time.

To check the reasoning of the previous paragraph, the formula (24) can be extended to account for variability of Horton coefficients with river order ω :

$$\frac{T_s}{T_p} = \frac{n_1 \left[1 + \sum_{\omega=1}^{\omega_{max}-1} (c_{n,\omega+1} * c_{L,\omega+1}) \right]}{\max [n_1/m, 1] + \sum_{\omega=1}^{\omega_{max}-1} (\max [c_{n,\omega+1} * n_1/m, 1] * c_{L,\omega+1})}, \quad (25)$$

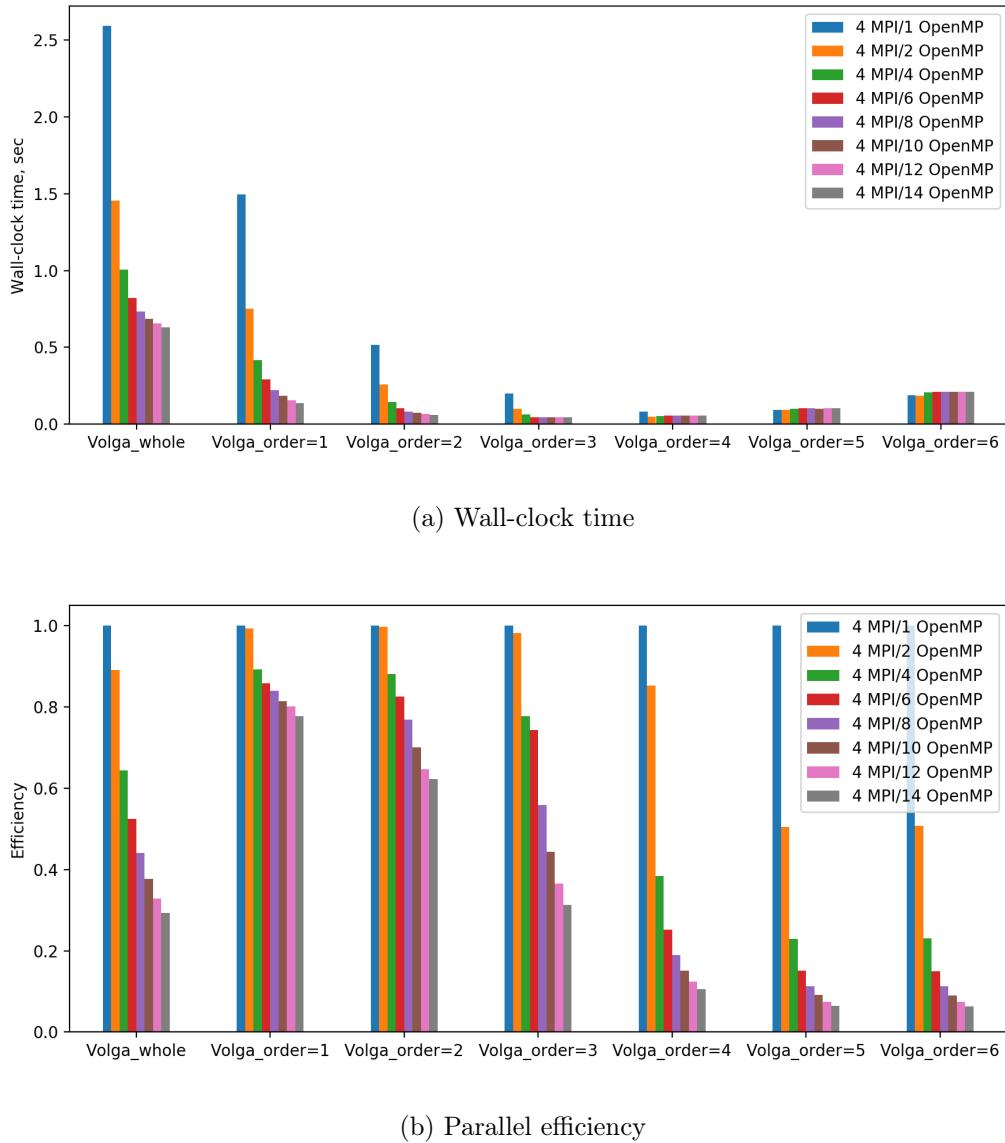


Figure 7. Wall-clock time and parallel efficiency of the river model for Volga basin using 4 MPI processes and 1–14 OpenMP threads on Lomonosov-2 supercomputer; the metrics are given for the whole basin (“Volga_whole”) and for groups of rivers having the same order (“Volga_order=<n>”) (OpenMP series)

where $c_{n,\omega} \doteq \Pi_{i=2}^{\omega} C_{n,i}^{-1}$ and $c_{L,\omega} \doteq \Pi_{i=2}^{\omega} C_{L,i}^{*k}$. Substituting to (25) the values from Tab. 1 provides $T_s/T_p = 3.28$ for 14 OpenMP nodes, which is much closer to measured acceleration 4.11 compared to that given by formula (24) with reference Horton metrics (≈ 12 times). The real acceleration appears to be faster, as the expression (25) does not take into account overheads for processing each river, related to invoking subroutines, allocating memory etc. In serial mode, these overheads increase especially the total wall-clock time for solution of model equations for small-order rivers (due to their large number), and since parallelization of these river orders is more efficient, this improves the parallel performance for the whole basin.

To summarize, processing of the Volga basin is significantly accelerated up to 4–6 OpenMP threads, this limit taking place due to serial processing of longest rivers of order 5 and 6. However, with finer meshes of land surface models, which are expected in future, the orders of resolved

large river basins are to increase and thus the efficiency OpenMP-based approach presented here is anticipated to improve, according to estimates presented in Section 3.

Conclusion

This paper presents a new version of the INM RAS-MSU land surface scheme where the river hydrodynamic and thermodynamic model is embedded into the parallel execution framework using two levels of parallelism: the first is MPI-based independent processing of river basins, and the second uses OpenMP technique to parallelize the simulation of rivers of the same Strahler order. Numerical experiments have been performed for the East European domain with resolution $0.5^\circ \times 0.5^\circ$. The MPI implementation of the soil model is based on conventional even longitude-latitude decomposition of the model domain, inherited from the atmospheric model. The soil model parallel efficiency at 1–144 cores was shown to be 0.52–0.79 and limited by the presence of ocean area, and by imbalance of computational load between soil columns depending on the presence of snow cover and number of iterations for the surface temperature needed to advance the soil profiles. The acceleration of the river model at MPI level (not exceeding 4 times) is defined by the size of the largest river basin in the domain (Volga), whereas at OpenMP level the potential for acceleration of large river basin simulation is shown to be close to number of threads used. OpenMP-level speedup was hindered in our numerical experiments by the underestimation of river orders at coarse land surface model resolution (recommended performance for the Volga basin attained at 4–6 threads with 2.5–3 times acceleration).

The future development of the parallel code includes MPI+OpenMP implementation of the soil model, optimization of MPI domain decomposition for soil model under presence of ocean surface, and further tuning the MPI+OpenMP configuration of the river model.

Acknowledgements

The work is partially supported by the Russian Ministry of Science and Higher Education (agreement No. 075-15-2019-1621, MPI implementation of the soil model), by the Russian Science Foundation (project No. 21-71-30003, MPI+OpenMP implementation of the river model), and the Russian Foundation for Basic Research (project No. 20-05-00773, configuration of numerical experiments, preparation of the input data). 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. Bell, V.A., Kay, A.L., Jones, R.G., Moore, R.J.: Development of a high resolution grid-based river flow model for use with regional climate model output. *Hydrology and Earth System Sciences* 11(1), 532–549 (2007). <https://doi.org/10.5194/hess-11-532-2007>
2. Bowring, S.P.K., Lauerwald, R., Guenet, B., et al.: ORCHIDEE MICT-LEAK (r5459), a global model for the production, transport, and transformation of dissolved organic carbon from Arctic permafrost regions – Part 1: Rationale, model description, and simulation proto-

- col. Geoscientific Model Development 12(8), 3503–3521 (2019). <https://doi.org/10.5194/gmd-12-3503-2019>
3. Bowring, S.P.K., Lauerwald, R., Guenet, B., et al.: ORCHIDEE MICT-LEAK (r5459), a global model for the production, transport, and transformation of dissolved organic carbon from Arctic permafrost regions – Part 2: Model evaluation over the Lena River basin. Geoscientific Model Development 13(2), 507–520 (2020). <https://doi.org/10.5194/gmd-13-507-2020>
 4. Brooks, R., Corey, A.: Hydraulic Properties of Porous Media. Tech. rep., Colorado State University, Fort Collins (1964)
 5. Clapp, R., Hornberger, M.: Empirical equations for some soil hydraulic properties. Water Resources Research 14(4), 601–604 (1978)
 6. Conil, S., Douville, H., Tyteca, S.: The relative influence of soil moisture and SST in climate predictability explored within ensembles of AMIP type experiments. Climate Dynamics 28(2), 125–145 (2007). <https://doi.org/10.1007/s00382-006-0172-2>
 7. Döll, P., Lehner, B.: Validation of a new global 30-min drainage direction map. Journal of Hydrology 258(1-4), 214–231 (2002). [https://doi.org/10.1016/S0022-1694\(01\)00565-0](https://doi.org/10.1016/S0022-1694(01)00565-0)
 8. Downing, J., Cole, J., Duarte, C., et al.: Global abundance and size distribution of streams and rivers. Inland Waters 2(4), 229–236 (2012). <https://doi.org/10.5268/IW-2.4.502>
 9. Du, C.: Comparison of the performance of 22 models describing soil water retention curves from saturation to oven dryness. Vadose Zone Journal 19(1) (2020). <https://doi.org/10.1002/vzj2.20072>
 10. 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 (2018). <https://doi.org/10.1515/rnam-2018-0028>
 11. Falloon, P., Betts, R., Bunton, C.: New Global River Routing Scheme in the Unified Model. Tech. rep., Hadley Centre (2007)
 12. 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 (1980). <https://doi.org/10.2136/sssaj1980.03615995004400050002x>
 13. Guth, P.L.: Drainage basin morphometry: a global snapshot from the shuttle radar topography mission. Hydrology and Earth System Sciences 15(7), 2091–2099 (2011). <https://doi.org/10.5194/hess-15-2091-2011>
 14. Huang, B., Mehta, V.M.: Influences of freshwater from major rivers on global ocean circulation and temperatures in the MIT ocean general circulation model. Advances in Atmospheric Sciences 27(3), 455–468 (2010). <https://doi.org/10.1007/s00376-009-9022-6>
 15. Lucas-Picher, P., Arora, V.K., Caya, D., Laprise, R.: Implementation of a large-scale variable velocity river flow routing algorithm in the Canadian Regional Climate Model (CRCM). Atmosphere-Ocean 41(2), 139–153 (2003). <https://doi.org/10.3137/ao.410203>

16. Lykossov, V., Palagin, E.: Dynamics of coupled heat and moisture transport in the soil-atmosphere system. *Russian Meteorology and Hydrology* 8, 48–56 (1978), (in Russian)
17. Malakhova, V., Golubeva, E.: The role of the Siberian rivers in increasing dissolved methane in the East Siberian shelf. *Bull. Nov. Comp. Center, Num. Model. in Atmosph.* 13, 43–56 (2014)
18. Medvedev, A.I., Stepanenko, V.M.: The influence of external parameters on river runoff in the INM RAS – MSU land surface model. *IOP Conference Series: Earth and Environmental Science* 611, 012023 (2020). <https://doi.org/10.1088/1755-1315/611/1/012023>
19. Miralles, D.G., Teuling, A.J., van Heerwaarden, C.C., Vilà-Guerau de Arellano, J.: Mega-heatwave temperatures due to combined soil desiccation and atmospheric heat accumulation. *Nature Geoscience* 7(5), 345–349 (2014). <https://doi.org/10.1038/ngeo2141>
20. Mualem, Y.: A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resources Research* 12(3), 513–522 (1976). <https://doi.org/10.1029/WR012i003p00513>
21. Raymond, P.A., Hartmann, J., Lauerwald, R., et al.: Global carbon dioxide emissions from inland waters. *Nature* 503(7476), 355–359 (2013). <https://doi.org/10.1038/nature12760>
22. Sheng, M., Lei, H., Jiao, Y., Yang, D.: Evaluation of the Runoff and River Routing Schemes in the Community Land Model of the Yellow River Basin. *Journal of Advances in Modeling Earth Systems* 9(8), 2993–3018 (2017). <https://doi.org/10.1002/2017MS001026>
23. Solomon, A., Heuzé, C., Rabe, B., et al.: Freshwater in the Arctic Ocean 2010–2019. *Ocean Science* 17(4), 1081–1102 (2021). <https://doi.org/10.5194/os-17-1081-2021>
24. Stepanenko, V., Medvedev, A., Korpushenkov, I., et al.: A River Routing Scheme for an Earth System Model. *Numerical Methods and Programming* 20, 396–410 (2019). <https://doi.org/10.26089/NumMet.v20r435>, (in Russian)
25. Tarboton, D.G., Bras, R.L., Rodriguez-Iturbe, I.: The fractal nature of river networks. *Water Resources Research* 24(8), 1317–1322 (1988). <https://doi.org/10.1029/WR024i008p01317>
26. 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>
27. 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 (2020). <https://doi.org/10.1134/S0001433820030123>
28. Volodina, E., Bengtsson, L., Lykosov, V.N.: Parameterization of heat and moisture transfer in a snow cover for modelling of seasonal variations of land hydrological cycle. *Russian Journal of Meteorology and Hydrology* (5), 5–14 (2000)
29. Vorobyev, S.N., Karlsson, J., Kolesnichenko, Y.Y., et al.: Fluvial carbon dioxide emission from the Lena River basin during the spring flood. *Biogeosciences* 18(17), 4919–4936 (2021). <https://doi.org/10.5194/bg-18-4919-2021>

30. Vreman, A.W.: The adjoint filter operator in large-eddy simulation of turbulent flow. *Physics of Fluids* 16(6), 2012–2022 (2004). <https://doi.org/10.1063/1.1710479>
31. Ye, A., Duan, Q., Zhan, C., et al.: Improving kinematic wave routing scheme in Community Land Model. *Hydrology Research* 44(5), 886–903 (2013). <https://doi.org/10.2166/nh.2012.145>
32. Zhang, H., Liu, J., Li, H., et al.: The Impacts of Soil Moisture Initialization on the Forecasts of Weather Research and Forecasting Model: A Case Study in Xinjiang, China. *Water* 12(7), 1892 (2020). <https://doi.org/10.3390/w12071892>

Machine Learning Approaches to Extreme Weather Events Forecast in Urban Areas: Challenges and Initial Results

Fabio Porto¹, Mariza Ferro¹, Eduardo Ogasawara², Thiago Moeda³, Claudio D. T. Barros¹, Anderson Chaves Silva¹, Rocio Zorrilla¹, Rafael S. Pereira¹, Rafaela Castro², João Victor Silva², Rebecca Salles², Augusto Fonseca², Juliana Hermsdorff⁴, Marcelo Magalhães⁵, Vitor Sá⁶, Antonio Adolfo Simões¹, Carlos Cardoso¹, Eduardo Bezerra²

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

Weather forecast services in urban areas face an increasingly hard task of alerting the population on extreme weather events. The hardness of the problem is due to the dynamics of the phenomenon, which challenges numerical weather prediction models and opens an opportunity for Machine Learning (ML) based models that may learn complex mappings between input-output from data. In this paper, we present an ongoing research project which aims at building ML predictive models for extreme precipitation forecast in urban areas, in particular in the Rio de Janeiro City. We present the techniques that we have been developing to improve rainfall prediction and extreme rainfall forecast, along with some initial experimental results. Finally, we discuss some challenges that remain to be tackled in this project.

Keywords: machine learning, rainfall forecast, extreme events.

Introduction

Precipitation nowcasting, a few hours ahead forecast for extreme rainfall events, is a relevant research topic with important impact on urban areas monitoring decision-making. In large cities, such as Rio de Janeiro, heavy precipitations events have been registered, especially during summer, causing property damage, citizens mobility disruption, and even deaths. A single extreme rainfall event, occurred in 2011 in a Rio de Janeiro nearby town, claimed the lives of more than 900 people. As the effects of climate change become stronger, more frequent episodes such as this are likely to be observed [12]. Despite the efforts dispensed to improve strong events forecast accuracy, results of the Rio Operation Center⁷ (COR), a department of the municipality responsible for throwing alerts of extreme events, need improvements. In a 2019 internal study, da Silva [10] analyzed 168 rain alerts, from February 2019 to May 2019, emitted by COR. On the total forecasts emitted by COR classified as *strong rain*, only 12% did materialize as such. More interestingly, 35% of the these alerts corresponded to actual *no rain* observed. Conversely, considering the total forecast events for *moderate rain*, 49% had *no rain* and 2% faced *heavy rain*. Thus, there is a clear need to improve on extreme weather forecasts in urban areas, and in particular for the Rio de Janeiro city. Current approach followed by COR involves the interpretation by meteorologists of the results of numerical weather predictions (NWPs), such as: COSMO [1] and the Weather Research and Forecasting (WRF) model [2]; and the follow-up on radar images, electromagnetic discharge and other meteorology related sensors. In this context,

¹National Laboratory of Scientific Computing, Petropolis, Brazil

²Federal Center for Technological Education, Rio de Janeiro, Brazil

³National Observatory, Rio de Janeiro, Brazil

⁴Sistema de Alerta Rio da Prefeitura do Rio de Janeiro, Rio de Janeiro, Brazil

⁵Fundação Instituto de Geotécnica do Município do Rio de Janeiro, Rio de Janeiro, Brazil

⁶Centro de Operações Rio, Rio de Janeiro, Brazil

⁷<http://cor.rio/>

a recent research track has explored the opportunities of applying machine learning models in the prediction of extreme weather events [11, 16, 18]. While the prediction of normal weather condition benefits from the huge history of recorded weather related observations, which can be directly used for model training, a major challenge in the extreme weather context is the small number of events. Fortunately, extreme weather events are still rare, specially in a particular region, such as the Rio de Janeiro city. As a result, few data points are available recording the data patterns of these events. Learning under such a constrained scenario has been studied under the umbrella of *learning with small data* [5, 23].

In this paper, we describe our initial contributions towards using Machine Learning (ML) models for predicting extreme weather events. We focus on a particular type of extreme event, namely, rainfall. We present three alternative approaches we have been investigating to solve this forecasting problem, along with initial validation experiments for each one of them. The first approach aim at building deep learning models that learn spatio-temporal signals obtained from precipitation observations, captured from weather stations, as well as weather forecast data computed by NWPs. The second approach leverages DL models built using the first approach but focusing on a particular spatial forecasting region of interest. Finally, the third one, proposes an extreme weather forecast dataflow.

The remainder of this paper is structured as follows. Section 1 formalizes the problem. Next, Section 2 contextualizes the problem presenting the geographic regions of interest and available data sources. In Section 3, we describe the ML approaches we have been investigating for weather forecast. Section 4 discusses experiments evaluating the presented approaches. Section 5 presents related work. Then, in Section 6, we describe the challenges we foresee in this project. Finally, we conclude highlighting some final remarks and pointing to future work.

1. Problem Statement

In this paper, we deal with the problem of precipitation forecasting. We take a machine learning approach to it. We assume the existence of several spatiotemporal data sources from which the parameters of the forecasting models can be fitted.

Informally, a spatiotemporal data source provides measurements that have stamps in both space and time. Formally, we define a spatiotemporal data source as a sequence of observations made at regular time intervals: $\tilde{X} = [X_1, X_2, \dots, X_T]$. The interval Δt between two observations defines the *temporal resolution* of the data source. Each observation $X_i \in \mathbb{R}^{H \times W \times C}$, for $i = 1, 2, \dots, T$ consists of a regular grid that determines the spatial location of interest, where H and W specify the number of horizontal and vertical elements (i.e., subdivisions) in the grid, respectively. We call each of the $H \times W$ subdivisions in the grid a *cell*. The values H and W control the *spatial resolution* of the data source, since they determine the height and width of each cell, which we denote by Δh and Δw , respectively. For each element in the grid map, C represents how many meteorological variables (e.g., precipitation, temperature, humidity) are measured simultaneously. Each data source has its particular set of observed meteorological variables.

In a machine learning-based spatiotemporal forecasting model, its parameters are fitted using a set of one or more spatiotemporal data sources as training data. Such a model can be viewed a function f that maps an input sequence of past observations $\tilde{X}_{in} = [X_{t-T_{in}+1}, \dots, X_{t-1}, X_t]$

to an output sequence of predicted observations $\tilde{X}_{out} = [\hat{X}_{t+1}, \hat{X}_{t+2}, \dots, \hat{X}_{t+T_{out}}]$ (see Eq. 1). The lengths of these sequences (denoted by T_{out} and T_{in}) may differ. However, while $C_{in} \geq 1$ (i.e., several meteorological variables can be used as predictors), $C_{out} = 1$ (since there is only one target variable, namely, precipitation).

$$\hat{X}_{t+1}, \hat{X}_{t+2}, \dots, \hat{X}_{t+T_{out}} = f(X_{t-T_{in}+1}, \dots, X_{t-1}, X_t) \quad (1)$$

If it is the case that more than one spatiotemporal data source is available, we assume that, as a preprocessing step, these data sources are conciliated with relation to spatial and temporal resolutions to form an aggregated data source in which each element in a grid map contains the union of all predictor variables in the component data sources.

The problem of precipitation forecasting is very challenging, and several factors contribute to this. For example, some of the data sources may have missing data, some elements in the grid maps may not be observable, conciliation of different spatial and/or temporal resolutions may be needed, inherent sparseness of precipitation data, just to mention a few.

2. Available Data Sources

As we discuss in Section 5, a number of ML based models for extreme events have been proposed lately. A common understanding is that extreme events nowcasting considers temporal extrapolation of trends derived from several types of sensors informing about local weather conditions, such as radar, satellite, meteorological stations, among others [33]. Indeed, while investigating approaches to extreme rainfall events in the city of Rio de Janeiro, we are confronted with a set of data sources that can contextualize each extreme event, compensating for the scarcity of samples and for resulting data imbalance during training.

In Fig. 1, we depict the geographical area of interest for our work. This region was delimited with the help of meteorologists that make part of our team. The total corresponding area is $84.23 \text{ Km} \times 48.42 \text{ Km}$. The upper left and lower right coordinates of the rectangle corresponding to the region are (lat = -23.1339033365138 , lon = -43.8906028271505) and (lat = -22.649724748272934 , long = -43.04835145732227), respectively.

In this figure, one can find markers for the location of the 33 weather stations, monitored by COR. These stations provide online real-time information about weather conditions in the city. Most of these stations are only pluviometric (i.e., measure only precipitation), while seven of them are meteorological (i.e., besides precipitation, they also gather data on temperature, humidity, wind and pressure). The temporal resolution of these stations is fifteen minutes.

By analysing the small dots and triangles marked in Fig. 1, we can also observe a great imbalance in the distribution of these stations throughout the city. Moreover, the considered area contains a chain of mountains that crosses the city with significant coverage by the tropical forest, which brings humidity to the city and, at the same time, offers obstacles to the dynamics of weather systems. Rio de Janeiro is a coastal city, and is under the influence of SACZ (South Atlantic Convergence Zone), two factors that add to the complex climatic scenario observed in the city.

Most of the weather stations available in our study have been collecting data since 1997 (with some interruptions due to maintenance events). Table 1 presents summary information about the distribution of rainfall (precipitation) severity levels, considering the time series of

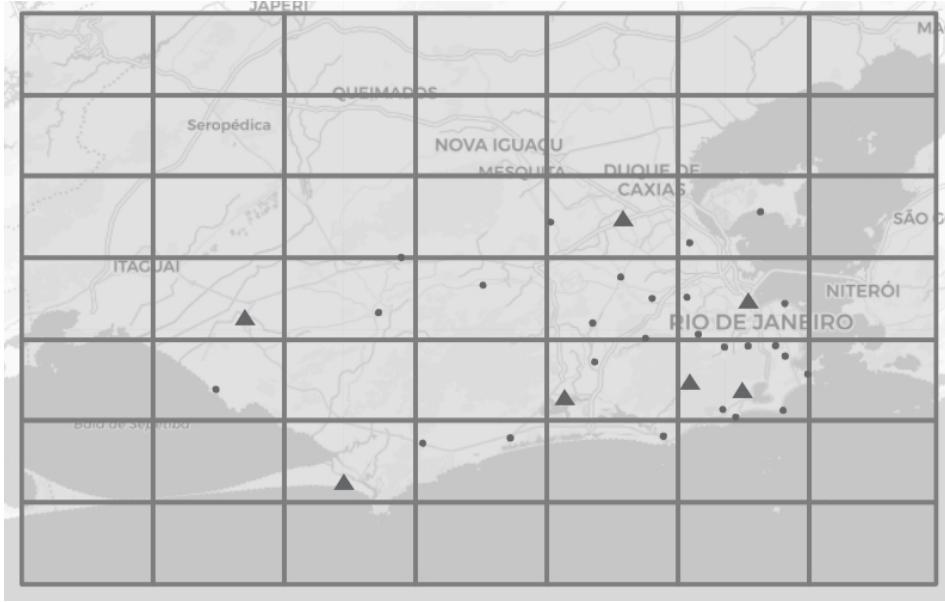


Figure 1. Spatial region of interest for our study in the Rio de Janeiro city. The light grey area correspond to land. The dark grey area correspond to sea. Small black dots mark the location of pluviometric stations. Black triangles mark the location of meteorological stations

all thirty three stations in the period from 2018-01-01 to 2021-12-31. These levels were defined by a group of meteorologists at AlertaRio, an organization linked to the city's municipality and responsible for providing weather bulletins to the population. One can notice that this data source corresponds to a very unbalanced dataset. Specifically, only 0.17% of the observations are considered extreme events, according to the AlertaRio's severity level classification scheme. However, the occurrence of these kind of events has historically caused great negative impact to the metropolitan area of Rio de Janeiro, and their precise forecasting would be of paramount importance to the city.

Table 1. Rain rate statistics of data provided by the 33 weather stations monitored by COR in the period from 2018-01-01 to 2021-12-31

Rain Rate (mm/h)	Proportion (%)	Severity Level
$0 \leq x < 5$	97.31	No / Light
$5 \leq x < 25$	2.19	Moderate
$25 \leq x < 50$	0.33	Heavy
$x \geq 50$	0.17	Rainstorm

In addition to the weather stations, there is a list of other complementary data sources that we plan to use to train ML models for extreme precipitation forecast. In Tab. 2, we depict the different data sources we aim at using to contextualize extreme weather events, and their corresponding responsible organization. Below, we list additional relevant information about these other data sources.

Table 2. Available data sources and their providers

Simulations from meteorological models	IAG ⁸
Fluviometric stations	INEA ⁹
Radars	INEA, AlertaRio ¹⁰
Weather buoys	Brazilian Navy ¹¹ , Portal SIMCosta ¹²
Electromagnetic activity	Furnas ¹³ , SOS Raios ¹⁴
Pluviometric/meteorological stations	GeoRio
Satellite data	INMET ¹⁵
Weather balloon	Tom Jobim Airport

- *Simulations from meteorological models.* We have available data from simulations produced by NWP systems, namely, COSMO and GFS25. The geographical area filtered from the two datasets covers the Rio de Janeiro city and was fixed at Latitude $[-19.5, -24.5]$ and Longitude $[-40.5, -45.5]$. The time interval was set to the period from January 2015 to March 2021. From the data obtained in the spatio-temporal region defined above, we computed a new more fine grained grid dataset. The latter includes a grid of 300×300 meters within the limits of the area defined by the original filtered dataset. The values for each new grid point are the fruit of an interpolation computing an average among the three nearest points in the filtered dataset inversely weighted by their distance to the new point.
- *Fluviometric stations.* INEA provides twenty three fluviometric stations that record river levels every 15 minutes. This data has been collected since 2008.
- *Radar data.* INEA has two meteorological radars, both with a radius do 250 Km covering the Rio de Janeiro area and their surroundings. Their coordinates are (Lat: $-22^{\circ}59'35.81077''S$, Lon: $-43^{\circ}35'16.65427''W$) and (Lat: $-22^{\circ}24'20.99917''S$, Lon: $-41^{\circ}51'37.65632''W$). This data is being collects since 2015. Each one of them produces data every five minutes.
- *Weather buoys.* These sensors collect variables such as dew point, water temperature, pressure, humidity, wind speed and direction. New data is available every one hour. This data source is available since 2019.
- *Satellite data.* We also collect electromagnetic activity data from the GOES satellite through Amazon API. This data source is available since 2018. This data source produces a data file every twenty seconds.
- *Weather baloon.* Every 6 hours, a weather balloon (filled with hydrogen or helium gas) equipped with a radiosonde raises to the atmosphere from the Tom Jobim International airport, located in Rio de Janeiro. Its goal is to collect meteorological data (humidity, temperature, air pressure) at different altitudes. This balloon reaches an altitude of 25 Km. While the balloon is suspended, its radiosonde sends the data via radio to a ground station.

⁸<https://www.iag.usp.br/>⁹<https://alertadecheias.inea.rj.gov.br/>¹⁰<http://alertario.rio.rj.gov.br/>¹¹<https://www.marinha.mil.br/>¹²<https://simcosta.furg.br/>¹³<https://www.furnas.com.br/>¹⁴<https://detectorderaios.com.br/>¹⁵<https://portal.inmet.gov.br/>

In addition to the available data sources describing real-time weather condition, COR meteorologists provided us with information regarding previous extreme rainfall events, with the dates and time of their occurrence.

3. ML Approaches for Precipitation Forecast

To face the challenges involving extreme rainfall prediction in urban areas using ML models, we count with our previous experiences in building spatiotemporal ML models for weather predictions. Our expectation is that we can build on these experiences to tackle the particularities of extreme weather events. In particular, we are currently investigating three approaches to produce ML models for precipitation forecast. In this section, we briefly describe each of these approaches. In Section 4, we conduct some initial experiments we have run to validate these approaches.

3.1. Regression Approach

We start by describing the first approach, which considers the target variable as a continuous one. Hence, in this approach we are trying to fit regression models. We investigate the application of two deep learning architectures in this approach, namely, YConvLSTM [34] and STConvS2S [7]. Both models, which are results of recent investigations carried out by the authors, are suited to capture spatial and temporal features by design and automatically make a connection between temporal and spatial features. As a result, these models can capture complex dependencies, both intra- (time only or space only) and inter-dimensional (time and space) patterns. In this section, we provide a general view of these two architectures. In Section 4, we describe how these generic schemes are instantiated to be used with the data available in our initial experiments.

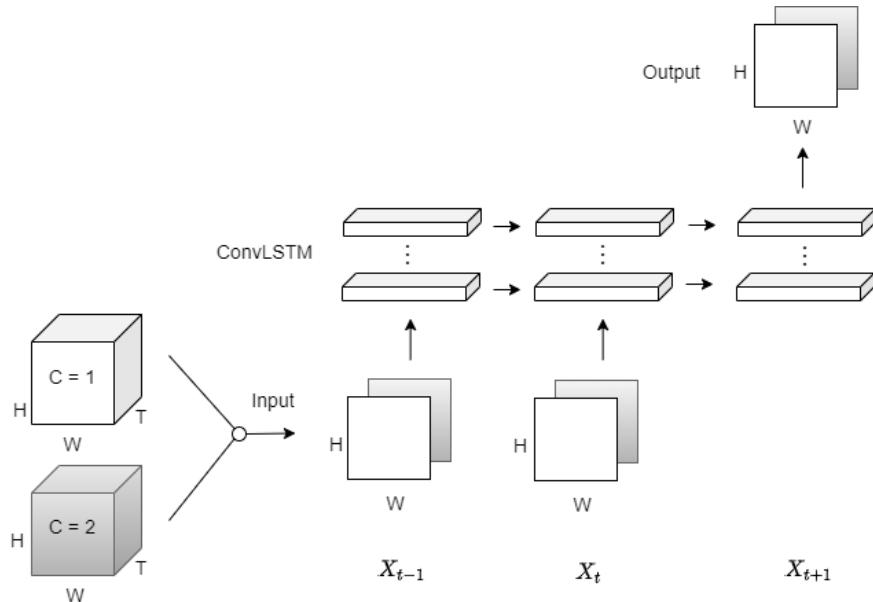


Figure 2. YConvLSTM architecture uses ConvLSTM layers to generate $X_{t+T_{out}}$ predictions that are a combination of C components, with dimension $H \times W$, applied as a channel in the convolutional layer. In this work C represents several NWP models used as input

Figure 2 depicts the YConvLSTM architecture. The latter is an adaptation of the CONVLSTM model [30], which was proposed to improve rainfall nowcasting. The main novelty introduced by the YConvLSTM deep learning model is to adopt a learning process that combines predictions from multiple numerical weather prediction models. In YCONVLSTM, the different predictions produced by the component numerical models are conciliated into a single spatial and temporal grid. Thus, at each point of the grid and time instant, a list of prediction values produced by the NWP models is associated. The slicing of the 3D grid at each time-instants produces frames that are input to the deep learning model. The list of prediction values associated at each 2D frame is mapped to the channel structure expected by the convolution input frame. The prediction process is qualified as auto-regressive, as it reads a single variable type as input (i.e. precipitation volume) and produces future predictions of the same variable type.

This can be seen as a multivariate regression model, since the variable of interest is predicted by a combination of predictors.

Figure 3 depicts the architecture of the STConvS2S model. STConvS2S [7] is a sequence-to-sequence architecture comprised exclusively of convolutional layers. Convolutions are performed with factorized 3D kernels. The architecture is comprised of three component blocks, each one in turn is a sequence of factorized convolutional layers. These blocks are applied sequentially to a given input. The first component is the *temporal block*, responsible for extracting temporal features through the temporal kernels. The temporal block uses causal convolutions to maintain temporal coherency during prediction. The second component is the *spatial block*, responsible for receiving the output of the previous block and extracting spatial features through the spatial kernels. Following the spatial block is the *temporal generator block*, designed to increase the sequence length T_{in} if the task requires a longer predictive horizon, where $T_{out} \geq T_{in}$. Finally, the output of this last block is further fed into a final convolutional layer to complete the prediction. Different from YConvLSTM, STConvS2S is an autoregression model, because the target variable is predicted based on its own past states.

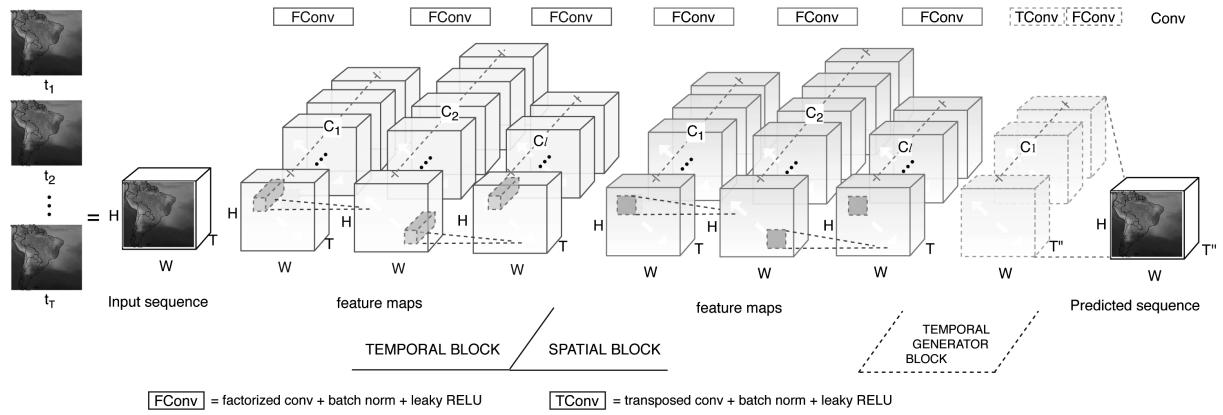


Figure 3. STConvS2S architecture is composed only of 3D CNNs, which are divided into three blocks to predict an output with temporal size T_{out} . The first block learns a temporal representation of the input sequence followed by a block that extracts only spatial features. This architecture can uses the channel in the first convolutional layer to combine several spatiotemporal inputs with dimension $H \times W \times T$. Source: [7]

3.2. DJEnsemble Approach

Another initiative developed within the DEXL Lab¹⁶ builds on the traditional ensemble of models to achieve improved local weather forecasts. We start by taking into account the well known “No-Free-Lunch Theorem” [29, 36], which, informally, states that *no learner can succeed on all learning tasks*. A corollary to this theorem is that for a given problem there will be a region of the data space where an optimal model trained with a sampled distribution of the domain will produce predictions that are sub-optimal. We built on this result and introduced a data pre-processing step in ML model selection for weather prediction [24].

3.2.1. Data pre-processing

This pre-processing step aims at identifying regions of the data space that share similar data patterns. We consider the spatial domain of interest discretized in a set of point locations, for instance the positions of weather sensors, and model the observations at each point as time-series.

Thus, let us consider a discretized spatiotemporal domain $\mathcal{D} = \{d_1, d_2, \dots, d_n\}$, where $d_i = \{p_i = (x, y), s_i\}, 1 \leq i \leq n$, $p = (x, y)$ is a spatial location (longitude, latitude) and $s_i = <v_1, v_2, \dots, v_k>$ is a time-series of observations. We accounted for seasonality effects on time-series of weather observations by splitting them into annual periods. Thus, each time-series corresponds to the observations of one location during one year. We partition the domain \mathcal{D} into disjoint sub-domains $\mathcal{S}, S \subseteq \mathcal{D}$. Such partitioning is obtained through the clustering of the set of time-series s_i at each domain point $p_i \in D, 1 \leq i \leq n$. The *k-means* clustering algorithm was used to compute the disjoint partitioning of the domain. In order to compute the distance between time-series, we adopted the *Dynamic Time Warping (DTW)* algorithm [27]. Thus, after clustering, each time-series s_i is annotated with the cluster-id it was associated to. The choice involving the number (ie. k) of partitions for a given time-series dataset is obtained by computing partitions for different values of k and using the *Silhouette* score [26] to rank the resulting clustering. The *Silhouette* score is obtained by first computing a per time-series mean internal clustering DTW distance (a) divided by the mean extra-cluster DTW distance, for the nearest extra-cluster (b). The *Silhouette* score is obtained by computing the mean over all per time-series scores. Finally, in order to define rectangular spatial areas that conform to the expected frame structure of data inputs to Deep Learning (convolutional) models, we structured the domain time-series into disjoint rectangular tiles, $T = \{t_1, t_2, \dots, t_t\}$. In this setting, each time-series $s_i \in D$ is associated to a single tile $t_j \in T$. Moreover, tiles are constructed such that a maximum ϵ of variation in different clusters is observed. Thus, given $s_i \in t_j$, the set of time-series associated to a tile t_j , we define $tx = \arg \max_k \sum(x_i, x_i = 1 \text{ if } s_i \in c_k)$ as the maximum number of time-series associated to the same cluster in tile t_j , with c_k identifying a cluster. Then, the tiling algorithm assures the constraint for all tiles $t_j \in T, 1 \leq j \leq t$: $\frac{tx}{|t_j|} \leq \epsilon$, where $|t_j|$ is the number of time-series in tile t_j . Therefore, at the end of the pre-processing stage, we have a tiled spatiotemporal domain, where each tile exhibits similar data behaviour corresponding to its weather history. As a final step in data pre-processing, we compute for each tile t_i its time-series representative, s_r . The representative in a tile t_i is a time-series $s_r \in S$ whose average *DTW* distance, $dtw(s_r, s_j)$, between s_r and all time-series $s_j \in t_i$ is the smallest. Finally, we expect that the structuring of spatiotemporal domain into tiles to provide a more precise learning experience such that a model optimized to a tile is less prompt to suffer from the effect of the *non-free-lunch* theorem when applied to tiles with similar data distribution.

¹⁶<http://dexl.lncc.br>

3.2.2. Model selection

Once the spatiotemporal domain \mathcal{D} has been structured into tiles, the DJEnsemble approach is ready to compute predictions. We consider spatiotemporal predictive queries $Q = \{R, M\}$, where $R \subseteq \mathcal{D}$ is a spatiotemporal region where weather predictions are to be computed and $M = \{m_1, m_2, \dots, m_m\}$ is a set of pre-trained candidate ML models. Each m_j model has been trained in a subset \mathcal{S}_j of the domain \mathcal{D} . Therefore, for each m_i model, there is a set of tiles $\{t_1, \dots, t_k\}$ that were used for the model m_i training. We can interpret that as building specialized ML models for a region of particular weather characteristics. Thus, we could have a model built on time-series of the south region of the Rio de Janeiro city, which is a coastal area between the sea and the mountains, and another model built on the north of the city where weather tends to be hotter and less influenced by the sea humidity. The DJEnsemble approach works by finding for each predictive query Q the set of models in $M' \subseteq M$ whose training data, induced by the data distribution of tiles used for their training, most closely approximates the data distribution of the tiles covered by the query region $Q.R$. This is obtained by comparing the distances between time-series representatives among the query tiles and the model training tiles. Observe that the set of candidate models M may not have been trained on data from query region $Q.R$. Nevertheless, DJEnsemble is still able to indicate the best candidate model whose data distribution most closely approximates that of the query region. Finally, we want to highlight that the improvements on extreme weather events precision forecasts may involve the ability to use specialized models whose training capture nuances of the weather behavior of a region. We expect that the ideas induced by the DJEnsemble may contribute to a more precise ML based extreme weather forecast.

3.3. Classification Approach

In this approach, we frame the problem of forecasting extreme rainfall events as a binary classification task. It considers extreme precipitation as an anomaly on the time-series of rainfall observations. We consider a particular type of spatiotemporal data source, namely, data coming from a set of meteorological stations, each one of them having collected observations about multiple weather variables over a given period of time, one of them necessarily being precipitation. We also assume the times series coming from these meteorological stations all have the same temporal resolution. For the particular data source we used, this temporal resolution was fifteen minutes.

3.3.1. Construction of the labelled dataset

As a first step in this approach, we had to build a classification dataset, in which the target variable is binary: the value 1 indicates the occurrence of an extreme precipitation, and the value 0 indicates otherwise. Hence, we had to map continuous values of precipitation to discrete (binary) values. For this, we built a semi-automatic instance labelling system. In particular, we used a particular type neural network, the Self-organizing Map (SOM) [19]. SOM neural networks are models based on the functioning of neurons in the human brain's cortex, in which learning happens through a process of competition among its neurons. During training, a multidimensional data point is mapped to a 2D topological map. The latter suggests a visual interpretation of the input data, where discrepant points can be highlighted [20]. To define a

SOM architecture, several hyperparameters must be defined, such as dimensions and shape of the topological map, and the neighborhood function.

To train the SOM model, we concatenated all the time series coming from the meteorological stations. As a result, a multivariate time series was built containing observations from all stations. Notice that, when building this dataset, we completely disregarded the spatial locations of the meteorological stations. The rationale for that is that we wanted the SOM model to capture patterns of anomaly (with respect to the precipitation variable) irrespective of the spatial location.

During training, the data instances were processed in incremental windows of 96 instances, each window corresponding to one day of measurements of the meteorological stations. When processing the set of windows, say w_i , the model also receives the information provided by the previous map on w_{i-1} , in order to form the map solution set S .

As a last activity in this step, the solution set of generated maps S is visually analyzed to assist in labelling the instances. The descriptive method we established for the visual analysis of SOM maps to detect a shape anomaly consists of interpreting the amplitudes of the displacement rates of the distances between the spatial components x , y and z in relation to the sequence of maps $s_i \in S$. A shape anomaly is detected and annotated in Y if there is a discrepant amplitude of the component z , correlated to (x, y) , whose displacement varies in a discrepant way in relation to the adjacent maps s_{i-1} and s_{i+1} , where (x, y) represents the topographic ordered pair in the map and the component z the topographical error.

3.3.2. Assessment of the labelled dataset

The visual interpretation of the SOM model in the first step was helpful because it assisted in the labelling of observations as either normal or anomalous. The goal of this second step was to assess the quality of the labelling conducted in the first step. For this, we fitted a decision tree (DT) model to the labelled dataset.

A DT is a symbolic classification method applied in this work as a tool to assess the quality of the annotations of detected events. In the structure of a DT, each data instance follows a unique path for its classification, depending on a set of comparison rules, of the *If-Then* type, which are performed in each node of the tree, determining whether the result should proceed left or right, thus building the DT [31]. The DT learning algorithm splits the training example into subsets, represented in its leaves, in which points falling in the same leave are explained by a conjunction of predicates over independent features of the dataset.

In our scenario, the dependent variable is the label indicating the value status as normal or anomalous, whereas the independent variables are the features describing each time instant: precipitation, humidity, etc. If we consider that the independent variable observe a certain state when anomalous phenomenon occurs, which differs from the state of a normal status, then we can expect leaves comprising a single value. Therefore, we propose to compute the entropy of a DT classification. DTs with lower entropy would indicate that leaves share similar values and the annotations of anomaly would tend to be correct. Thus, as a criterion for evaluating labelling step, we established that the more accurate the adjustment of the DT algorithm, the better would be the qualitative labeling process of anomaly detection. If the results obtained after fitting the DT model were not satisfactory, the process is restarted with a new hyperparameter configuration for training the SOM model. Otherwise, we proceed to the third (and last) step to fit a classification model to the previously labelled dataset.

3.3.3. Model fitting through LSTMs

In this last step, the goal was to fit the binary classification model to predict the label (either extreme/anomalous or normal) for a given instance. For this, we used a Long Short-term Memory (LSTM) neural network, due to the capacity of its units to learn long sequential patterns of data behavior. The LSTM architecture is a type of recurrent neural network, which are designed to analyze the behavior of data sequences over time. As stated in [13], an LSTM model addresses the problem of vanishing gradients, incorporating functions (*gates*) into its state dynamics to maintain or discard information. The original LSTM formulation features three gates: *input*, *forget* and *output*.

After model fitting, the model is evaluated. If the obtained results are not satisfactory, the process is restarted from step 1 (dataset labelling). Otherwise, the resulting classification model is ready to be used for inference.

3.4. Discussion

The YConvLSTM model [34] implements a post-processing ensemble approach on the output of NWPs. It builds on the architecture proposed by Shi et al. [30] extending it to cope with the input coming from n NWP simulators. We also discussed STConvS2S, originally presented in [7]. The model comprises a autoregressive CONV2D+1 architecture suited for spatiotemporal prediction. Although these two models were able to minimize rainfall prediction error with respect to individual NWPs, they were not designed for the forecasting of extreme weather events. In fact, as discussed in Ding et al. [11], deep learning models tend to either *underfit* or *overfit* on extreme weather predictions, due to the imbalance on training datasets. Thus, as an initial attempt to close this gap, we presented a dataflow of machine learning models, combining *Self Organized Maps (SOM)*, *Decision Tree* and *LSTM*, specifically designed for extreme precipitation detection. The task is modeled as a binary classification where extreme events are annotated to the observational data. Finally, we presented the *DJEnsemble* approach that considers a spatiotemporal ensemble of ML models.

4. Experiments

In this section we report initial experiments with all approaches described in Section 3. These experiments aim at highlighting the potential of the different approaches towards the challenges in extreme weather events forecast.

4.1. Validating the Deep Learning Approaches

In this section, we present some preliminary experiments we conducted to train and validate out two Deep Learning models, YConvLSTM and STConvS2S (see Section 3). In these experiments, we trained both models using the same dataset. This dataset contains precipitation simulation data obtained from two numerical models, COSMO and GFS25. The temporal range of the observations is from January 2018 to April 2021. The data are arranged in a 7×7 grid covering the city of Rio de Janeiro (see Fig. 1). The time distance between each grid (i.e., the temporal resolution) is 3 hours.

For training the YConvLSTM and STConvS2S models, we set the following learning task: we use the previous ten observations (at timesteps $t - 9, t - 8, \dots, t$) to predict the next observation

(at timestep $t + 1$). We use as predictors the outputs of the above mentioned NWP models. Our target variable is precipitation. Furthermore, we used data in the temporal range between years 2018 and 2019. Data from 2020 were used for validation. Finally, we set data collected in 2021 for testing their generalization performance, which we estimated using Mean Squared Error (MSE). The sizes (i.e., number of observations) of the training, validation and test sets are 5840, 2928, and 849, respectively.

The hardware infrastructure used in training both models has the following hardware settings: CPU Intel(R) Core(TM) i7-7740X CPU @ 4.30GHz; GPU: GeForce GTX 1080 Ti 11GB; RAM: 32GB.

We conducted 10 runs of training using the STConvS2S architecture. For these ten runs, we got an average training time of approximately 14 min, with a standard deviation of 3.91 seconds. Also for these 10 runs, the resulting average MSE (on the test dataset) was 2.0052, with a standard deviation of 0.2145. Figure 4 presents the learning curves resulting from training the STConvS2S model in one of the runs. The hyperparameters used to fit the STConvS2S model were the following: learning rate = 5e-6; optimizer: RMSProp, with alpha set to 0.99; eps: 1e-6; weight decay set to 1e-1. We trained the model for 200 epochs, and used a batch size of 64. The resulting MSE on the test set was 1.802.

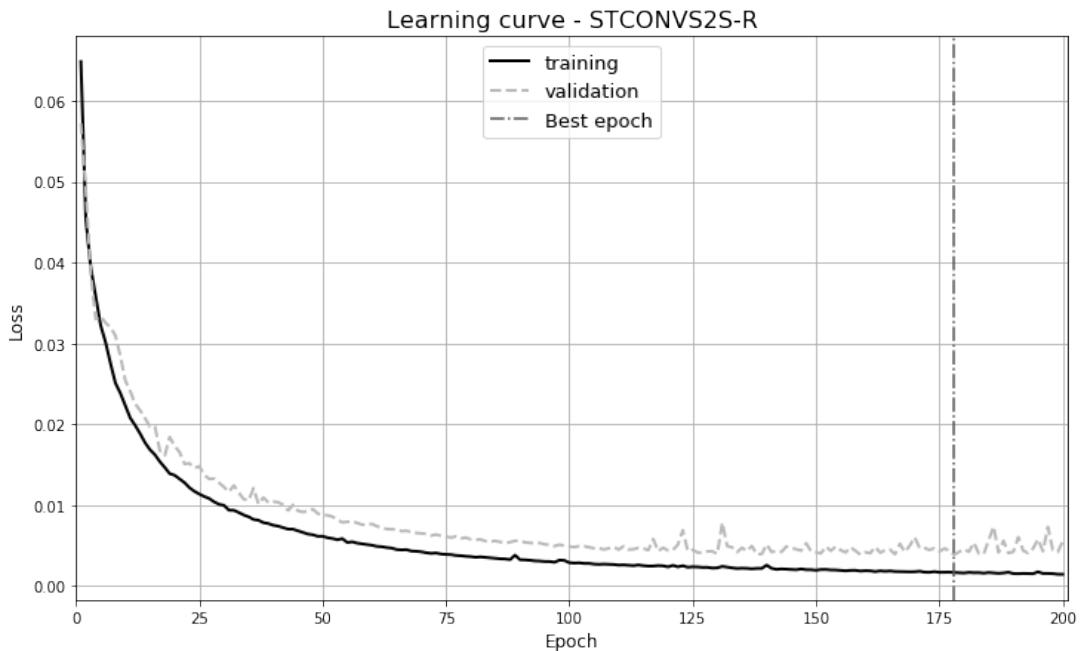


Figure 4. Learning curves obtained during one of the runs of STConvS2s model training. The losses are MSE values measured over the rainfall domain. Hence their unit is *squared millimeters* (mm^2) of rain over the specified period. The vertical dashed line indicates the epoch in which the best model (measured in the validation set) was obtained

To assess the difficulty of detecting extreme events, we manually picked ten events between January and April of 2021 that were considered extreme in our test dataset. We wanted to know if the model was capable of making good predictions of these picked up extreme events. We analysed two scenarios. We first considered all the spatial observations of these ten extreme events that presented the highest precipitation. The resulting MSE in this case was 22.20. In the second scenario, we considered the ten largest precipitation values in the same time period. In this case, the MSE value increased to 174.88. As a conclusion, the model error considering

only extreme events was much higher when compared to the average MSE value measured in the whole test set. We attribute this difference in predictive quality to the sparseness of extreme events, one of the many challenges we face in this project (see Section 6 for further discussion on this subject).

In a second validating experiment, we trained the YConvLSTM model using the same hardware infrastructure described above. The model was trained for 200 epochs with a batch size of 64, as its training time is significantly higher than the STConvS2S. The architecture consists of two ConvLSTM layers with 32 and 64 filters each and was optimized using the Adam optimizer and Keras default hyperparameters. We achieved the loss of 1.773 MSE on the testing set. The average training time was 50 minutes. Figure 5 presents the learning curves resulting from training the YConvLSTM model in one of the runs.

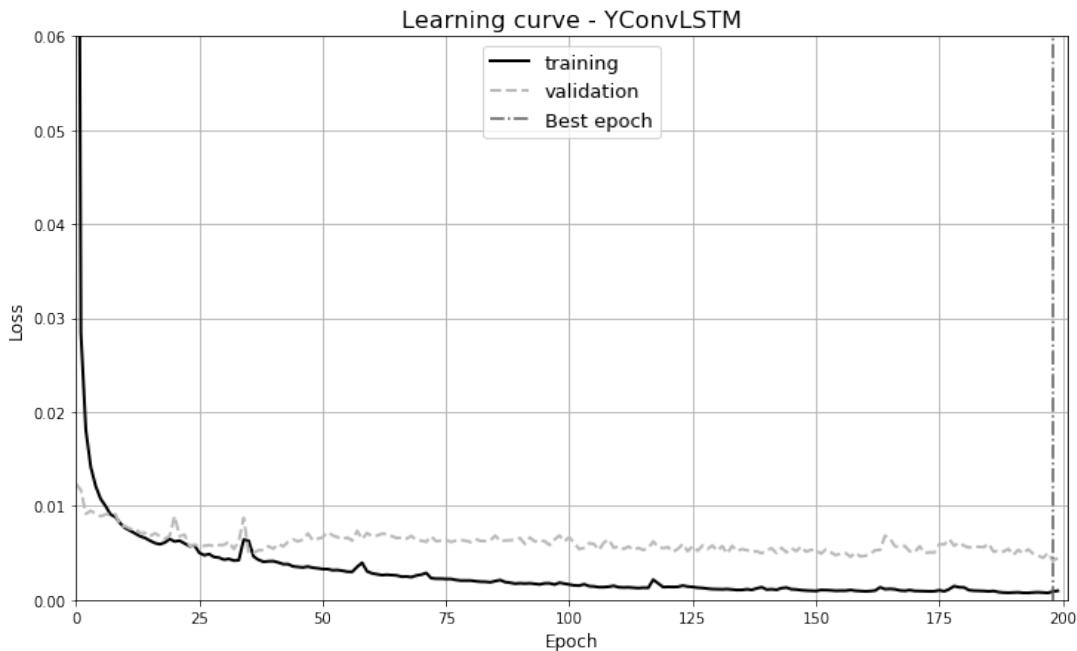


Figure 5. Learning curves obtained during one of the runs of YConvLSTM model training. The losses are MSE values measured over the rainfall domain. Hence their unit is *squared millimeters* (mm^2) of rain over the specified period. The vertical dashed line indicates the epoch in which the best model (measured in the validation set) was obtained

We also conducted the same experiment regarding extreme events on the predictive model produced by the YConvLSTM architecture (we used the same manually picked 10 extreme events in our test dataset). In the first scenario, the resulting MSE was 100.23, and in the second scenario the value increased to 534.18. As already happened with STConvS2S, the YConvLSTM model also had much difficulty to correctly predict extreme events.

4.2. Validating the DJEnsemble Approach

In this section, we depict an experiment comparing the DJEnsemble approach against other ensemble types. In the traditional ensemble approach, each available model is run over the entire query region and the prediction results are linear combinations of each prediction. In the stacking ensemble approach, a model is trained with data produced by a set of component models. The new model can be a linear regression learning algorithm, for example. The predictions are

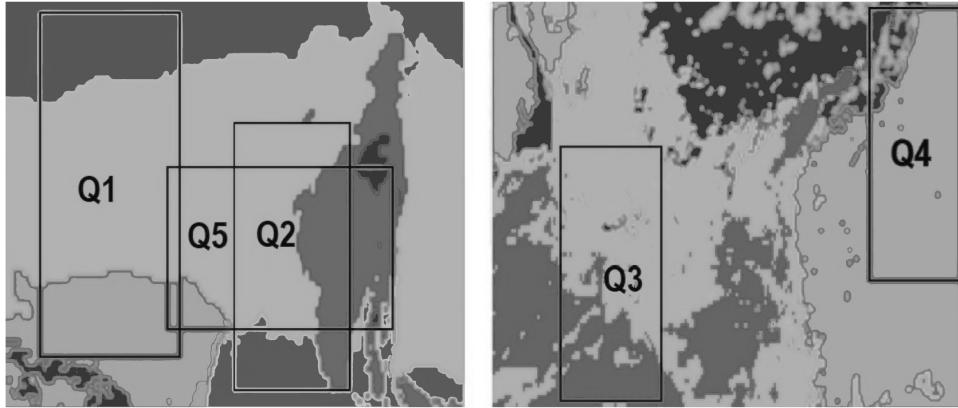


Figure 6. Queries over the temperature (left) and rainfall (right) domains

obtained by invoking the stacking model (ie. the predictor) on the outcome of the component models' predictions.

The data used in the experiments is a subset of the Climate Forecast System Reanalysis (CFSR) dataset that contains air temperature observations from January 1979 to December 2015 covering the space between 8N-54S latitude and 80W-25W longitude (temperature dataset) [22]. Additionally, we use a subset of the rainfall dataset from NASA's TRMM and GPM missions, with rainfall collected for the same spatial region as in the CSFR dataset over 22 years (rainfall dataset) [14]. The inclusion of experiments with temperature and precipitation aim to assess the approach prediction quality under different spatio-temporal patterns. We present the results of five queries executed on these data. Temperature is considered by meteorologists as an easy variable to model as opposed to rain, which is considered one of the most complex variables to predict. The temperature is homogeneously distributed in large spatial regions, while the rainfall is more heterogeneous and concentrated in small areas. Based on these, we can argue that we define queries on data with different behavior (i.e. data distribution). Figure 6 depicts the regions corresponding to every query. Each color represents a region with different data distribution (i.e. cluster). Table 3 summarizes the results for a traditional ensemble, a stacking ensemble, and DJEnsemble. DJEnsemble achieves the best accuracy for all the queries. The gap between DJEnsemble and the other ensembles is as much as a factor of 9.

Table 3. RMSE over the temperature (Q_1, Q_2 and Q_5) and precipitation (Q_3 and Q_4) domains. Precipitation values represent an estimate for the volume of rainfall, in millimeters, in a global 0.1° spatial grid and half hourly intervals. The temperature values correspond to a resolution of 6 hours intervals and a spatial grid of 0.5° , in Celsius

Query	R[lat, lon]	Trad. ensemble	Stacking	DJEnsemble
Q1	[70:130, 95:140]	25.01	7.28	3.35
Q2	[60:110, 40:80]	27.88	5.52	4.29
Q3	[125:175, 25:90]	14.28	13.96	5.34
Q4	[0:40, 60:130]	8.80	12.94	6.04
Q5	[59:100, 25:100]	29.10	5.04	3.18

These results, originally presented in [24], are reproduced here to illustrate the potential of the DJEnsemble approach towards more precise data driven ML model predictions of meteorological variables.

4.3. Validating the Classification Approach

In this section, we present an experimental evaluation of the method introduced in Section 3.2.2. The hardware configuration used in these experiments is the following: CPU: 280 × Intel(R) Xeon(R) Gold 5220 CPU @ 2.20GHz; RAM: 512GB.

We used data from the seven meteorological stations (see Fig. 1). Specifically, we considered the periodic measurements of three variables, namely *precipitation*, *temperature* and *humidity*. We selected a subset of these time series covering the period from December 1st, 2015 to March 17th, 2021. To prepare the data, some pre-processing steps were required. Initially, we applied data normalization and interpolation of missing values through the mean value. In the results reported in this study, there was no treatment for removing noise in the data, as this operation had a negative influence on the dataset labelling step (Section 3.3.1). The normalization of the input data for an interval between [0, 1] was obtained through the method *MinMaxScaler* [28]. Working with data at the same scale is a good practice when it comes to distance calculation algorithms. It also speeds up the optimization process for generating models via gradient-based learning algorithms, as in the case in SOM neural networks.

In the first step (see Section 3.3.1), we started by training the SOM model. For this, 100.000 iterations for training were defined, the learning rate starting at 0.6 with the radius of the training area starting at 0.5. Then, the visual analysis of the generated maps and the subsequent annotation of the instances were performed on the data set.

After the first step, the resulting labelled dataset was subdivided into six disjoint subsets (A1 to A6). Table 4 summarizes these subsets.

Table 4. Information about Meteorological datasets

Subset	Number of Instances	Period
A1	38.112	01/12/2015 a 31/12/2016
A2	35.040	01/01/2017 a 31/12/2017
A3	35.040	01/01/2018 a 31/12/2018
A4	35.040	01/01/2019 a 31/12/2019
A5	35.136	01/01/2020 a 31/12/2020
A6	7.296	01/01/2021 a 17/03/2021

To exemplify the way in which the maps produced by the SOM model were visually analyzed to pinpoint extreme precipitation events, the SOM output for subset **A6** is presented in Fig. 7. (However, this process was applied to all subsets.). The amplitudes of the component *z* correspond to the similarity errors of the instances in the accommodation in the node in the topographic mesh of the produced map. High amplitudes represent extreme events. Figure 8, in addition to the *z* component, the *x* and *y* components correspond to the topographic coordinates of the map. Vertical lines in red correspond to the beginning of the annotation of detected anomalies. The correlation between the three components represents the similarities, as well as the breaking of the similarity in relation to the input space. Thus, these correlations between

the peaks of the z component and the sudden and prolonged change in the x and y components were interpreted as anomalies of form.

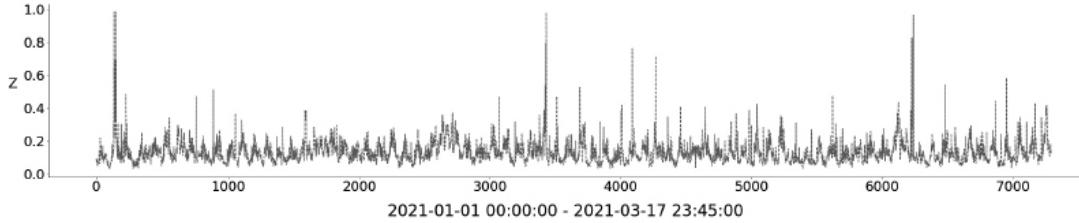


Figure 7. z z -component produced by SOM for subset **A6**

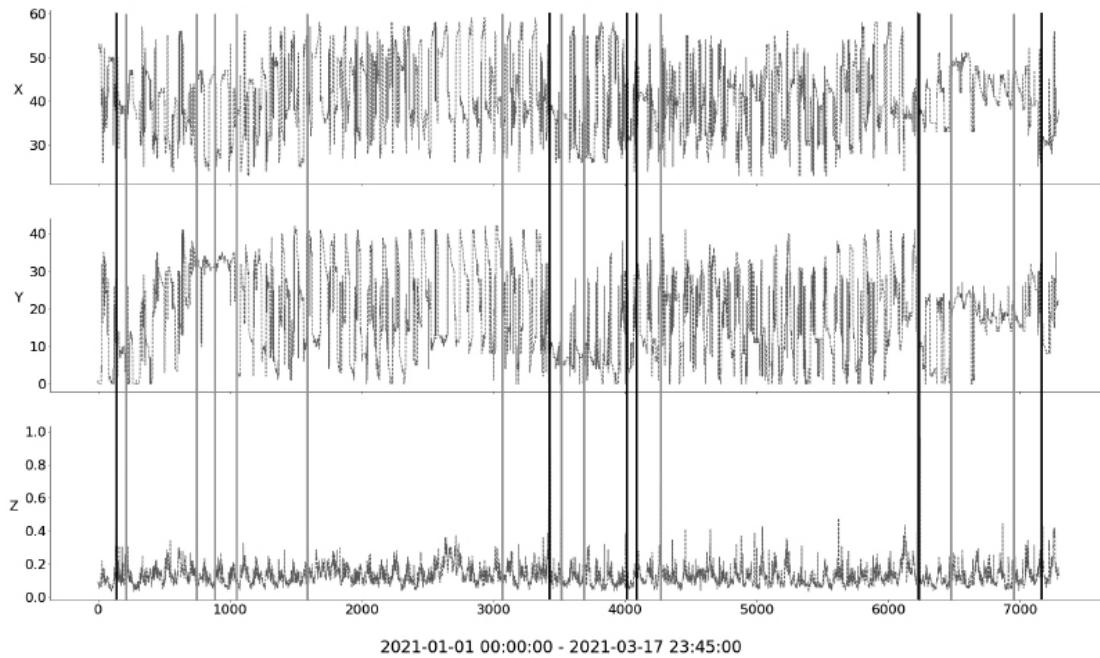


Figure 8. z components x , y , and z produced by the SOM for the subset **A6**. Vertical lines in black correspond to major shape anomalies detected and vertical lines in light gray correspond to minor anomalies

To validate the accuracy of the classification performed in the previous step, seeking to reduce subjectivity inherent to the visual analysis, a DT algorithm was applied and the result evaluated for each subset. The annotated dataset was divided into 70% for training and 30% for testing. We evaluated the labels annotated in the datasets by analyzing the test of the models fitted with such data, using the cross validation, ROC curve, and F1-score metrics. Obtained results are shown in Tab. 5.

Table 5. Summary of decision tree results for the entire dataset

Dataset	ROC	Precision	Sensitivity	F1-score	Cross Validation (10x)
A1 to A6	0.8976	0.98/0.85	0.99/0.81	0.98/0.83	0.9700 (+/- 0.01)

Finally, the step described in Section 3.3.3 was carried out to generate the binary classification model for predicting the occurrence of extreme events. An LSTM neural network model was trained for each subset. Similar to what we did in the second step, the data instances in each subset were also split in 70% used for training and 30% for testing. An *Embedding* layer was used in order to make the data streams continuous and dense.

Due to the imbalance of binary labels, experiments were performed with different balance proportions using the Synthetic Minority Over-sampling Technique (SMOTE) class from the imbalanced-learn library [15]. As mentioned by the documentation of this library, SMOTE and its variations work best if done in combination with some major class subsampling method. Thus, the applied methods of oversampling and undersampling are applied consecutively through a pipeline on the data provided by SMOTE and the random method *RandomUnderSampler*. Thus, all LSTM models were trained, tested and evaluated using the SMOTE methodology, according to the aforementioned strategy.

The network architecture was defined with a single hidden layer with 21 LSTM units. The *Adam* optimizer with a learning rate of 0.001 was used. The number of training epochs was set to 100, and the batch size equal to 32. The loss function used was the *binary_crossentropy*. The value of 252 was determined for the length of the string that returns to the model. In order to reduce *overfitting*, one *Dropout* layer with value of 0.3 were used. Finally, an output layer of just one unit was defined with the *sigmoid* activation function providing the result as a probability value. Each classification model trained with one given subset was validated against the other subsets.

We built ROC curves to evaluate the classification performance of the models fitted with each subset. Figure 9 illustrates the results obtained for the model trained on subset **A6** (we present just one curve for lack of space).

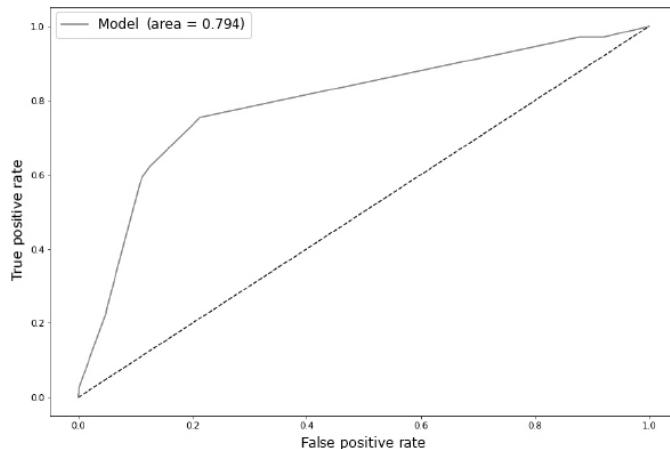


Figure 9. ROC plot of trained model with **A6** dataset

Comparisons between classification performances are shown in Tab. 6. Each model was trained with a given subset and then applied to the other subsets for inference. The first column corresponds to the models generated by the subset in parentheses. The best results are highlighted in **bold**. We considered the union of the subsets as a sample of the complete data domain. Under this assumption, and considering the proportion to the total of samples at each subset, we computed the standard deviation (StdDev) for the F_1 score for each model. The standard deviation average for all models is 0.07764.

Table 6. Summary of LSTM model validation results. F_1 -score metric

Model/Data	A1	A2	A3	A4	A5	A6	StdDev
m(A1)	–	0.4659	0.6779	0.4476	0.6152	0.6590	0.0441
m(A2)	0.5877	–	0.6797	0.4490	0.6141	0.6598	0.0367
m(A3)	0.4116	0.0964	–	0.2919	0.6110	0.6571	0.0945
m(A4)	0.3924	0.4620	0.6811	–	0.1102	0.0903	0.1021
m(A5)	0.5898	0.4697	0.6796	0.4479	–	0.6607	0.0433
m(A6)	0.6267	0.0025	0.0125	0.7743	0.5609	–	0.1450

5. Related Work

Nowcasting deals with the problem of short-term forecasting of extreme weather events. These events may include the determination of the future track of a particularly severe storm for warning purposes, the estimation of the amount of additional precipitation that will fall in a given area in the next few hours, severe wind, winter precipitation types, etc. In addition, various types of meteorological data could be used to forecast severe weather events, including radar measurements, satellite data, and weather stations' observations [35]. This section presents a literature review of recent solutions for weather nowcasting applied on different source of data. The works presented here focus on Artificial Intelligence, mainly in Machine Learning based solutions.

Ravuri et al. [25] developed a Deep Generative Model of rainfall (DGMR) for radar's probabilistic nowcasting of precipitation. Their nowcasting algorithm is a conditional generative model that predicts N future radar fields given M past radar fields using radar-based estimates of surface precipitation at a given time point. Learning is framed in the algorithmic framework of a conditional generative adversarial network (GAN), which is specialized for the precipitation prediction problem and is driven by two loss functions. These functions guide the parameter adjustment by comparing real radar observations to those generated by the model. The first loss is defined by a spatial discriminator, a CNN that aims to distinguish individual observed radar fields from generated fields. The second loss is defined by a temporal discriminator, a 3D CNN that aims to distinguish observed and generated radar sequences. The model is trained on a large corpus of precipitation events, 256×256 crops extracted from the radar stream, of length 110 min (22 frames). All models are trained on radar observations for the UK from 2016–2018 and evaluated on a test set from 2019. DGMR performance ranked first for its accuracy and usefulness in 89% of cases against two state-of-art methods (PySTEPS and Unet). According to the authors, their model produces realistic and spatiotemporally consistent predictions over regions up to $1,536 \text{ km} \times 1,280 \text{ km}$ and with lead times from 5–90 min ahead. Despite the accuracy and operational utility, the prediction of heavy precipitation at long lead times remains difficult for all approaches.

The work of [8] proposes a classifier named RadRAR (Radar products values prediction using Relational Association Rules) for convective storms nowcasting based on radar data. In addition, RadRAR is intended to prove that relational association rule mining applied on radar data helps discriminate between severe and normal weather conditions. RadRAR is trained on radar data collected from normal weather conditions and learns to predict whether the radar echo values will be higher than 35dBZ, indicating the occurrence of a storm. Experiments are

conducted on real radar data provided by the Romanian National Meteorological Administration for the 5th of June 2017. It was a day with moderate atmospheric instability manifested through thunderstorms accompanied by heavy rain and medium-size hail. Analyzing the experimental results and the comparison to existing approaches, the authors conclude that the RadRAR is effective for predicting if the radar echo values are higher than 35dBZ, obtaining performances better than the results from the literature and a Critical Success Index of 61%.

Agrawal et al. [3] present an application of DL to the problem of predicting the instantaneous rate of precipitation one hour into the future from radar data with high-resolution ($1\text{ km} \times 1\text{ km}$). More specifically, the model uses the ubiquitous U-Net Convolutional Neural Network (CNN). Dataset is from multi-radar multi-sensor (MRMS), removing non-meteorological artifacts and projects the combined observations onto a rectangular grid. Forecasting is treated as an image-to-image translation problem where is given a sequence of n input radar images that start at some point of time, t_{in1} , and end at t_{inn} . The task is to generate the radar image at some point in the future, t_{out} . Before training the model, data is transformed, labeling images by quantifying precipitation rates into four discrete ranges, based on three thresholds of millimeters of rain per hour. This step provides three binary classifications that indicate whether the rate exceeds thresholds that roughly correspond to trace rain, light rain, and moderate rain. In sequence, they partition the US into $256\text{ km} \times 256\text{ km}$ tiles and independently make predictions for each tile. Model is trained on data collected in 2018 and tested on data for 2017 and 2019. Results show their model performs better than MRMS persistence, optical flow (OF) and, the High Resolution Rapid Refresh (HRRR) system (the current best operational NWP available from NOAA) one-hour forecast methods. However, once the prediction window is increased to approximately 5 hours, the HRRR models consistently outperform their approach.

Sønderby et al. [32] introduce MetNet, a Neural Weather Model (NWM) that forecasts rates of precipitation with a lead time of up to 8 hours at the high spatial resolution ($1\text{ km} \times 1\text{ Km}$) and at the temporal resolution of 2 minutes using radar images and spectral bands of satellite as input data. MetNet covers a $7000 \times 2500\text{ km}$ geographical area corresponding to the continental United States. MetNet relies on mosaicked ground based radar and satellite imagery as input and the predictions take in the order of seconds independently of lead time and can be done in parallel. They cast precipitation forecasting as a structured prediction problem where the output comes in the form of a three-dimensional tensor. Each value of the tensor corresponds to a time and a location and indicates the corresponding rate of precipitation measured in mm/h. Target precipitation rates are estimated by the MRMS ground based radars as a function of the returned radar echoes. MetNet architecture combines a CNN, which encodes the time slices sampled every 15 minutes with a Convolutional LSTM to processes the time slices in the direction of time. Results show the performance of MetNet at various precipitation thresholds and find that MetNet outperforms HRRR Numerical Weather Prediction at forecasts of up to 7 to 8 hours on the scale of the continental United States.

The work [9] explores SOM to detect patterns on real radar data from Romania National Meteorological Administration. The main goal is to provide better insight regarding how the values of weather radar products are evolving in time, both in calm and severe weather conditions, with the broader goal of using these findings for weather nowcasting. In addition, they investigated if SOM can distinguish severe weather conditions from radar data. The exported raw data collected through the radar scans during one day (24 h) on a particular geographic region is provided as a sequence of dimensional matrices. A matrix corresponds to a specific

timestamp t and a meteorological product. For each time moment t , a sequence of matrices (3D data grid) is available, containing the values for various radar products at time t . Two data sets were constructed for representing the radar data collected during the timestamps on a day. One dataset uses the entire set of meteorological products provided by the radar (i.e., 24). Another one employs only 13 products: base reflectivity of particles on six elevations, velocity on six elevations, and the estimated quantity of water contained by a one square meter column of air. To detect the underlying structure of the datasets, the SOM model is applied to obtain an unsupervised two-dimensional representation of the datasets. The corresponding U-matrices are analyzed and compared for assessing the relevance of the meteorological products used in detecting the timestamps when a certain meteorological event occurred. Through this methodology, SOM detects temporal intervals where certain meteorological phenomena occur. The U-matrix visualization for the SOM built using the proposed data model shows readable changes in the meteorological products almost 2 hours before the event, which might help forecast the start of the phenomena. In general, they conclude that there is a slow change in the values over time, except when certain severe phenomena occur.

RainNet [4] introduces a deep neural network that aims at learning representations of spatiotemporal precipitation field movement and evolution from radar data to provide skillful precipitation nowcasts. RainNet follows an encoder-decoder architecture, as U-Net [3]. The encoder progressively downscales the spatial resolution using pooling, followed by convolutional layers. The decoder progressively upscales the learned patterns to a higher spatial resolution using upsampling, followed by convolutional layers. RainNet was trained to predict precipitation at a lead time of 5 min, using several years of weather radar. Dataset covers Germany with a spatial domain of 900 km \times 900 km and has a spatiotemporal resolution of 1 km in space and 5 min in time, respectively. Verification experiments were carried out on 11 summer precipitation events from 2016 to 2017. Since RainNet uses a convolutional architecture and does not use LSTM layers to propagate information through time, it was only trained to predict precipitation at 5 min lead times. So, a recursive approach was implemented using RainNet predictions to achieve a lead time of 60 min. Results showed that RainNet competes against a conventional nowcasting model based on optical flow. This work confirms [3] that CNNs provide a firm basis for competing with conventional nowcasting models based on optical flow.

Song et al. [33] present an Artificial Neural Network (ANN) nowcasting model for hourly summer precipitation over the Eastern Alps. The ANN model, a Multilayer Perceptron (MLP), is developed for the nowcasting of hourly precipitation for a one-hour lead time. Several predictors from diverse sources are used as input, including QPE, QPF, three INCA convective analysis data (CAPE, CIN, MCONV), and three radar data composite measurements of five C-band radars (MAXCAPPi, ECHOTOP, VIL). The performance of the MLP model is compared with the traditional INCA extrapolation technique and a multiple linear regression approach (MLR) model. The available period for both radar products and raw INCA forecasts is the summer period of 2012–2014 (June to September) yielding 12 months in total. The dataset is divided into the training set, containing ten randomly chosen months, and the verification set, containing the remaining two months. The independent verification months, August 2013 and July 2014, are used to evaluate the performance of the proposed nowcasting methodology. Precipitation case studies were analyzed, and the results showed that both the MLR and MLP models could predict the precipitation similarly to the INCA extrapolated forecast. However, the forecast by MLP model is better than INCA, providing more detailed small-scale structures. Also, the

precipitation intensity of the MLP model forecast is closer to gridded precipitation observations than the INCA.

The study presented in [6] uses video prediction deep learning (VPDL) algorithms applied in precipitation nowcasting for São Paulo, Brazil. They use the PredRNN++ as a VPDL model to predict reflectivity images and precipitation edges from weather radar images for up to 1-h lead time and compare the results with ENCAST, an extrapolation-based model used for precipitation nowcasting. PredRNN++ utilizes causal LSTM to capture complex dependencies and variations and gradient highway unit (GHU) to keep the gradients during training. This study uses a dual-polarization S-band (SPOL) Doppler weather radar from the Department of Water and Electric Energy, manufacture Selex ES GmbH, installed at Ponta Nova, State of São Paulo, Brazil. These radar measurements are restricted to 100 km of range for the period of March 2015 to December 2019 with a time interval of 5 min. The results have some limitations, and the authors propose several improvements; despite this, they advocate that VPDL model has good potential as an additional tool to assist nowcasting.

Jiang, Ma and Wu [37] main purpose is to propose nowcasting precipitation prediction at target station within 1–2 hours in the future using the radar historical map sequence in the past 1.5 hours. Also, a solution to deal with missing data in this kind of dataset. For this purpose, they train a CNN and two auxiliary ML models (NN and gradient boosting decision tree – GBDT) using different features from the real Hubei dataset. This dataset comprises radar reflectivity maps, precipitation and rainfall time recorded by ground monitoring stations, and stations' altitude, latitude, and longitude. For training are used 2300 samples and the remaining 200 samples for evaluation. Results showed that the precipitation nowcasting mechanism was suitable for radar reflectivity data with single or cumulative altitudes. The models showed better performance than traditional optical flow methods, and the CNN model performed best compared with the other 2 ML models. Finally, the authors conclude that their way of dealing with missing data on the experiments does not have a large negative impact on prediction performance.

6. Challenging Aspects

In this section, we present some of the challenges we are facing in the current project, with emphasis on our geographic region of interest, the Rio de Janeiro municipality. For some of them, we also present the lines of investigation we plan to follow.

- *Missing data.* An important line of research in this project has to do with an investigation of the procedure for handling missing data. As an example, we have the case of simulation data from meteorological models. In this case, not all models present data for the entire time series. It would be interesting to investigate how incomplete data for certain criteria can be used to improve training done with data from periods where there is complete observation.
- *Relief influence.* One challenging aspect that was raised by the specialists is that forecast of meteorological variables in the city of Rio de Janeiro is directly influenced by the geography of the location. Hence, we plan to investigate how information about the municipality's relief can be used to increase the predictive power of forecasting models.
- *Data sparsity.* Another complicating factor found concerns the sparseness of observed data related to a combination. We actually empirically observed the result of this data property in our preliminary experiments with Deep Learning models (see Section 4.1). A possible

solution to this problem involves an investigation of data subsampling and augmentation techniques. Another envisioned solution is the training of prediction models using Physics Informed Machine Learning [17]. This approach allows the training of models to take into account restricted the evolution of the underlying physical phenomena, as a way of compensating for the sparseness of the observed data.

- *Spatially non-uniformity of observations.* Another interesting aspect worthy of scientific investigation concerns the fact that observations from various data sources planned to be used in model training are collected in regions of space that do not form a regular grid. For example, the 33 meteorological stations under AlertaRio's responsibility are distributed in such a way that most of them are located in the eastern region of the municipality (see Fig. 1). Another example concerns observations from metoceanographic buoys. This is a complicating factor when considering the use of classical convolutional neural network architectures, which assume that training data is represented as regular lattices. Possible solutions are the use of neural network architectures for graphs, such as Graph Neural Networks [21].
- *Adding other data sources.* During the first phase of the project, the team identified several data sources potentially relevant for tuning the prediction models (see Tab. 2). It is natural to think that there must be other sources not yet considered that might be equally relevant. We have already mentioned the issue of relief before. Other data sources that are on the team's investigation horizon are related to aerosols, COR cameras located in key positions of the city, and radar images all potentially useful at the *nowcasting* prediction level.
- *Consideration of the dynamics of events.* At this point of investigation, we are interested in evaluating how we can capture the fine dynamics of events in space-time. We consider that this real-time dynamic must interact with the predictions, correcting them. As an example, we have the effect of side winds influencing the occurrence of convective rain and impacting its location and timing. Other elements are the so-called meteorological phenomena, such as Trough¹⁷, and how to consider them in the learning process.

Conclusion

Extreme weather events have become stronger and more frequent worldwide. The concentration of the population in large urban areas places the occurrence of extreme weather events, such as strong rain, as a risk for the population. Governments have been investing in apparatus to monitor weather sensors and in organizations responsible for alerting the population and taking measures ahead of the extreme event. In this context, this paper describes an initiative we have been developing with the Rio de Janeiro municipality in building ML models to forecast extreme rainfall events a few hours ahead. Our initial investigations show that building ML models to predict extreme events is a very challenging problem, due to their scarcity of occurrences and fast dynamics.

As for the computational experiments, we presented initial results in this paper. We plan to do a more thorough experimental comparison using stronger baselines in future work. However, our initial computational experiments have led us to believe that, to achieve more precise forecasts, we will need to combine multiple data modalities. Hence, we plan to enrich the data we learn from with other available weather sensors we allude to in Section 2. We also envision the need of need fundamental ML research to cope with the challenges aspects of the problem we pointed out in Section 6.

¹⁷An elongated geographic region presenting relatively low atmospheric pressure.

Acknowledgements

The authors are funded by CNPq productivity research fellowship. The authors thank the Centro de Operações Rio, GeoRio, and INEA for providing us meteorological expertise and the several data sources we investigate in our project. The authors also thank professor Pedro Dias and professor Demerval Moreira, both from IAG, for providing us with NWP simulation data used in our experiments. Finally, experiments were run in the Petrus cluster at DEXL/LNCC.

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. <https://www.cosmo-model.org/>, accessed: 2021-09-20
2. <https://www.mmm.ucar.edu/weather-research-and-forecasting-model>, accessed: 2021-09-20
3. Agrawal, S., Barrington, L., Bromberg, C., et al.: Machine learning for precipitation nowcasting from radar images. CoRR abs/1912.12132 (2019), <http://arxiv.org/abs/1912.12132>
4. Ayzel, G., Scheffer, T., Heistermann, M.: Rainnet v1.0: a convolutional neural network for radar-based precipitation nowcasting. Geoscientific Model Development 13(6), 2631–2644 (2020). <https://doi.org/10.5194/gmd-13-2631-2020>
5. Bendre, N., Marn, H.T., Najafirad, P.: Learning from few samples: A survey (2020). <https://doi.org/10.48550/ARXIV.2007.15484>
6. Bonnet, S.M., Evsukoff, A., Morales Rodriguez, C.A.: Precipitation Nowcasting with Weather Radar Images and Deep Learning in São Paulo, Brasil. Atmosphere 11(11) (2020). <https://doi.org/10.3390/atmos1111157>
7. Castro, R., Souto, Y.M., Ogasawara, E., Porto, F., Bezerra, E.: STConvS2S: Spatiotemporal Convolutional Sequence to Sequence Network for weather forecasting. Neurocomputing 426, 285–298 (2021). <https://doi.org/10.1016/j.neucom.2020.09.060>
8. Czibula, G., Mihai, A., Czibula, I.: RadRAR: A relational association rule mining approach for nowcasting based on predicting radar products values. Procedia Computer Science 176, 300–309 (2020). <https://doi.org/10.1016/j.procs.2020.08.032>
9. Czibula, G., Mihai, A., Mihuleț, E., Teodorovici, D.: Using Self-Organizing Maps for Unsupervised Analysis of Radar Data for Nowcasting Purposes. Procedia Comput. Sci. 159(C), 48–57 (2019). <https://doi.org/10.1016/j.procs.2019.09.159>
10. daSilva, F.: Projeto pesquisa operacional (2019), internal Report, in PT
11. Ding, D., Zhang, M., Pan, X., et al.: Modeling extreme events in time series prediction. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1114–1122. KDD '19, ACM, New York, NY, USA (2019). <https://doi.org/10.1145/3292500.3330896>

12. Gates, B.: How to avoid a Climate Disaster: The Solutions We Have and the Breakthroughs We Need. Random House Large Print Publishing (2021)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Computation* pp. 52–65 (1997)
14. Huffman, G., Bolvin, D., Braithwaite, D., et al.: NASA Global Precipitation Measurement (GPM) Integrated Multi-satellitE Retrievals for GPM (IMERG) v5.2. NASA (2014)
15. Imbalanced_Learn: The imbalanced-learn (2021), <https://imbalanced-learn.org/stable/>
16. Jaye, A., Bruyère, C.L., Done, J.M.: Understanding future changes in tropical cyclogenesis using Self-Organizing Maps. *Weather and climate extremes* 26, 100235 (2019)
17. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., et al.: Physics-informed machine learning. *Nature Reviews Physics* 3(6), 422–440 (2021). <https://doi.org/10.1038/s42254-021-00314-5>
18. Kim, S., Kim, H., Lee, J., et al.: Deep-Hurricane-Tracker: Tracking and Forecasting Extreme Climate Events. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1761–1769 (2019). <https://doi.org/10.1109/WACV.2019.00192>
19. Kohonen, T.: The self-organizing map. *Proceedings of the IEEE* 78(9), 1464–1480 (1990). <https://doi.org/10.1109/5.58325>
20. Kohonen, T.: Essentials of the self-organizing map. *Elsevier - Neural Networks* 37, 52–65 (2013). <https://doi.org/10.1016/j.neunet.2012.09.018>
21. Liu, Z., Zhou, J.: Introduction to Graph Neural Networks. Morgan & Claypool (2020)
22. NCAR: NCEP Climate Forecast System Reanalysis (CFSR) 6-hourly Products, January 1979 to December 2010 (2010). <https://doi.org/10.5065/D69K487>
23. Pereira, R.: Strategies and techniques for deep learning on small data. Ph.D. thesis, National Laboratory of Scientific Computing (2020)
24. Pereira, R., Souto, Y., Chaves, A., et al.: DJEnsemble: A Cost-Based Selection and Allocation of a Disjoint Ensemble of Spatio-Temporal Models, pp. 226–231. ACM, New York, NY, USA (2021). <https://doi.org/10.1145/3468791.3468806>
25. Ravuri, S., Lenc, K., Willson, M., et al.: Skilful precipitation nowcasting using deep generative models of radar. *Nature* 597(7878), 672–677 (2021). <https://doi.org/10.1038/s41586-021-03854-z>
26. Rousseeuw, P.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* 20, 53–65 (1987), <https://wis.kuleuven.be/stat/robust/papers/publications-1987/rousseeuw-silhouettes-jcam-scienceopenarchiv.pdf>
27. Sakoe, Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26(1), 43–46 (1978)

28. Scikit_Learn: Scikit-learn: Machine Learning in Python (2011), <https://scikit-learn.org/stable/index.html>
29. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press (2017)
30. Shi, X., Chen, Z., Wang, H., et al.: Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. pp. 802–810. NIPS’15, MIT Press, Cambridge, MA, USA (2015)
31. Skiena, S.S.: The Data Science Design Manual, vol. 1. Springer, New York (2017)
32. Sønderby, C.K., Espeholt, L., Heek, J., et al.: MetNet: A Neural Weather Model for Precipitation Forecasting. CoRR abs/2003.12140 (2020), <https://arxiv.org/abs/2003.12140>
33. Song, L., Schicker, I., Papazek, P., et al.: Machine Learning Approach to Summer Precipitation Nowcasting over the Eastern Alps. Meteorologische Zeitschrift 29(4), 289–305 (2020). <https://doi.org/10.1127/metz/2019/0977>
34. Souto, Y.M., Porto, F., Moura, A.M., Bezerra, E.: A Spatiotemporal Ensemble Approach to Rainfall Forecasting. In: Proceedings of the International Joint Conference on Neural Networks. pp. 574–581 (2018). <https://doi.org/10.1109/IJCNN.2018.8489693>
35. Wang, Y., Coning, E., Harou, A., et al.: Guidelines for Nowcasting Techniques (2017)
36. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997). <https://doi.org/10.1109/4235.585893>
37. Xiang, Y., Ma, J., Wu, X.: A precipitation nowcasting mechanism for real-world data based on machine learning. Mathematical Problems in Engineering 2020, 1–11 (2020). <https://doi.org/10.1155/2020/8408931>

Data Assimilation by Neural Network for Ocean Circulation: Parallel Implementation

Haroldo F. Campos Velho¹, Helaine C. M. Furtado²,
Sabrina B. M. Sambatti³, Carla O. F. Barros⁴, Maria E. S. Welter⁴,
Roberto P. Souto⁴, Diego Carvalho⁵, Douglas O. Cardoso^{6,7}

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

Data assimilation (DA) is an essential issue for operational prediction centers, where a computer code is applied to simulate physical phenomena by solving differential equations. The procedure to determine the best initial condition combining data from observation and previous forecasting (*background*) is carried out by a data assimilation method. The Kalman filter (KF) is a technique for data assimilation, but it is computationally expensive. An approach to reduce the computational effort for DA is to emulate the KF by a neural network. The multi-layer perceptron neural network (MLP-NN) is employed to emulate the Kalman in a 2D ocean circulation model, and algorithmic complexity to KF and NN is presented. A shallow-water system models the ocean dynamics. Synthetic measurements are used for evaluating the MLP-NN for the data assimilation process. Here, a parallel version for the DA procedure by the neural network is described and tested, showing the performance improvement for a parallel version of the NN-DA.

Keywords: data assimilation, artificial neural network, shallow water equations, parallel processing.

Introduction

An essential step for the prediction centers is identifying the better set of initial conditions for each prediction period. Several issues are involved for producing good predictions from a mathematical model, starting with calculation of the best possible initial condition. In this context, the analysis typically employs data assimilation (DA) procedure, which combines observational data with the previous prediction (*background*) of a numerical model to obtain an optimal estimate of the evolving system state. Weather services were the first centers to employ schemes for data assimilation, using several methods such as optimal interpolation, Kalman filter, variational approaches, and particle filter [8, 11, 15].

The mentioned methods for DA are computing-intensive [11], and one alternative to reduce computer processing time is to adopt a neural network (NN) formulation to emulate or replace parts or the entire method. Here, the artificial neural network architecture based on *multi-layer perceptron* (MLP) formulation will be employed. The MLP is a supervised fully-connected neural network, requiring a training algorithm to compute interconnecting weights. The back-propagation algorithm is applied for training the MLP-NN as a surrogate for the Kalman filter for the DA process.

The literature has registered applications of neural networks for DA in forecasting systems in different areas of geophysics: meteorology [6, 7], hydrology modeling [3], and space weather application [4]. Here, the DA by the neural network is applied to the shallow-water equation

¹National Institute for Space Research, São José dos Campos, Brazil

²Federal University of Western Pará, Santarém, Brazil

³Independent researcher, São José dos Campos, Brazil

⁴National Laboratory for Scientific Computing, Petrópolis, Brazil

⁵Federal Center for Technological Education Celso Suckow da Fonseca, Rio de Janeiro, Brazil

⁶Federal Center for Technological Education Celso Suckow da Fonseca, Petrópolis, Brazil

⁷Polytechnic Institute of Tomar, Tomar, Portugal

(SWE) designed to represent ocean circulation dynamics as described in Bennett's book [2]. The shallow water system can be used for different applications in geophysical fluid dynamics, the dimension describing the domain and mainly parameters associated to the simulation are used to determine a good representation of this system for the application expressed in the numerical experiment. Next section will be to describe the SWE configured to the ocean circulation [2].

The supervised MLP-NN is designed to emulate the Kalman filter for DA – see also references [9, 16] for 2D shallow water system and reference [4] for space weather applications. Besides, in this paper, a parallel version of the DA procedure is presented aiming to enhance the computational performance for neural data assimilation. In some previous results, the MLP-NN was used to emulate KF, focusing on data assimilation. The algorithmic complexity of these two schemes is included in this paper.

The following section describes the mathematical model used as a prediction system. Data assimilation methods – Kalman filter and neural network – are presented in Section 2. The strategy for parallel implementation is commented on in Section 3. Results for speed-up and efficiency are shown in Section 4. The last section offers conclusions and final remarks.

1. Shallow-Water Equations as a Model for Ocean Circulation

The shallow-water approach works for fluid dynamics simulation, assuming that the vertical dimension is much smaller than the horizontal dimension. Integrating the NavierStokes equations on vertical coordinate, a 2D system of partial differential equations (PDE) is derived. The latter PDE system is called *shallow water equations* (SWE). The SWE can be solved by applying a numerical algorithm, and it is able to model the ocean circulation dynamics. Here, the same SWE presented in the Bennett's book [2] is employed. The 2D SWE is described with fluid depth (H), coupled to a velocity field (u, v). The three independent variables (u, v, q) are under influence of gravitational force (g) acting on the fluid. Mathematical equations for the worked model are given by:

$$\frac{\partial u}{\partial t} - fv + g \frac{\partial q}{\partial x} + r_u u = F_u , \quad (1)$$

$$\frac{\partial v}{\partial t} + fu + g \frac{\partial q}{\partial y} + r_v v = F_v , \quad (2)$$

$$\frac{\partial q}{\partial t} + H \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + r_q q = 0 , \quad (3)$$

where $t > 0$ and $(x, y) \in (0, L_x) \times (0, L_y)$, the Coriolis parameter is denoted by f , the damping coefficients are represented by (r_u, r_v, r_q) – these coefficients are linked with the linearization process of the nonlinear terms, (u, v) is the velocity field, F_u and F_v are external forcing, H is the mean depth of the ocean, and the free perturbed ocean surface is defined by q . Boundary conditions are shown in Fig. 1.

The forcing terms are expressed as:

$$\begin{aligned} F_u &= -C_d \rho_a u_a^2 / (H \rho_w) , \\ F_v &= 0 , \end{aligned} \quad (4)$$

where C_d is the drag coefficient, ρ_a and ρ_w are air and ocean water densities – respectively, u_a is the zonal wind forcing.

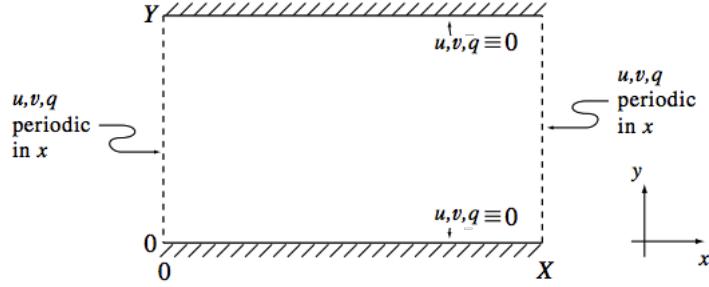


Figure 1. Boundary conditions for the equations (1)–(3)

The spatial discretization for the system (1)–(3) is carried out by finite difference technique, and the forward-backward method is applied for time integration [14]. The space mesh 2D discretization follows the Arakawa grid-C scheme – see Fig. 2.

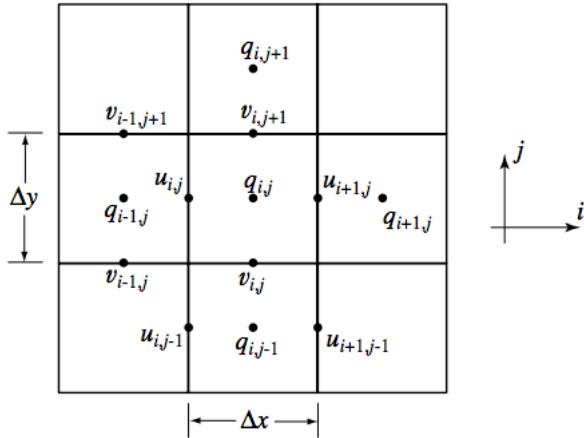


Figure 2. The Arakawa grid-C employed for space discretization

2. Data Assimilation Methods

As already mentioned, data assimilation is a scheme to compute the initial condition by combining the background fields with the available observations, producing the *analysis*. The supervised artificial neural network is self-configured to emulate the Kalman filter. Therefore, both DA methods are presented in this section.

2.1. Kalman Filter

A Kalman Filter is a well-known scheme to estimate the unknown state by least-squares taking into account the Gaussian statistics for the measurement and the modeling errors. Data assimilation algorithm using Kalman filter is written as:

1. Forecast model for state vector (M_N is the matrix of the system): $x_{n+1}^f = M\{x_n^a\} \approx M_n x_n^a$.

2. Update the forecasting covariance matrix (W^{mod} is the modeling covariance error matrix, and P^a the analysis covariance matrix – see item 5):

$$P_{n+1}^f = M_n P_n^a M_n^T + W_n^{\text{mod}}.$$
3. Compute the Kalman gain (W^{obs} is the measurement covariance error matrix):

$$K_{n+1} = P_{n+1}^f H_{n+1}^T [W_{n+1}^{\text{obs}} + H_{n+1} P_{n+1}^f H_{n+1}^T]^{-1}.$$
4. Compute the analysis (DA) – x_{n+1}^a , with x^{obs} being the state observation vector:

$$x_{n+1}^a = x_{n+1}^f + K_{n+1} [x_{n+1}^{\text{obs}} - (H_{n+1} x_{n+1}^f)].$$
5. Update the analysis covariance: $P_{n+1}^a = [I - K_{n+1} H_{n+1}] P_{n+1}^f.$

Data assimilation using the Kalman filter has a high computational effort. The processing DA cost can be mitigated by using artificial neural networks trained to emulate the KF. In fact, the MLP-NN has a lower computational complexity than KF, as shown below.

2.2. Artificial Neural Network (ANN)

Artificial neural networks (ANN) have been used for many applications. However, The neural network employed to DA is a relatively recent application. The supervised multilayer perceptron (MLP) [10], using a back-propagation algorithm for the training phase, is used here to emulate the Kalman filter by reducing the computational effort during the DA process [16]. The best topology for the MLP-ANN is found by solving an optimization problem with cost functional:

$$f_{\text{obj}} = \text{penalty} \times \left[\frac{\rho_1 \times E_{\text{train}} + \rho_2 \times E_{\text{gen}}}{\rho_1 + \rho_2} \right], \quad (5)$$

where the errors associated to the training and generalization evaluations are denoted by E_{train} and E_{gen} , balancing parameters for the generalization and training errors are given by: $\rho_1 = \rho_2 = 0.5$. The penalty term is a degree of neural network complexity. The procedure searches for a neural network with a fewer neurons and faster training iteration. The penalty factor is expressed by:

$$\text{penalty} = c_1 e^{(n_{\text{neurons}})^2} + c_2 (n_{\text{epochs}}) + 1 \quad (6)$$

with $c_1 = 5 \times 10^8$ and $c_2 = 5 \times 10^5$ as used by Anochi and co-authors [1]. The objective function (5) is solved by a meta-heuristic called multi-particle collision algorithm (MPCA) [12]. Figure 3 shows the isovales for q -variable in the shallow water system (1)–(3) at time-step $t = 30$, where TRUE, KF, and ANN are reference, Kalman filter, and neural network configured by MPCA (emulating Kalman filter) [16], respectively.

2.3. KF and MLP-NN: Algorithmic Complexity

Unlike in Section 2.2, the expression complexity is not linked to the number of artificial neurons or how fast the convergence is for the training phase. The *algorithmic complexity* is related to the number of arithmetic operations of an algorithm. Cintra and co-authors [5] have applied MLP-NN to reduce the Kalman filter complexity, where the NN was trained as an operator to matrix inversion. Indeed, depending on the application, the Kalman filter complexity can be reduced when the system matrix can be partitioned into smaller dimension matrices [17].

For general applications, the Kalman filter complexity has order $O(M^3)$ in terms of a number of floating-point multiplications [13] – M being the state-vector dimension, due to the matrices operations in the Kalman filter algorithms.

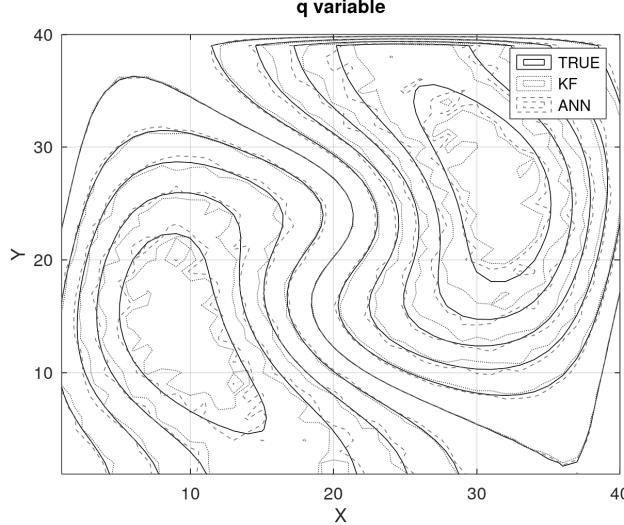


Figure 3. Data assimilation for shallow water system 2D applying Kalman filter and neural network at $t = 30$ – see also reference [16]

In the analysis for neural network algorithm, we are going to consider the $M_0 = 2 \times M_2$ inputs – with: M_2 is the number of observations and background grid points, L the number of layers (including output layer), $M_{\ell,1}$ ($\ell = 1, 2, \dots, L - 1$) the number of artificial neurons for the ℓ -th hidden layer – for simplicity, we are considering: $M_{1,1} = M_{2,1} = \dots = M_{L-1,1} = M_1$. For each neuron, a linear combination of M inputs has the complexity:

$$s = \sum_{i=1}^M \theta_i X_i \sim O(M) , \quad (7)$$

where θ_i are the connection weights and X_i the inputs, respectively. The value s feeds the activation function $\varphi(s)$, representing the non-linear term in the supervised neural mapping. The activation function can be computed from a polynomial approximation by Horner's rule (*nested multiplication*):

$$\begin{aligned} \varphi(s) &= a_0 + a_1 s + a_2 s^2 + \dots + a_{N-1} s^{N-1} + a_N s^N \\ &= a_0 + s(a_1 + s(a_2 + \dots + s(a_{N-1} + a_N s) \dots)) \end{aligned} \quad (8)$$

with $O(N)$ of multiplications and additions. Denoting by C_{NN} the complexity of the MLP-NN, this value can be estimated by:

$$C_{NN} \sim M_1 [O(M_0) + O(N)] + (L - 1) M_1 [O(M_1) + O(N)] + M_2 [O(M_1) + O(N)] . \quad (9)$$

Considering the entries of inputs and outputs, one can consider: $O(M_0) \sim O(M_1) \sim O(M_2) \sim O(M)$, where the stronger assumption is $O(M_1) \sim O(M)$. For standard applications, $L \ll M$, and $O(N) < O(M)$. Therefore, $C_{NN} \sim O(M^2)$. For applications: $C_{KF} = O(M^3) > O(M^2) = C_{NN}$, showing that multilayer perceptron neural network has a smaller complexity than Kalman filter.

Before showing the results with parallel runs, a sequential execution was carried out for a comparison considering CPU-time between data assimilation performed by Kalman filter and neural network [9] with $N_x^{(1)} = N_y^{(1)} = 40$. Two numerical experiments were considered with 25

(Exp-1) and 100 (Exp-2) observation points. Table 1 shows the CPU-time (hr:min:sec) for two simulations with different number of observations for the data assimilation process.

Table 1. CPU-time for data assimilation using Kalman filter (KF) and multi-layer perceptron neural network (MLP-NN) with 25 (Exp-1) and 100 (Exp-2) observation points per data assimilation cycles

Experiments	KF	MLP-NN
Exp-1	00:42:02	00:01:39
Exp-2	01:19:03	00:05:01

From the results shown in Tab. 1, data assimilation process with neural network was about 25 and 15 times faster than Kalman filter for Exp-1 and Exp-2, respectively. These results for CPU-time are in agreement with results obtained with global atmospheric models. Cintra and Campos Velho did data assimilation experiments with SPEED global model (3D spectral model with very simplified physical parameterizations) [6], where the NN approach was 95 times faster than ensemble Kalman filter (EnKF). Another result with comparison with NN and EnKF with global atmospheric model was carried out with Florida State University (FSU) model. Data assimilation by the NN was 55 times faster than applying EnKF for the FSU model [7].

For parallel version implementation, the data assimilation by NN is much more effective than KF. In the analysis computed by NN, the procedure is performed at each variable for a specific grid point, combining background value with observation available. In the KF, the analysis is calculated by estimating the Kalman gain (item 3, Section 2.1), involving matrix multiplication and inversion, with another step for producing the analysis from a product between a matrix (Kalman gain) and a vector (difference between background and observation vectors) – see item 4, Section 2.1. Therefore, the parallel implementation is much more effective for NN than KF.

3. Parallel Version Strategy

The data assimilation process described in Section 2, is summarized by algorithm SW2D_DA (Algorithm 1). For the worked example here, only assimilation for the q -variable is assimilated. The algorithm (or function) to implement the shallow-water model (SW2D_MODEL) is called at all N_t timesteps. In contrast, the Kalman filter data assimilation algorithm (KF_DA) or the neural networks (ANN_DA, showed in Algorithm 2) is triggered at regular intervals of timesteps (data assimilation cycles), represented by $freqObsT$, called here as the *frequency of observation*.

In this article, the Kalman filter algorithm will not be shown in detail since the focus of the work was the parallelization of data assimilation by neural networks for a space domain with a high number of grid points. In Algorithm 2, the DA is carried out independently for each grid point. Therefore, the parallel strategy is to compute the DA for each grid point in parallel. Considering N_g the number of the grid points and N_p the number of processors, the analysis is computed by a trivial parallel approach, executing N_g/N_p computation cycles for completing the DA on the entire space domain.

The loops traversing the grid points in the horizontal and vertical directions are parallelized with OpenMP directives, and the FORTRAN source code where the parallel strategy was implemented is in Fig. 4. The same approach was employed to a parallel version of the shallow-water function (SW2D_MODEL), as can be seen in Fig. 5.

Algorithm: SW2D_DA

```

input :
     $q^{Model}$ : reference SW2D model values (true)
     $q^{Observ}$ : observed SW2D values (true + noise)
     $N_t$ : number of timesteps
     $freqObsT$ : frequency of observation
        (defines number of assimilation cycles)
     $N_x$ : number of grid points in horizontal direction
     $N_y$ : number of grid points in vertical direction
     $assimType$ : data assimilation type (KF or ANN)

output:
     $q^{Analysis}$ : result of data assimilation

begin
    for  $t \leftarrow 1$  to  $N_t$  do
        SW2D_MODEL( $N_x, N_y, u, v, q$ )
         $q_{(t)}^{Analysis} = q$ 
        if  $mod(t, freqObsT) = 0$  then
            switch  $assimType$  do
                case 1 do
                    | KF_DA( $N_x, N_y, q_{(t)}^{Analysis}$ )
                end
                case 2 do
                    | ANN_DA( $N_x, N_y, q_{(t)}^{Model}, q_{(t)}^{Observ}, q_{(t)}^{Analysis}$ )
                end
            end
        end
    end

```

Algorithm 1. Shallow-Water 2D Data Assimilation (SW2D_DA)

```

!$OMP PARALLEL DO      &
!$OMP DEFAULT(shared)  &
!$OMP PRIVATE(sX,sY,i,tid)
do sX = 1, gridX
    do sY = 1, gridY
        tid = omp_get_thread_num() + 1
        i = (sX-1)*gridY + sY
        xANN(1,i) = qModelnorm(sX,sY,tS)
        xANN(2,i) = qObsevnorm(sX,sY,tS)
        vco(:,1,tid) = matmul(wqco(:, :), xANN(:, i))
        vco(:,1,tid) = vco(:,1,tid) - (bqco(:,1))
        yco(:,1,tid) = (1.d0 - DEXP(-vco(:,1,tid))) / (1.d0 + DEXP(-vco(:,1,tid)))
        vcs(:,1,tid) = matmul(wqcs(:, :), yco(:,1,tid))
        vcs(:,1,tid) = vcs(:,1,tid) - bqcs(:,1)
        ycs(:,1,tid) = (1.d0-DEXP(-vcs(:,1,tid)))/(1.d0+DEXP(-vcs(:,1,tid)))
        qGl(sX,sY) = (ycs(1,1,tid)*(qModelMax-qModelMin) + qModelMax + qModelMin)/2.0
    enddo
enddo
!$OMP END PARALLEL DO
qAnalysis(:, :, tS) = qGl

```

Figure 4. Parallel OpenMP Fortran code for the Algorithm-2

Algorithm: ANN_DA

input :

q^{Model} : reference SW2D model values (true)
 q^{Observ} : observed SW2D values (true + noise)
 N_x : number of grid points in horizontal direction
 N_y : number of grid points in vertical direction

output:

$q^{Analysis}$: result of data assimilation

begin

for $i \leftarrow 1$ to N_x **do**

for $j \leftarrow 1$ to N_y **do**

$$v_{1,2}(i,j) = \sum_{l=1}^{\text{#neurons}} [w_{1l}(i,j) q^{Model} + w_{2l}(i,j) q^{Observ}(i,j) + b(i,j)]$$

$$q^{Analysis}(i,j) = \tanh[v_1(i,j)] + \tanh[v_2(i,j)]$$

end

end

end

Algorithm 2. Artificial Neural Network Data Assimilation (ANN_DA)

```

!$OMP PARALLEL
!$OMP DEFAULT(shared)
!$OMP PRIVATE(i, j)

!$OMP DO
    do i = 1, ni - 1
        do j = 1, nj - 1
            divx(i,j) = cx * (uGl(i+1, j) - uGl(i, j))
        enddo
    enddo
!$OMP END DO

!$OMP DO
    do j = 1, nj - 1
        divx(ni,j) = cx * (uGl(1,j) - uGl(ni,j))
    enddo
!$OMP END DO
...
!$OMP END PARALLEL

```

Figure 5. Parallel OpenMP Fortran code of shallow-water 2D model

4. Results

The shallow-water system was defined by considering the ocean circulation. The numerical values for the parameters are shown in Tab. 2, with $t_{\max} = N_t \Delta t$, the spatial domain discretization given by Δx and Δy , and N_x and N_y are, respectively, the number of grid points in horizontal and vertical directions, and the upper indexes ⁽¹⁾ and ⁽²⁾ are related to the 40-point and 2560-point grid sizes. Finally, the data assimilation cycle (the frequency of observation in Algorithm 2) is performed at each 10 time-steps ($freqObsT = 10$).

The executions were made in one compute node of the Santos Dumont supercomputer (an ATOS machine). The computer node has two CPU Intel Xeon E5-2695v2 with 48 cores and 384 Gigabytes of RAM. Initially, for serial performance comparison purposes between the original assimilation method with Kalman Filter and the method with neural networks, a 40-point grid size was used. According to results from Furtado and co-authors [9], the KF method

Table 2. Parameters used in the integration for the SW-model

Parameter	Value	Parameter	Value
$\Delta t (h)$	180	$r_u (s^{-1})$	1.8×10^4
N_t	200	$r_v (s^{-1})$	1.8×10^4
$t_{\max} (h)$	3.6×10^4	$r_q (s^{-1})$	1.8×10^4
$\Delta x (km)$	10^5	$\rho_a (kg/m^{-3})$	1.275
$\Delta y (km)$	10^5	$\rho_w (kg/m^{-3})$	1.0×10^3
$N_x^{(1)}$	40	C_d	1.6×10^{-3}
$N_y^{(1)}$	40	$H (m)$	5000
$N_x^{(2)}$	2560	$g (m/s^{-2})$	9.806
$N_y^{(2)}$	2560	$f (s^{-1})$	1.0×10^{-4}

is much more computing expensive than the ANN method. In addition to the ANN method being considerably faster, the final result obtained is relatively close to that of KF, and also to the reference solution (TRUE) for the q shallow-water variable at grid position (8,8), as can be seen in Fig. 6a. Similar comparisons between KF and ANN methods have already previously been done – see references [4, 9, 16].

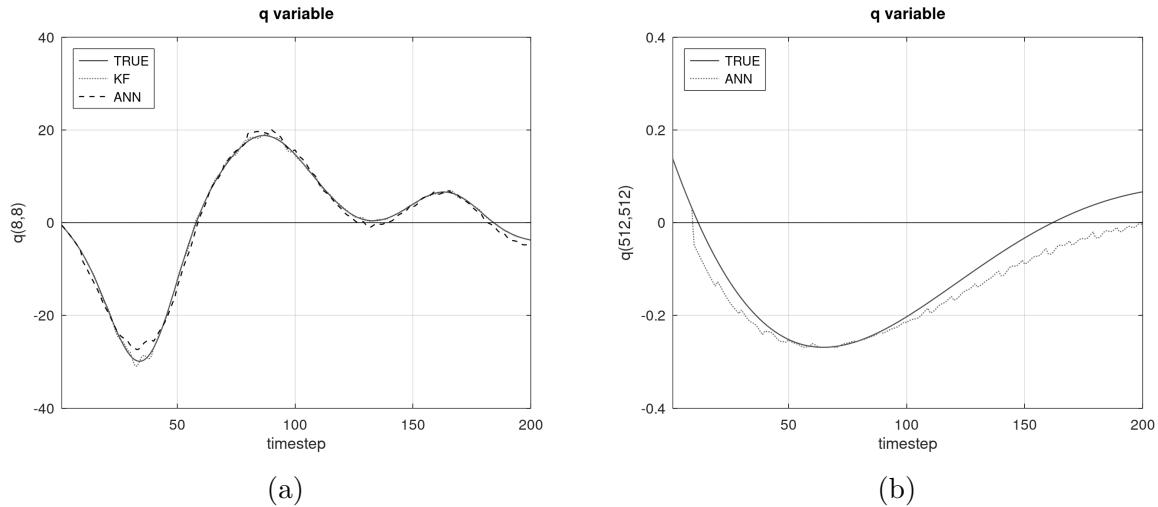


Figure 6. The reference (TRUE) shallow-water, KF and ANN data assimilation values of variable q at grid position (8,8) for 40-point grid size (a), and at grid position (512,512) for 2560-point grid size (b), using weights and bias obtained for the 40-point grid size neural network

The evaluation of parallel performance of the ANN assimilation method for a computational problem with a high number of grid-points was tested. The number of grid-points used for this purpose was one with 2560 points in the horizontal and vertical coordinates, i.e., $N_x^{(2)} = N_y^{(2)} = 2560$. For this grid size, it was unfeasible to obtain the KF data assimilation result, with the actual source code. Figure 6 shows the comparison result only between the reference q values (TRUE) and the ANN assimilation result (ANN_DA) at grid position (512,512), but using the same neural network employed for the 40-point grid size. As mentioned, the neural network for the finer resolution problem is the same of that configured to emulate the Kalman filter with

the a coarser computational mesh. Even so, we can observe the neural network dynamics close to the curve of the true solution – see Fig. 6b.

The serial execution profiling of the Fortran implementation of Shallow-Water 2D Data Assimilation algorithm (SW2D_DA, in Algorithm 1) is shown in Tab. 3. The biggest hotspot is the function SW2D_MODEL, which integrates the 2D shallow-water model in all 200 timesteps. The second hotspot is the ANN_DA function, which emulates data assimilation obtained by Kalman filter using an artificial neural network. Important to note that this function is activated only at the end of each 10-timesteps cycle. Therefore, it is called in only 20 times from a total of 200 timesteps.

Table 3. Serial performance profiling

Function	Time (s)	Time share (%)
SW2D_MODEL	217.1	74.7
ANN_DA	41.2	14.2
OTHERS	32.3	11.1
Total time	290.6	100.0

The parallel performance of the functions SW2D_DA and ANN_DA, obtained using up to 32 OpenMP threads, is presented in Tab. 4. A reduction about ten times from the serial time was achieved in the first function (SW2D_Model), while a less significant reduction was observed in the second function (ANN_DA).

The processing time reduction in the shallow-water function SW2D_DA results from the good parallel efficiency achieved, especially with up to 16 threads. Using 32 OpenMP threads, the runtime reduces from 217.1 seconds to 34.7 seconds. However, we believe the speed-up obtained with 32 threads could be even better.

In order to have a better understanding for the results with 16-threads and 32-threads, the number of grid points were increased to $N_x = N_y = 4000$ (in Tab. 5), just to verify if there is a saturation for processing demand, i.e., enhancing the computational load, a better speed-up should be obtained. However, we got a worse performance than 2500 grid points. Therefore, such behavior from the results show other issues are acting. Probably the performance results are linked to the cache misses and/or synchronization among the processing cores. Further investigation is needed to improve the parallel efficiency with this number of threads.

The parallel performance obtained with the ANN_DA function was better than 2D shallow-water function. Using 32 OpenMP threads, the runtime for ANN_DA function is reduced from 41.2 seconds to 2.1 seconds for the 2560-grid size, obtaining a speed-up of almost 20 times concerning the serial execution, according to values presented in Tab. 4. A similar speed-up was also reached for the 4000-grid size, shown in Tab. 5. In this case, the runtime is reduced from 91.9 seconds to 5.0 seconds, obtaining a speed-up about 18 times.

After the parallelization performed in the two main hotspots, the remaining code (OTHERS in Tab. 3), not listed here, are instructions used to prepare the memory for SW2D_Model and ANN_DA routines. Since it amounts to 11.1% of the total time, improving these functions' performance is not mandatory in future developments.

Table 4. Parallel performance of shallow-water 2D model and ANN assimilation for 2560-grid points in both X and Y coordinates

SW2D_MODEL				ANN_DA			
#threads	Time (s)	Speed-up	Eff	#threads	Time(s)	Speed-up	Eff
1	217.1	1.0	1.00	1	41.2	1.0	1.00
2	119.8	1.8	0.91	2	21.4	1.9	0.96
4	70.0	3.1	0.78	4	11.2	3.7	0.92
8	45.6	4.8	0.60	8	6.0	6.9	0.86
16	31.1	7.0	0.44	16	3.1	13.3	0.83
32	34.7	6.3	0.20	32	2.1	19.6	0.61

Table 5. Parallel performance of shallow-water 2D model and ANN assimilation for 4000-grid points in both X and Y coordinates

SW2D_MODEL				ANN_DA			
#threads	Time (s)	Speed-up	Eff	#threads	Time(s)	Speed-up	Eff
1	555.5	1.0	1.00	1	91.9	1.0	1.00
2	312.5	1.8	0.89	2	49.7	1.8	0.92
4	195.8	2.8	0.71	4	28.0	3.3	0.82
8	134.3	4.1	0.52	8	15.4	6.0	0.75
16	114.6	4.8	0.30	16	8.1	11.3	0.71
32	155.5	3.6	0.11	32	5.0	18.4	0.57

5. Final Remarks

The parallel processing techniques were applied to reduce data assimilation processing time with neural networks for domains with an increased grid points density, presenting a more significant speeding-up. However, a deeper study must be carried out to obtain a better parallel efficiency of the function implemented to the shallow-water 2D algorithm. A preliminary strategy got implemented using OpenMP. Thus, one way to improve parallel performance can be through a better choice of thread scheduling. Looking at a higher level of parallelism, one can also use MPI to execute the code in a distributed memory machine, a cluster p.ex., through a subdivision of the spatial domain. In this case, one can even run costly instances of the shallow-water problem, using a grid containing an even more significant number of points.

Acknowledgements

The authors acknowledge the National Laboratory for Scientific Computing (LNCC/MCTI, Brazil) for providing HPC resources of the SDumont supercomputer, which have contributed to the research results reported within this paper. URL: <http://sdumont.lncc.br>. The authors would also like to thank the Brazilian agencies for their research support. Author HFCV thanks the National Council for Scientific and Technological Development (CNPq, Portuguese) for the research grant (CNPq: 312924/2017-8).

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. Anochi, J.A., Campos Velho, H.F., Hernandez Torres, R.: Two geoscience applications by optimal neural network architecture. *Pure and Applied Geophysics* 1776(1), 1–21 (2019). <https://doi.org/10.1007/s00024-019-02386-y>
2. Bennett, A.F.: Inverse Modeling of The Ocean and Atmosphere. Cambridge University Press (2002)
3. Boucher, M.A., Quilty, J., Adamowski, J.: Data assimilation for streamflow forecasting using extreme learning machines and multilayer perceptrons. *Water Resources Research* 56 (2020). <https://doi.org/10.1029/2019WR026226>
4. Campos Velho, H.F., Härter, F.P., Rempel, E.L., Chian, A.: Neural networks in auroral data assimilation. *Journal of Atmospheric and Solar-Terrestrial Physics* 70(10), 1243–1250 (2008). <https://doi.org/10.1016/j.jastp.2008.03.018>
5. Cintra, R.C., Campos Velho, H.F., Todling, R.: Redes neurais artificiais na melhoria de desempenho de métodos de assimilação de dados: filtro de Kalman. *TEMA: Trends in Computational and Applied Mathematics* 11(1), 29–39 (2010). <https://doi.org/10.5540/tema.2010.011.01.0029>
6. Cintra, R.S.C., Campos Velho, H.F.: Data assimilation by artificial neural networks for an atmospheric general circulation model. In: El-Shahat, A. (ed.) *Advanced Applications for Artificial Neural Networks*, chap. 14, pp. 265–285. Intech (2018). <https://doi.org/10.5772/intechopen.70791>
7. Cintra, R.S.C., Campos Velho, H.F., Cocke, S.: Tracking the model: data assimilation by artificial neural network. In: IEEE International Joint Conference on Neural Networks – IJCNN, Vancouver, Canada, July 24-29, 2016. vol. 4, pp. 403–410 (2016). <https://doi.org/10.1109/IJCNN.2016.7727227>
8. Daley, R.: Atmospheric Data Analysis. Cambridge University Press (1993)
9. Furtado, H.C., Cintra, R.S.C., Campos Velho, H.F., et al.: Neural network for data assimilation method applied to shallow water equation. In: 2nd International Symposium on Uncertainty Quantification and Stochastic Modeling, Rouen, France, July 7-11, 2014. pp. 299–311 (2014)
10. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall Inc. (1994)
11. Kalnay, E.: Atmospheric Modeling, Data Assimilation, and Predictability. Cambridge University Press (2003)
12. Luz, E.F.P., Becceneri, J.C., de Campos Velho, H.F.: A new multi-particle collision algorithm for optimization in a high performance environment. *Journal of Computational Interdisciplinary Sciences* 1(1), 3–10 (2008). <https://doi.org/10.6062/jcis.2008.01.01.0001>

13. Mendel, J.: Computational requirements for a discrete Kalman filter. *IEEE Transactions on Automatic Control* 16(6), 748–758 (1971). <https://doi.org/10.1109/TAC.1971.1099837>
14. Mesinger, F., Arakawa, A.: Numerical methods used in atmospheric models. *Global Atmospheric Research Program – World Meteorological Organization* (1976)
15. Reich, S., Cotter, C.: Probabilistic Forecasting and Bayesian Data Assimilation. Cambridge University Press (2015)
16. Sambatti, S.B.M., Campos Velho, H.F., Furtado, H.C.M., et al.: Self-configured neural network for data assimilation using FPGA for ocean circulation. In: 3Conference of Computational Interdisciplinary Science (CCIS 2016), São José dos Campos (SP), Brazil (2016)
17. Vaidehi, V., Krishnan, C.N.: Computational complexity of the Kalman tracking algorithm. *IETE Journal of Researcht* 44(3), 125–134 (1998). <https://doi.org/10.1080/03772063.1998.11416038>

Multistage Iterative Method to Tackle Inverse Problems of Wave Tomography

Alexander V. Goncharsky^{1,2}, Sergey Y. Romanov^{1,2}, Sergey Y. Seryozhnikov^{1,2}

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

This paper is concerned with developing the methods for solving inverse problems of low-frequency ultrasound tomography under scalar wave models using supercomputer technologies. Unlike X-ray tomography, the inverse problem considered is posed as a problem of minimizing a non-convex residual functional. The multistage iterative method (MSM) is proposed as a method for obtaining an approximate solution to the inverse problem. Convergence of the method to the exact solution is achieved via the use of low-frequency sounding signals at the initial stages of the iterative method. The method is illustrated on model problems focused on ultrasound tomographic diagnostics of soft tissues in medicine. Finite-difference time-domain method is used to solve the wave equation, which accounts for most of the computational complexity of the method. The multistage method reduces the computation time, since the initial stages use low-resolution finite difference grids. The effectiveness of the MSM method is investigated on GPU and SIMD-capable CPU computing platforms. Numerical simulations showed that modern processors equipped with AVX-512 FPUs are capable of solving small-scale problems of wave tomography. For large-scale tasks, GPUs equipped with fast on-board memory are preferred. The numerical algorithm is data-parallel and well-suited for GPU architecture. The proposed method can be used in medical imaging and nondestructive testing applications.

Keywords: *ultrasound tomography, coefficient inverse problem, gradient method, numerical simulation.*

Introduction

At present, it is hard to imagine a field of science or technology where tomographic imaging is not used. First tomographs that appeared in the middle of the latest century used X-ray radiation. However, the history of tomography can well begin at the beginning of the latest century, when X-rays have been discovered (Nobel Prize of the year). At the same time, Radon's solution of the inverse problem of reconstructing a function of two variables given its linear functionals has been published [1]. This result essentially solved the mathematical problems of tomography in a linear model. However, it took humanity half a century of intensive scientific research to develop first X-ray tomographs.

Currently, various tomographs (X-ray, MRI, positron emission tomography) are widely used in medicine, science and technology. All these technologies are united by the fact that, from a mathematical point of view, the problems of interpreting the data of tomographic experiments are inverse problems that can be solved within the framework of linear mathematical models. Solving such problems does not presently pose serious mathematical concerns. Personal computers are sufficient for data interpretation in a linear problem.

At the end of the latest century, it became possible to research a very interesting field of wave tomography, which employs ultrasonic, electromagnetic, seismic or optical radiation for sounding. For all these problems, it is necessary to use nonlinear mathematical models to interpret the data of tomographic experiments.

The progress in wave tomography developments has been driven by several factors. The first is the development of modern methods for solving inverse problems. First results obtained

¹Lomonosov Moscow State University, Moscow, Russia

²Moscow Center of Fundamental and Applied Mathematics, Moscow, Russia

by Academician Tikhonov in the 60s [2, 3] were continued in the works of his students and followers [4–6]. By the end of the nineties, exhaustive results had been obtained in the field of solving ill-posed linear and nonlinear problems. Tikhonov brought the concept of a regularizing algorithm as a method for the approximate solution of the inverse problem. Within the framework of this concept, effective numerical methods have been developed for solving a wide range of problems in mathematical physics [7–11]. Important results were obtained in the field of using iterative schemes for the approximate solution of nonlinear inverse problems [5]. Intensive research was carried out in the field of solving coefficient inverse problems of mathematical physics [12, 13]. Fast-growing supercomputer technology was another factor that contributed to the development of wave tomography. The solution of inverse problems under the wave model requires ample computational resources due to large problem dimensions and its nonlinearity. It is impossible to solve inverse problems of wave tomography without the use of supercomputers [14, 15].

In short, the situation with wave tomography at the moment can be characterized as follows. Most works on wave tomography consider scalar wave models that take into account the effects of diffraction, refraction, and even multiple scattering. The inverse problem in this case can be posed as a problem of minimizing the residual functional between the experimental data (measured wave field at the detectors) and the wave field computed using the mathematical model of wave propagation. The most important recent result in wave tomography is the ability to calculate the gradient of the residual functional explicitly [16–18]. This result makes it possible to use gradient-based methods for minimizing the residual functional to obtain an approximate solution to the inverse problem. Since the functional is not convex and has local minima, the problem of finding the global minimum of the functional arises. Despite the large number of works on this topic [19–21] this problem is unsolvable in a general case. In this paper, it is proposed to narrow the search area using additional information in order to find the global minimum of the residual functional.

Using additional information in solving inverse problems is not a new approach. In [4, 5] it was proposed to use such information about the sought-for functions as their monotonicity or convexity for constructing approximate solutions to ill-posed problems. In a finite-difference approximation, the problem reduces to minimizing a functional on a convex polyhedron with known vertices. Effective numerical algorithms have been developed for this approach.

To solve the problems of wave tomography, the authors propose a multistage iterative method (MSM) that uses additional prior information specific to these inverse problems. As a possible application of wave tomography, medical tomographic imaging for differential diagnosis of breast diseases is considered. Several groups of researchers are intensively working in this field [22–24]. These developments are currently at the stage of prototypes. At the moment, the main problem is constructing effective algorithms for data interpretation. This paper demonstrates that MSM can effectively find an approximate solution to inverse problems of tomographic image reconstruction in application to medical ultrasound imaging of soft tissues. The question of choosing the optimal computing platform for the proposed method is discussed.

Developing ultrasound tomography devices is a challenging task. A natural question arises: what advantages ultrasound tomography can provide in comparison with existing diagnostic methods. Let us try to give an answer to this question using the example of medical diagnostics, where ultrasound devices have been successfully used for a long time. To discuss the problem more specifically, we will narrow the field of medical diagnostics to soft tissue imaging. Unlike conventional ultrasound instruments, ultrasound tomographs can characterize the inspected tis-

sues. Just as in X-ray tomography, the doctor can obtain the value of the sound wave velocity at any point of the image. This result opens up the possibility of classifying neoplasms by the speed of sound in them. Neural networks can automate the tissue classification process [25]. Why cannot this be done using standard ultrasound diagnostic devices? Both X-ray and ultrasound tomography employ sounding waves transmitted through the object. Conventional ultrasound instruments detect only reflected waves, and this information is principally insufficient for tissue characterization. Finally, unlike X-ray tomography, ultrasound tomography is completely safe and therefore can be used for regular screening.

The article is organized as follows. Section 1 describes the inverse problem of wave tomography and the solution method. In Section 2 we introduce the proposed multistage iterative method. Section 3 describes the finite difference numerical method employed in the solution algorithm, Section 4 describes the parallel implementation of the solution algorithm. Section 5 compares the performance of the algorithm on various CPU and GPU computing systems. Section 6 presents model problems to demonstrate the proposed multistage method. Conclusion summarizes the study and points directions for further work.

1. Formulation of the Inverse Problem of Wave Tomography and its Solution Method

In this study, we consider the waves described by the scalar wave equation. In the scalar model, the scalar wave field $u(\mathbf{r}, \mathbf{q}, t)$, which represents the acoustic pressure, can be computed from the given initial data using the equation

$$c(\mathbf{r})u_{tt}(\mathbf{r}, \mathbf{q}, t) - \Delta u(\mathbf{r}, \mathbf{q}, t) = \delta(\mathbf{r} - \mathbf{q})g(t) \quad (1)$$

$$u(\mathbf{r}, \mathbf{q}, t = 0) = u_t(\mathbf{r}, \mathbf{q}, t = 0) = 0. \quad (2)$$

Here, $c^{-0.5}(\mathbf{r}) = v(r)$ is the speed of sound in the medium, $\mathbf{r} \in \mathbb{R}^2$, Δ is the Laplacian operator with respect to \mathbf{r} , δ is the Dirac delta function, which defines a point source at \mathbf{q} . The sounding pulse emitted by the source is described by function $g(t)$. Short broadband sounding pulses with a useable frequency range of 100–600 kHz and a duration up to 50 μs can be used for sounding in medical ultrasound tomography. The sounding pulses are further discussed in Section 6.

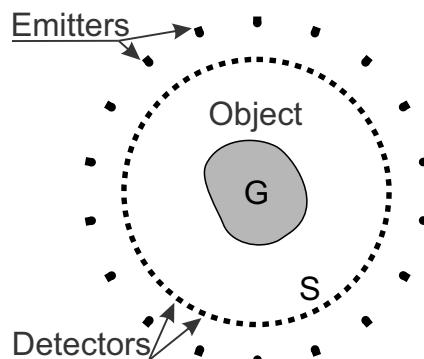


Figure 1. Ultrasound tomographic imaging scheme

The inverse problem of ultrasound tomography can be formulated as follows. Figure 1 shows the scheme of ultrasound tomographic imaging. The object with an unknown speed of sound $v(\mathbf{r})$ occupies region G. The object G, emitters and detectors of ultrasound are placed in a homogeneous medium with a known speed of sound $v_0 = c_0^{-0.5} = \text{const}$. The emitters are located at coordinates \mathbf{q}_j . A total of M emitter positions $j = 1, \dots, M$ are located around region G. Measurements of the wave field $u(\mathbf{r}, \mathbf{q}, t)$ are taken on a circle S surrounding region G.

Function $g(t)$, which describes the sounding pulse, is known. In the inverse problem, the objective is to determine an unknown function $v(\mathbf{r})$ at $\mathbf{r} \in G$, using experimental data $U(s, \mathbf{q}_j, t)$ obtained at the boundary S ($s \in S$) for emitter positions $\mathbf{q}_j, j = 1, \dots, M$. Thus, the wave field $u(s, \mathbf{q}_j, t)$ from equations (1)–(2) satisfies the following equation for all emitter positions \mathbf{q}_j :

$$u(s, \mathbf{q}_j, t)|_{s \in S} = U(s, \mathbf{q}_j, t). \quad (3)$$

The system of equations (1)–(3) defines the inverse problem. Thus, solving inverse problem of ultrasound tomography in the scalar model involves reconstructing the unknown wave velocity $v(\mathbf{r})$ in region G according to equations (1)–(3).

The residual functional $\Phi(c)$ of the argument $c(\mathbf{r})$ is the difference between the experimental data and the data computed from equations (1)–(2). The residual functional for a computed wave field on the boundary S determined by a given speed of sound $c(\mathbf{r})$ can be written as

$$\Phi(c) = \sum_{j=1}^M \frac{1}{2} \int_0^T \int_S (u(s, \mathbf{q}_j, t) - U(s, \mathbf{q}_j, t))^2 \, ds \, dt. \quad (4)$$

Here $U(s, \mathbf{q}_j, t)$ are the experimental data on the boundary S for the time interval $(0, T)$, and $u(s, \mathbf{q}_j, t)$ is the wave field obtained via solving the direct problem (1)–(2), which depends on the specified coefficient $c(\mathbf{r})$. For multiple sounding wave sources, the residual functional is the sum over $j = 1, \dots, M$ of the residual values obtained for each source. For each fixed source j , the integral is summed over time $(0, T)$ and over the boundary S – for all the detectors receiving the signal from the selected source. Mathematically, the inverse problem is posed as a problem of finding a function $\bar{c}(\mathbf{r})$ that minimizes the residual functional (4) $\bar{c}(\mathbf{r}) : \min_{c(\mathbf{r})} \Phi(c) = \Phi(\bar{c})$. The $\bar{c}(\mathbf{r})$ function is taken as an approximate solution to the inverse problem. Gradient methods have proven effective for minimizing the residual functional $\Phi(c)$. A rigorous mathematical formulation for the gradient of the residual functional has been obtained in [17, 18, 26]. The gradient of the functional (4) has the form

$$\Phi'(c) = \sum_{j=1}^M \int_0^T \frac{1}{2} w_t(s, \mathbf{q}_j, t) u_t(s, \mathbf{q}_j, t) \, dt,$$

where $u(s, \mathbf{q}_j, t)$ is the solution of the main problem (1)–(2) and $w(s, \mathbf{q}_j, t)$ is the solution of the conjugate problem (6)–(7). Both solutions depend on $c(\mathbf{r})$ coefficients [14, 18].

$$c(\mathbf{r}) w_t(\mathbf{r}, \mathbf{q}_j, t) - \Delta w(\mathbf{r}, \mathbf{q}_j, t) = u(s, \mathbf{q}_j, t) - U(s, \mathbf{q}_j, t)|_{s \in S}, \quad (6)$$

$$w(\mathbf{r}, \mathbf{q}_j, t = T) = w_t(\mathbf{r}, \mathbf{q}_j, t = T) = 0. \quad (7)$$

The inverse problem of wave tomography in the considered formulation is a nonlinear coefficient inverse problem. In nonlinear problems, typically, the residual functional (4) is not convex,

which means that the functional may have local minima. As a consequence, gradient methods for minimizing the residual functional from an arbitrary initial approximation may converge to a local minimum, but not to the global one.

2. The Main Idea of the Multistage Method for Obtaining Approximate Solutions to Nonlinear Inverse Problems of Ultrasound Tomography

As shown in Section 1, inverse problem of ultrasound tomography can be solved by minimizing the residual functional (4), which may have local minima. There are many works concerned with finding global minima of functionals. However, this problem has no solution in a general case. In problems of wave tomography, an important prior information is present, which is that the area of convergence of iterative processes of minimizing the residual functional strongly depends on the wavelength of the sounding radiation. If the central frequency of the sounding pulses tends to 0, then passing to the limit in equation (1) reduces it to a linear integral equation with respect to the unknown function $c(\mathbf{r})$. This idea of using linear models in problems of wave diagnostics is actively discussed in [7, 27–30].

It would seem that this result opens up wide possibilities for solving inverse problems of nonlinear wave tomography, but this is actually not the case for several reasons. The first reason is that in real ultrasound imaging, the center frequency of the sounding wave determines the resolution, and generally all medical ultrasound devices have center frequencies of 1 MHz or higher. It is not possible to obtain experimental data in the frequency range close to zero.

The second reason is that the experimental data are measured with some error. In order to use the linear approximation, it is necessary to calculate the second derivative of measured waveforms, which is poorly conditioned. Nevertheless, the idea of using low frequencies in problems of ultrasound diagnostics is fruitful, and this idea is used in the proposed multistage iterative method (MSM) for obtaining an approximate solution to the inverse problem of ultrasound tomography.

In this article, the capabilities of the MSM method are illustrated using the problems of ultrasound tomographic imaging of soft tissues in medicine, namely, breast imaging. The problem of early-stage breast cancer diagnosis is one of the most important problems of modern medicine. A characteristic feature of soft tissue imaging is that the difference between the speed of sound in soft tissues and the speed of sound in surrounding water v_0 is quite low and does not exceed 15%. It seems natural to use the known speed of sound v_0 in a homogeneous medium surrounding an object in region G as an initial approximation in iterative processes. However, as it will be shown via numerical simulations, the choice of an initial approximation in the form of a constant v_0 in nonlinear problems does not guarantee convergence to the global minimum of the residual functional.

Multistage iterative method (MSM) proposed in this study for solving coefficient inverse problems of ultrasound tomography ensures convergence of the gradient descent algorithm to the global minimum. By increasing the mean wavelength of the sounding signal we expand the range of initial approximations from which the gradient descent method of minimizing the residual functional converges to the global minimum. This idea is at the heart of the MSM method, and it is demonstrated in this study on a large number of model problems that simulate the problem of

ultrasound tomographic imaging for breast cancer diagnosis for various configurations of objects and sounding pulses.

In a practical implementation of the MSM method, experimental data for two (or more) frequency bands with different central frequencies f_1 and f_2 , $f_1 < f_2$ are used. First, the inverse problem is solved via an iterative gradient descent method for minimizing the residual functional using the lowest frequency band f_1 . The initial approximation is chosen as a constant equal to the speed of sound v_0 in the environment. Then, at the second stage, the inverse problem is solved via an iterative method using a higher frequency band f_2 . The result of solving the inverse problem at the first stage at a lower frequency f_1 is used as an initial approximation of the speed of sound for the gradient descent method at the second stage.

The described scheme of MSM method application consists of two stages. In a real situation, it may be necessary to apply several successive stages for three or more frequency bands. The number of stages depends on the task of ultrasound tomography considered. It is important that at each stage the initial approximation is close enough to the point of the global minimum of the residual functional for the frequency band used at that stage. Numerous model calculations have shown that a five-stage method is optimal for tomographic imaging of soft tissues with low-frequency ultrasound in the 100–600 kHz band. To ensure the convergence of the MSM method to the exact solution of the inverse problem, the first stage should be carried out using a wavelength $\lambda = 12$ mm (the central frequency of the pulse is 125 kHz). Subsequent stages were carried out using wavelengths of 8, 6, 4 mm, and the last stage uses $\lambda = 3$ mm (500 kHz). The wavelength of 3 mm provides a high spatial resolution of the method – 1–1.5 mm in application to soft tissue imaging.

The following model problem illustrates the capabilities of the MSM method. The parameters of model problems correspond to breast ultrasound tomography. The variation of the speed of sound in model problems does not exceed 15%. In order to ensure the convergence of the iterative process, the developed methods of ultrasound diagnostics use significantly lower sounding frequencies compared to conventional medical ultrasound devices.

The following parameter values were used in the numerical simulations. The speed of sound in the medium surrounding the object is $v_0 = 1.5$ mm/ μ s (water), the speed of sound in the object varies from 1.4 mm/ μ s to 1.7 mm/ μ s. The calculations were carried out for five frequency bands with central frequencies of 125, 188, 250, 375 and 500 kHz (mean wavelengths of sounding pulses equal to 12, 8, 6, 4 and 3 mm, respectively). The size of the two-dimensional computational domain was 25 × 25 cm, the size of the finite difference grid for the 500 kHz band was 1600 × 1600 points, for the 125 kHz band – 480 × 480 points. In the calculations, the sources and receivers were located around the object. There were 24 source positions in total, located on a circle 200 mm in diameter. The receivers were located with a step of ≈ 1 mm on a circle 165 mm in diameter.

Figure 2 presents the results of solving an inverse problem. Figure 2a shows the original image of the object (simulated phantom). Figure 2b plots the residual functional (4) for three frequency bands with mean wavelengths of $\lambda = 3$ mm, 6 mm and 12 mm. The abscissa corresponds to the parameter α in the interval $(-0.1, 1.4)$, which determines the value of the function $c(\mathbf{r})$ by the following formula $c(\mathbf{r}; \alpha) = (1 - \alpha)c_0 + \alpha\bar{c}(\mathbf{r})$. Here, $c^{-0.5}(\mathbf{r}) = v(\mathbf{r})$ is the wave velocity in the medium, $c_0^{-0.5} = v_0 = \text{const}$. For $\alpha = 1$, we get $c(\mathbf{r}) = \bar{c}(\mathbf{r})$, which is the exact solution and the residual functional equals to 0. For $\alpha = 0$, $c(\mathbf{r}) = c_0$, which corresponds to the initial approximation of the iterative process. For short wavelengths $\lambda = 3$ mm and 6 mm, the value of the residual functional in the interval $0 < \alpha < 1$ first increases with α , and only then decreases to 0.

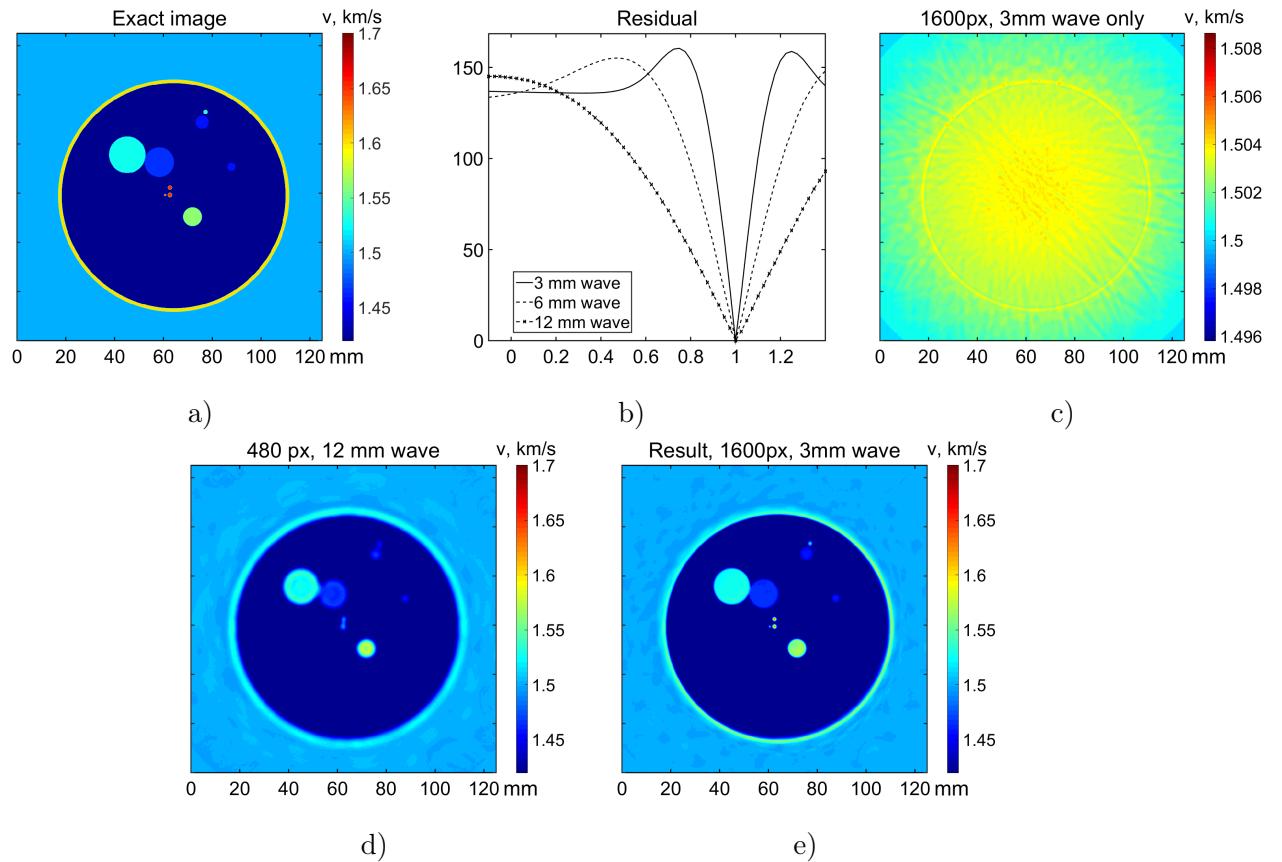


Figure 2. Numerical simulation: a – exact image (phantom); b – plots of the residual functional (4) for wavelengths $\lambda = 3$ mm, 6 mm and 12 mm; c – an image reconstructed using a wavelength $\lambda = 3$ mm from an initial approximation $c_0 = \text{const}$, d – an image reconstructed using a wavelength $\lambda = 12$ mm from an initial approximation $c_0 = \text{const}$, e – an image reconstructed via the multistage method

This illustrates the idea that the iterative gradient descent process for these wavelengths stops at a local minimum of the residual functional if the iterative process is started from an initial approximation of $c(\mathbf{r}) = c_0$. For the wavelength $\lambda = 12$ mm, the residual functional decreases monotonically to 0 as the parameter α changes from 0 to 1. This result allows us to assume that the initial approximation $c(\mathbf{r}) = c_0$ lies in the vicinity of the global minimum of the residual functional if the mean wavelength equals to 12 mm or longer. The global minimum is reachable from the initial approximation via the gradient descent method in this case.

To ensure the convergence of the iterative gradient descent process, a five-stage method has been used. For $\lambda = 12$ mm, the iterative process was started from an initial approximation of $c(\mathbf{r}) = c_0$. The resulting approximate solution was used as an initial approximation for the second stage with a mean wavelength $\lambda = 8$ mm, and so on. The result of the iterative process for $\lambda = 4$ mm was used as an initial approximation for the last stage with $\lambda = 3$ mm. The difference between wavelengths should be sufficiently small for the global minimum to be reachable from the initial approximation via the gradient descent method at each stage.

Figure 2c shows the image reconstruction results for a sounding pulse wavelength of $\lambda = 3$ mm, where $c_0 = \text{const}$ is chosen as the initial approximation for the iterative process. In this case, the iterative process stops at a local minimum, and the resulting solution is very different from the original image. Figure 2d shows the image reconstruction results for a sounding

pulse wavelength of $\lambda = 12$ mm; $c_0 = \text{const}$ is chosen as the initial approximation. In this case, the initial approximation lies in the vicinity of the global minimum, and an approximate solution to the inverse problem is obtained. However, due to the large wavelength, the spatial resolution of the resulting image is rather low.

Figure 2e shows the image reconstruction results using the multistage method with 5 stages. The central wavelength at the last stage is $\lambda = 3$ mm. The image obtained at the first stage for $\lambda = 12$ mm (shown in Fig. 2d) was used as an initial approximation for the iterative process using a wavelength $\lambda = 8$ mm at the second stage, and so on for wavelengths of 6 mm, 4 mm and 3 mm. The multistage method made it possible to avoid the iterative process stopping at a local minimum and to obtain the resulting high-quality image.

It turns out that the MSM method not only provides convergence to an approximate solution to the problem, but also significantly reduces the computation time. This issue will be discussed in more detail in Section 6.

3. Numerical Approximation of the Wave Equation

Finite-difference time-domain method (FDTD) was employed to solve equations (1)–(2). We define a uniform rectangular finite difference grid: $x_i = ih, y_j = jh, t_k = k\tau; i, j = 1, \dots, N, k = 1, \dots, M$, where h is the spatial discretization step, and τ is the time step. A second-order finite difference scheme approximates equation (1):

$$c_{ij} \frac{u_{ij}^{k+1} - 2u_{ij}^k + u_{ij}^{k-1}}{\tau^2} - \frac{\mathbf{L}_{ij}^k}{h^2} = 0.$$

Here, $u_{ij}^k = u(x_i, y_j, t_k)$ are the values of $u(\mathbf{r}, \mathbf{q}, t)$ at point (i, j) at the time step k for a fixed \mathbf{q} ; c_{ij} and a_{ij} are the values of $c(\mathbf{r})$ and $a(\mathbf{r})$ at point (i, j) . The first term approximates $c(\mathbf{r})u_{tt}(\mathbf{r}, \mathbf{q}, t)$, the second term approximates $a(\mathbf{r})u_t(\mathbf{r}, \mathbf{q}, t)$. The discrete Laplacian is denoted by \mathbf{L}_{ij}^k . A fourth-order numerical approximation [31] on a 5×5 -point stencil is used for the discrete Laplacian:

$$\mathbf{L}_{ij}^k = \sum_{m=i-2}^{i+2} \sum_{n=j-2}^{j+2} v_{mn} u_{mn}^k. \quad (8)$$

The parameters h and τ are related by the Courant stability condition $c^{-0.5}\tau < h/\sqrt{2}$. For the problem considered, we used a time step equal to $\tau = 0.3c_0^{0.5}h$, which ensured the stability of the finite difference method. The number of operations required to compute a wave propagation simulation is proportional to $O(N^3)$, where N is the number of grid points along spatial dimensions. The number of points N is chosen so that the precision of the wave simulation for the selected wavelength range is sufficient. Thus, computational complexity of the numerical method scales as a third power of wave frequency and spatial image resolution.

An approximate solution to the inverse problem is obtained via an iterative gradient descent method. Each iteration involves solving direct (1)–(2) and conjugate (6)–(7) problems in order to compute the gradient of the residual functional, which requires simulating the wave propagation process in forward and reverse time.

The numerical method was implemented in software for GPU and SIMD-capable CPU computing platforms. The discrete Laplacian computation (8) is the most compute-intensive operation in this method. The flowchart of the SIMD algorithm for computing the discrete Laplacian

is shown in Fig. 3. The Laplacian is spherically symmetrical, which makes it less computationally expensive than a convolution problem in a general case.

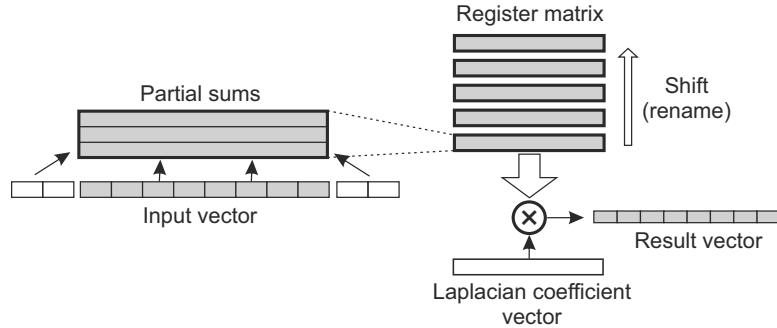


Figure 3. SIMD Discrete Laplacian computation algorithm

Y-marching method was employed to compute the convolution using registers for temporary storage. The results are calculated sequentially in vertical direction. Using the input data vector and horizontally adjacent cells, partial sums are computed and stored in the register matrix, which contains the data for 5 lines of the image. The result vector is computed by multiplying the register matrix by the Laplacian coefficient vector. The algorithm advances to the next line by shifting the lines in the register matrix up and reloading the last line from the input vector. The data is shifted via renaming the registers.

Modern processors (AVX, AVX-512, ARM NEON-class FPUs) typically have 32 SIMD registers, each of which holds a vector of 8 32-bit floating point elements for AVX FPU, 16 elements for AVX-512 FPU and 4 elements for ARM NEON FPU. There are three partial sums per line and five lines in the register matrix; thus, the input vector can be two registers long for the single wave simulation ($u(\mathbf{r}, \mathbf{q}, t)$ for the direct problem (1)–(2)) and one register long for the dual wave simulation ($u(\mathbf{r}, \mathbf{q}, t)$ and $w(\mathbf{r}, \mathbf{q}, t)$ for the conjugate problem (6)–(7)).

The computations on multi-core CPUs were parallelized using OpenMP. MPI interface was used for data exchange between computing nodes (CPU sockets or GPU devices). Figures 4a, b illustrate the order of computations for multi-core CPUs and GPUs, respectively.

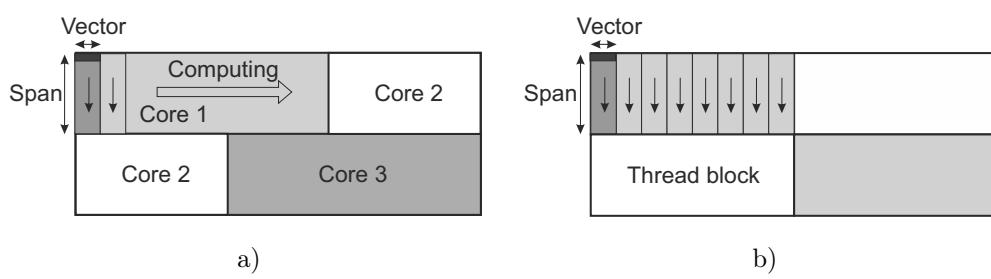


Figure 4. Parallelizing the computations on multi-core processors (a) and GPU (b)

For efficient use of cache memory, the length of the vertical segment (“Span”) of the Y-marching method was limited to some specified value. The span length typically ranges from 5 to 40 pixels. On multi-core CPUs, individual spans are computed in sequence along the horizontal dimension. The data of the previous span remains in the cache memory and is used to compute a part of the next span. The optimal span length depends on the image size and CPU cache properties and can be determined for each system via performance tests. The better the data fits into the cache, the larger span lengths are preferred. An equal amount of data is distributed to

each computing core. For GPU, the computations are performed in parallel within each thread block. The thread block size can be adjusted for better performance.

4. Parallel Implementation of the Inverse Problem Solution Algorithm

Figure 5a illustrates the direct problem solution algorithm. The algorithm simulates the wave field propagating through an inhomogeneous medium. A predefined numerical phantom simulates the object being imaged. The phantom specifies the speed of sound $c(\mathbf{r})$ in the imaging plane. Ultrasound emitters and detectors are placed in a circular formation around the phantom, as shown in Fig. 1.

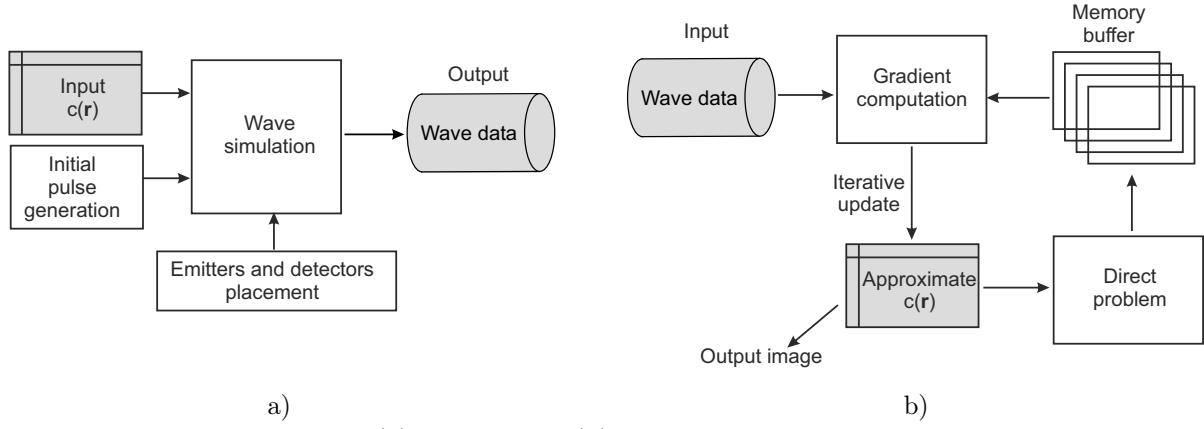


Figure 5. Direct (a) and inverse (b) problem solution algorithms

The wave field is simulated sequentially in time, starting from the initial pulse that is computed as a spherical wave radiating from the emitter position. The resulting wave field at the detector positions for each emitter is recorded in the output data.

Figure 5b illustrates the inverse problem solution algorithm. An approximate solution to the inverse problem of wave tomography is computed via the iterative gradient descent method. An initial approximation of $c(\mathbf{r})$ coefficients is set at the beginning of the iterative process. The direct problem is solved for the current approximation of $c(\mathbf{r})$.

The boundary values of computed $u(\mathbf{r}, \mathbf{q}, t)$ wave field are stored in the memory buffer. The buffer is used to reverse the wave propagation direction of $u(\mathbf{r}, \mathbf{q}, t)$ in formula (5). At the gradient computation stage (Fig. 5b) the data from the buffer are applied to the boundary of the computational domain in reverse time in order to compute $u(\mathbf{r}, \mathbf{q}, t)$ in reverse time simultaneously with $w(\mathbf{r}, \mathbf{q}, t)$.

The $w(\mathbf{r}, \mathbf{q}, t)$ wave is computed from $u(\mathbf{r}, \mathbf{q}, t)$ and the input wave data, and the gradient is computed using formula (5). The current approximate solution is updated by adding the computed gradient to the $c(\mathbf{r})$ coefficient array, and the process is repeated. The iterative process continues until the residual functional ceases to decrease. At the end of the process, the resulting approximate solution is the output of the inverse problem solution algorithm.

The iterative gradient descent method permits parallelizing the computations contained within a single iteration.

Figure 6a illustrates the parallelized computing process. Computation of the gradient of the residual functional can be subdivided into independent sub-tasks for each ultrasound emitter.

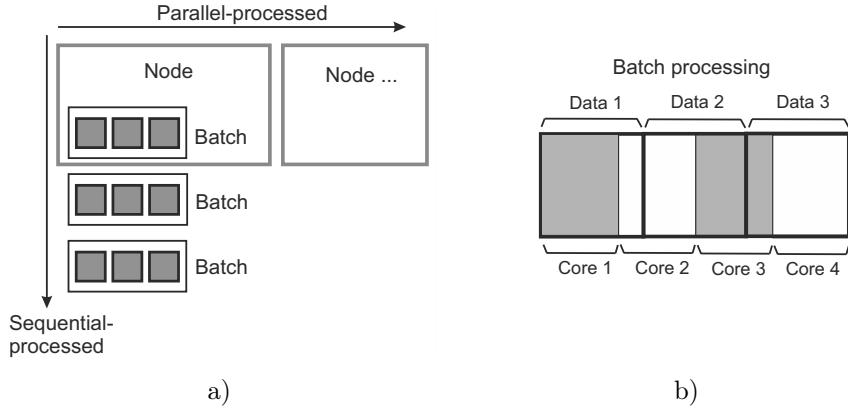


Figure 6. Parallelizing the computations on multi-core processors (a) and GPU (b)

The total number of emitters in wave tomography typically ranges from 10 to 100. The emitters are divided evenly between the computing nodes.

For each node, the computations are grouped into one or more batches executed sequentially. A batch consists of the data for several ultrasound emitters that are processed in parallel. For GPUs, the batch size is determined by the GPU memory capacity. A typical modern GPU can process most ultrasound tomography problems within its on-board memory as a single batch. For CPUs, the batch size is optimized for maximum performance, and typically is chosen close to the CPU last-level cache size. The data for a single emitter can amount to several megabytes; thus, small batch sizes of 1–2 emitters are common for CPUs.

Figure 6b illustrates the order of computations within a batch. The computations performed for each ultrasound emitter are identical and consist of a direct problem solution (wave simulation) and a gradient computation that involves simulating two wave fields $u(\mathbf{r}, \mathbf{q}, t)$ and $w(\mathbf{r}, \mathbf{q}, t)$. The only difference between emitters is in the data contents. Thus, wave fields for multiple emitters can be processed as a single data array in order to divide the computations evenly between computing cores.

The result of each iteration is the gradient of the residual functional, which is the sum of partial gradients computed for each ultrasound emitter. The partial gradients are summed up within each node, and then summed up over nodes using MPI interface. Data exchanges between nodes occur only once per iteration and therefore do not incur any noticeable delay.

5. Computing Performance for Different Computing Platforms

The software implementation of the algorithm has been tested on multiple computing platforms: Intel Haswell-EP (14 cores, AVX2 FPU), Intel 6240R (12 cores, AVX-512 FPU), NVidia Tesla P100 and NVidia Tesla V100 GPUs.

The multistage method involves solving inverse problems of wave tomography using data with bandwidth gradually increasing from stage to stage. Since the computation time strongly depends on the finite difference grid size, at each stage of the multistage method the grid size is chosen as the smallest size that still provides sufficient accuracy of the finite difference scheme. The computation time for a multistage task is the sum of the time intervals spent on each stage. To estimate the computation time for multistage tasks, the performance of computing platforms was tested on the computations of separate iterations of the gradient descent method for various grid sizes.

CPU performance significantly depends on cache utilization, as external memory is much slower than cache memory. Thus, for optimal performance we choose the batch size (the amount of data to be processed in parallel by the CPU, Fig. 6a) close to the CPU cache size. In order to optimize the computations, performance tests were conducted to determine the optimal batch size for parallel processing on each target system. The batch sizes determined may differ from the physical CPU cache size due to the use of an additional memory buffer (Fig. 5b).

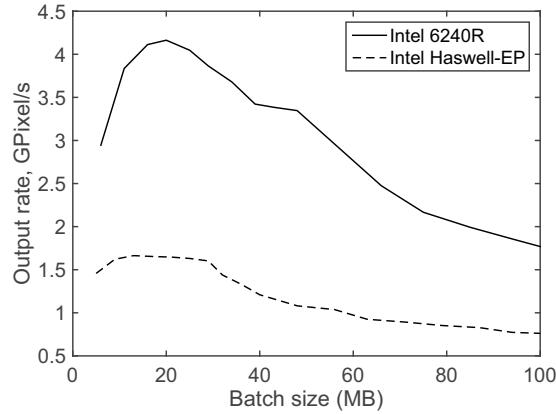


Figure 7. CPU performance depending on batch size

Figure 7 plots CPU performance depending on batch size. Output data rate is measured in the tests as the number of computed gradient pixels per second in gigapixels/s. Each pixel of a reconstructed image uses 32 bytes of data. The tests showed that using too large batch sizes results in a performance decrease due to cache misses. Too small batch sizes are also non-optimal because the task execution time for a small batch becomes very short and the thread synchronization latency becomes noticeable, especially on a fast CPU such as Intel 6240R. As a result, a batch size of 22 Mb was chosen for computations on Intel CPUs.

A single simulation frame uses $N \times N$ pixels of output, where N is the grid size along each dimension. Wave simulation requires approximately N time steps for the computed wave to reach the detectors. The wave simulation is performed for every ultrasound emitter in order to compute the gradient of the residual functional. The multistage method uses multiple grids with different resolutions. To determine the computing time for a task, we determine the computing time for a single wave simulation and multiply that by the number of emitters and the number of gradient descent iterations.

A series of tests were conducted to determine the computing time for each grid resolution. Figure 8a plots the computing devices' performance depending on the finite difference grid size. For Intel CPUs, the performance decreases for larger grids, as such grids do not fit in the CPU cache. For GPUs the performance may slightly increase with increased grid size due to more parallelism being available on larger grids. The NVidia Tesla P100 and NVidia Tesla V100 GPUs tested are equipped with sufficient amount of VRAM to process the whole inverse problem as a single batch in parallel.

Figure 8b plots the computing time per iteration per emitter corresponding to the output rate shown in Fig. 8a. The time scale is logarithmic in this plot. The number of operations to compute a wave simulation scales as a third power of the grid size. Employing smaller grids to complete the first iterations quickly in the multistage iterative method significantly decreases

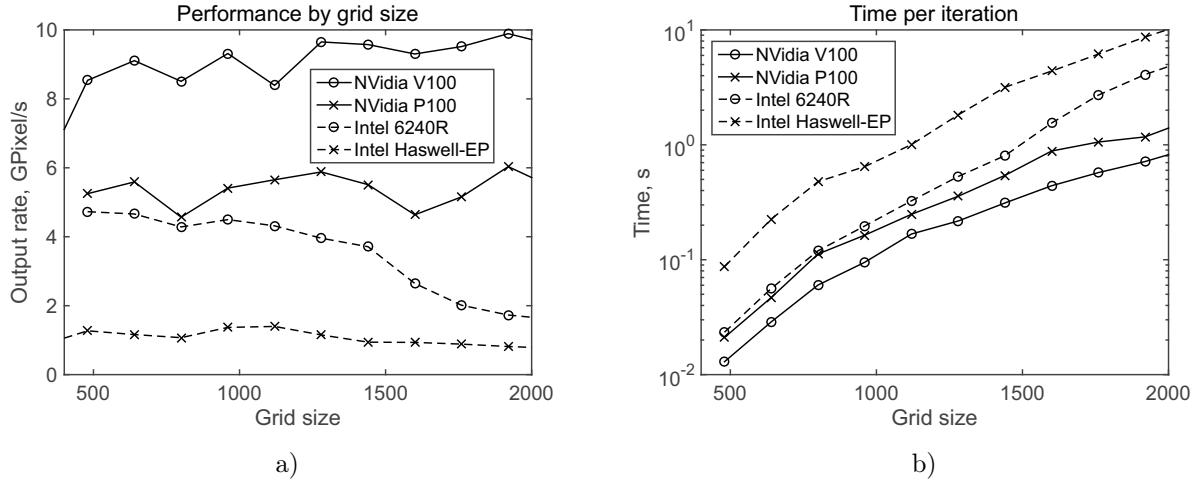


Figure 8. Device performance depending on finite difference grid size (a); corresponding computing time per iteration per emitter (b)

the total computing time compared to performing all the iterations using the highest-resolution grid.

Table 1. Multistage method parameters

Stage	1	2	3	4	5
Resolution, px	480	640	800	1200	1600
Wavelength, mm	12	8	6	4	3
Iterations	30	30	30	30	30
Time, sec	3	7	15	50	120

Numerical simulations showed that on average 30 iterations are sufficient at each stage of the multistage method to obtain an approximate solution suitable for the next stage, or to obtain the final result at the last stage. Multistage method parameters suitable for medical imaging for breast cancer diagnosis are summarized in Tab. 1. The computing time row in Tab. 1 lists the actual times achieved on a computing node of “Lomonosov-2” supercomputer [32] equipped with two NVidia Tesla V100 GPUs working in parallel.

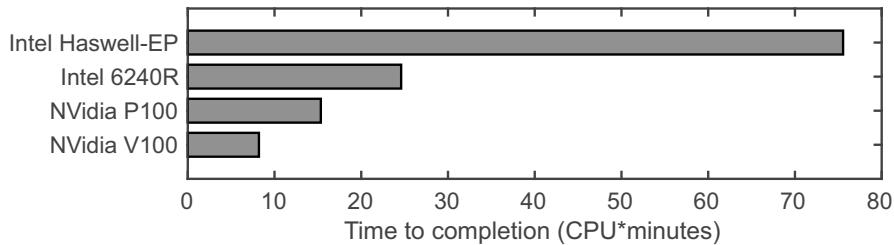


Figure 9. Overall performance of the computing devices

Using these parameters, the total computing time required to complete the inverse problem solution can be estimated for each computing device. Figure 9 shows the computing time for the devices tested. For example, to obtain 30 images per hour in a medical imaging setup (one image is a single cross-section of an object) a computing cluster of either 4 NVidia Tesla V100, 8 NVidia Tesla P100, 12 Intel 6240R or 36 Intel Haswell-EP processors would be needed.

Provided that almost all the data is cached, Intel 6240R with AVX-512 FPU is only 1.5 times behind NVidia Tesla P100 GPU. Thus, CPU or GPU clusters can be used for image reconstruction using the multistage method with image sizes up to 1600 pixels. Such grid sizes are sufficient for low-frequency wave tomography of relatively small objects, where the object size is on the order of 25–30 wavelengths. For higher frequencies or larger objects, larger grids would be required, putting CPU systems at a significant disadvantage compared to GPUs.

GPUs were found to be the preferred architecture for solving direct and inverse problems of wave tomography, especially for higher-resolution images. GPU performance on a typical FDTD algorithm is approximately proportional to its memory throughput. The numerical algorithm is data-parallel and requires neither synchronized data exchanges between computing cores nor cache coherence. Thus, the algorithm can benefit from the specific structure of graphics processors.

6. Model Problem Examples

The multistage iterative method for solving inverse problems of wave tomography proposed in this study is designed primarily to ensure convergence of the gradient iterative method to the global minimum. Approximate solutions of the inverse problem are computed using gradually increasing sounding pulse bandwidth and image resolution.

It may seem that an experimental setup with 5 different ultrasound emitters is needed to use a 5-stage iterative method in practice. However, the input data for multiple stages can be produced from a single experimental measurement via application of a low-pass filter to the broadband signal. This approach requires the signal to contain sufficiently strong low frequencies. Broadband ultrasound transducers with a usable frequency range of 100 to 600 kHz can be used for this task.

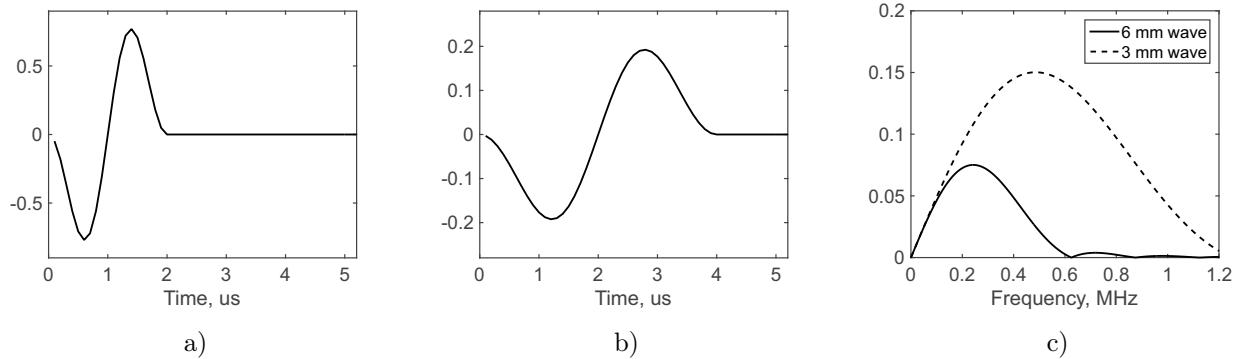


Figure 10. Sounding pulses: a – waveform with a mean wavelength of 3 mm; b – waveform with a mean wavelength of 6 mm; c – frequency spectra of the sounding pulses

Figure 10a shows the waveform of a broadband sounding pulse with a mean wavelength of 3 mm used in the presented numerical simulations at the last stage of the MSM method. Figure 10b shows the sounding pulse with a mean wavelength of 6 mm used for an intermediate stage of the MSM method. Figure 10c shows the frequency spectra of these two sounding pulses. The spectrum of the longer wave with a central frequency of 250 kHz is a part of the spectrum of the shorter wave with a central frequency of 500 kHz. A short pulse contains both low and high frequencies, lower parts of the spectrum can be filtered and used for low-resolution stages.

Lower-resolution approximations can be computed much faster; thus, the multistage iterative method can be used to improve computing time even if multiple stages are not necessary for convergence. Figure 11 presents an example of a low-contrast phantom, the image of which can be reconstructed via the gradient descent method using a wavelength of 3 mm and a constant initial approximation.

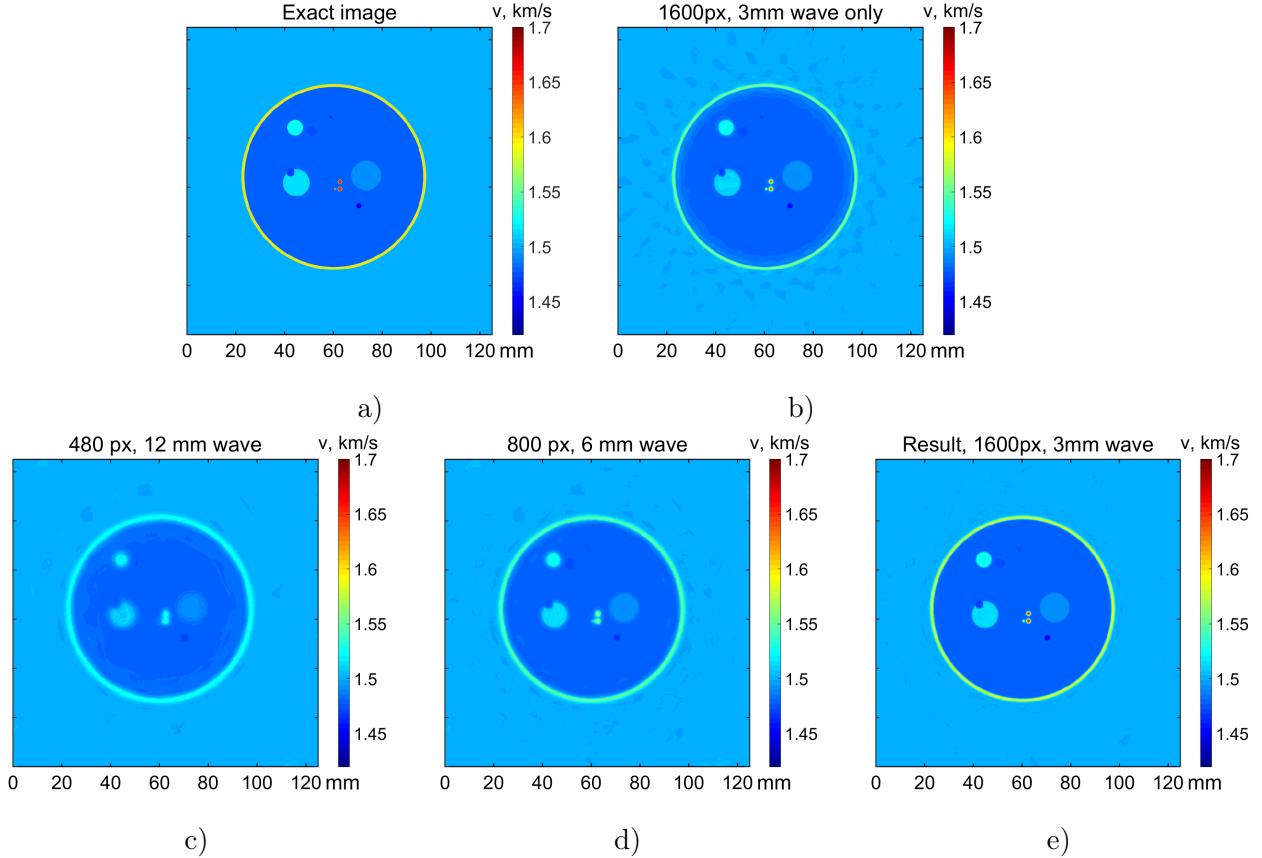


Figure 11. Single-stage and multistage reconstruction: a – exact image, b – image reconstructed at 3 mm wavelength from a constant initial approximation, c – image reconstructed at the first stage from a constant initial approximation, d – image reconstructed at the 3rd stage, e – image reconstructed using at the final stage

Figure 11a shows an exact image of the phantom. Figure 11b shows an image reconstructed using the gradient method from a constant initial approximation and with a pulse wavelength of 3 mm. Even without using the multistage method, the image is reconstructed quite accurately. However, to solve this inverse problem, 100–120 gradient descent iterations are required.

Figures 11c–e show approximate solutions obtained at various stages of the multistage method. Figure 11c shows an image reconstructed at the first stage of the multistage method from a constant initial approximation using a wavelength of 12 mm and a grid size of 480×480 points. Figure 11d shows an image reconstructed at the third stage of the multistage method using a wavelength of 6 mm and a grid size of 800×800 points. Figure 11e shows an image reconstructed at the last stage of the multistage method using a wavelength of 3 mm and a grid size of 1600×1600 points.

Figure 11 demonstrates that the image Fig. 11e obtained via the multistage method is closer to the original than the image in Fig. 11b obtained via the gradient descent method with fixed parameters and a constant initial approximation. Although the parameters of the iterative

method are the same for Fig. 11b and Fig. 11e, a better image quality in Fig. 11e is achieved due to a better initial approximation being used in the last stage of the MSM method. Instead of a constant, an image computed at the previous stage is used, which is much closer to the exact image than a constant.

In this example, the multistage method can reduce the computation time and improve the image quality. The parameters of the multistage method are listed in Tab. 1. The total number of gradient descent iterations at all stages is 150, however, only 30 of them are performed in a high resolution of 1600×1600 . These iterations are the most time-consuming. Starting from an initial approximation computed at the previous stage instead of a constant initial approximation, only 30 iterations at the highest resolution are sufficient to complete the process.

First stages of the multistage method require significantly less computation time than the last. The total computation time in this example is 195 seconds, of which the last stage takes 120 s. In 195 seconds, it is possible to perform 48 high-resolution gradient descent iterations, which is usually not enough to reconstruct an image with good accuracy. Thus, the multistage method allows us to reduce the computation time.

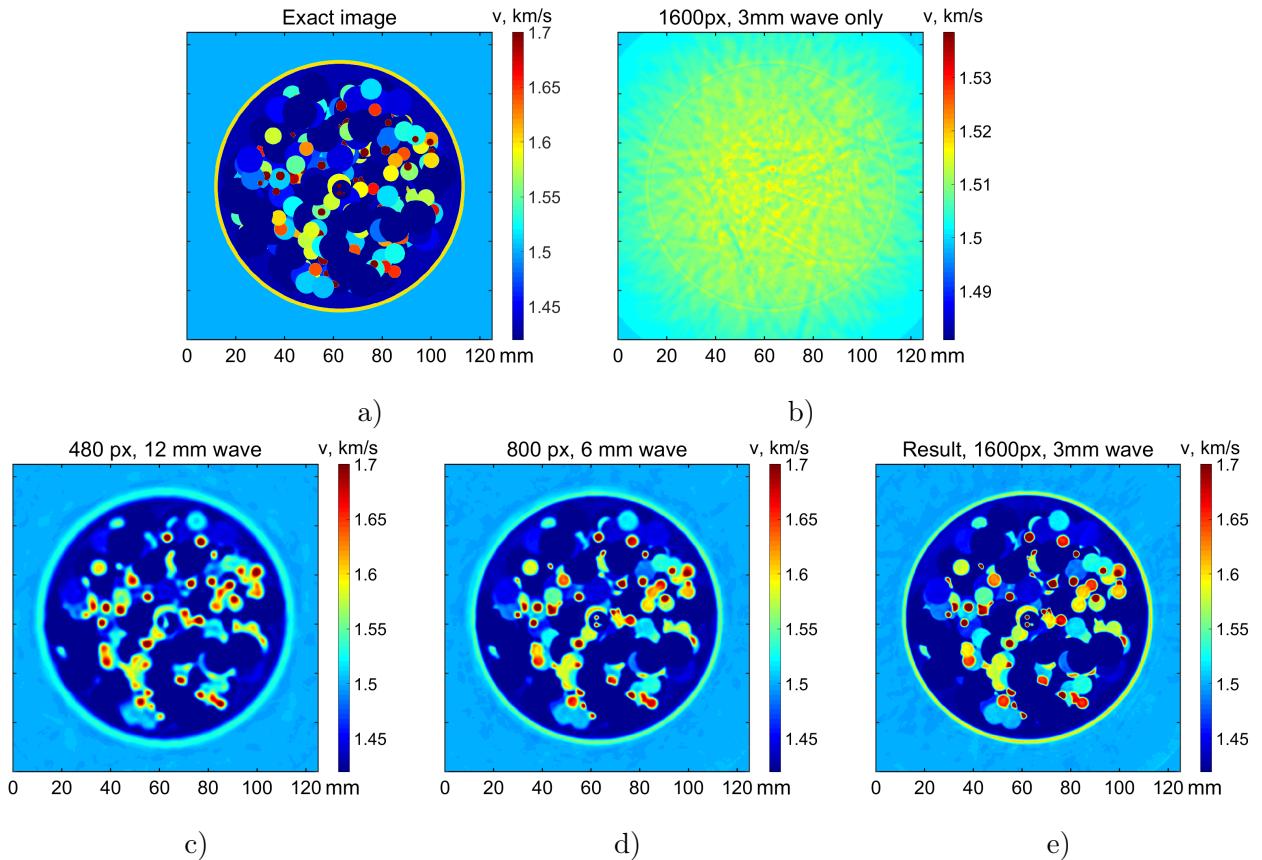


Figure 12. Complex phantom example: a – exact image, b – image reconstructed at 3 mm wavelength from a constant initial approximation, c – image reconstructed using 12 mm wavelength from a constant initial approximation, d – image reconstructed using 6 mm wavelength at the 3rd stage, e – final image after 5 stages

Figure 12 shows an example of reconstructing a complex internal structure of an object. Figure 12a shows the exact image of the model object (phantom). In this example, the gradient descent process does not converge using a constant initial approximation and a short sounding pulse with a wavelength of 3 mm. Figure 12b shows the reconstruction result for this case.

The iterative process stops at a local minimum of the residual functional and the image is not reconstructed. To obtain a tomographic image of such an object, it is necessary to apply the multistage method.

At the first stage, an approximate solution is computed using a constant initial approximation, a central wavelength of 12 mm and a grid size of 480×480 points. The result of the first stage is shown in Fig. 12c. This image has a low resolution, but is sufficiently close to the original. It is used as an initial approximation for the next stage of the multistage method. In total, 5 stages are performed according to Tab. 1. The image quality gradually improves from stage to stage. This means that the images can be analyzed before the whole reconstruction process is completed – as early as the sought-for image features are resolved. This property of the multistage method can improve image analysis time in practical applications. Highest-resolution stages take considerably more time to compute. Figure 12d shows the result of the third stage of the multistage method with an average wavelength of 6 mm and a grid size of 800×800 points. Figure 12e shows the result of the last stage with an average wavelength of 3 mm and a grid dimension of 1600×1600 points. Thus, the MSM method ensures the convergence of the iterative process of gradient-descent minimization of the residual functional and allows for high accuracy tomographic image reconstructions via wave tomography technology.

Conclusion

This article discusses the methods of ultrasound tomography, which can be used to inspect various objects, for example, in nondestructive testing or in medical imaging such as tomographic imaging of soft tissues for early-stage breast cancer diagnosis. A characteristic feature of ultrasound tomography is that it takes into account not only the reflected radiation, but also the radiation transmitted through the object, similarly to X-rays. It is well known that the higher the frequency, the higher the spatial resolution can be achieved in ultrasound imaging. Conventional medical ultrasound diagnostic devices usually employ frequencies above 1 MHz. The tomographic methods considered in this study use low frequencies in the 100–600 kHz band for medical ultrasound tomography. The proposed MSM method uses low frequencies for initial stages of the method to ensure its convergence and high frequencies for final stages to achieve high resolution.

The article discusses a supercomputer implementation of the multistage iterative method (MSM) for solving nonlinear inverse problems of ultrasound tomography. From a mathematical point of view, the problem is posed as a problem of minimizing the residual functional, which is not convex and has local minima. The method is based on a prior information which is typical for most inverse problems of wave tomography. The effectiveness of the MSM method is illustrated on a large number of model problems focused on ultrasound tomographic diagnostics of soft tissues. As shown in the article, the MSM method completely covers the problem of constructing an approximate solution in the problems of medical ultrasound tomographic diagnostics of soft tissues.

In contrast to previous works [33, 34], this article considers the inverse problem of ultrasonic tomography in the wave model without taking into account the absorption of the medium. The numerical algorithms have been adapted for SIMD-capable CPUs. The performance of computing systems is compared for different sizes of the computational grid and for a multistage run. To ensure the convergence of the iterative process of solving the inverse problem, filtering the spectrum of a broadband sounding pulse is proposed. This approach greatly simplifies the exper-

iment compared to using multiple narrow-band emitters. In this study, a significant acceleration of the algorithm has been achieved due to the use of smaller grids for lower frequencies. The method has been tested on phantoms of a complex structure close to reality.

Both CPU- and GPU-based computing clusters can be used to implement the MSM method in practice. It is shown that GPU supercomputers have an advantage, especially for large volumes of data. The numerical algorithm is data-parallel and well-suited for GPU architecture. Modern processors equipped with AVX-512 FPUs are capable of solving small-scale tasks that fit in the CPU cache memory. For large-scale tasks, it is always better to use a GPU equipped with fast on-board memory.

Recently, a new processor architecture has been developed, consisting of several hundred thousand computing cores located on a single silicon wafer. For problems of wave tomography, this line of work is also of great interest. Wafer-scale processors manufactured by Cerebras are designed for machine learning tasks. A specific feature of such tasks is a large number of operations being performed on a fixed data array. Wave tomography problems have the same property – the operation of wave propagation simulation is performed many times on the same data array in the iterative process. On an actual problem of physical process simulation, the wafer-scale system has achieved a performance of 860 TFlops [35]. To date, the wafer-scale architecture has the highest performance-to-memory ratio and is thus the most promising architecture for implementing large-scale wave tomography applications.

To conclude, we note that the use of supercomputer technologies for solving wave tomography problems opens up the possibility of using complex mathematical models describing such physical processes as diffraction, refraction, multiple scattering, and so on. Under such models, the inverse problems are nonlinear and the methods developed in this study can be used to solve such problems. First of all, the developed solution methods focus on the problems of medical ultrasound tomography and inverse problems of electromagnetic sounding.

The paper discusses the possibilities of solving inverse problems of wave tomography in the framework of scalar wave models. From a physical point of view, only one type of wave propagates in a scalar medium – longitudinal compression wave. This model is ideal, for example, for ultrasound tomographic screening for breast cancer diagnosis. However, the scalar model is no longer adequate to reality in such cases as, for example, obtaining ultrasound tomographic images of a knee joint. Unlike in soft tissues, multiple types of waves propagate in solid bodies. There are works concerned with solving direct problems of wave propagation in solids. In some works, attempts are made to solve inverse problems of ultrasound imaging in solids [36, 37]. These problems typically arise in the field of nondestructive testing. Solving inverse problems in vector models is a much more complicated task, compared to scalar models. Solving such problems is impossible without the use of supercomputers.

Acknowledgements

The paper was published with the financial support of the Ministry of Education and Science of the Russian Federation as part of the program of the Moscow Center for Fundamental and Applied Mathematics under the agreement No. 075-15-2019-1621. 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. Radon, J.: Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten. Berichte über die Verhandlungen der Königlich-Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse [Reports on the Proceedings of the Royal Saxonian Academy of Sciences at Leipzig, Mathematical and Physical Section], Leipzig: Teubner 69, 262–277 (1917)
2. Tikhonov, A.N.: Solution of incorrectly formulated problems and the regularization method. Soviet. Math. Dokl. 4, 1035–1038 (1963)
3. Tikhonov, A.N.: Regularization of incorrectly posed problems. Soviet Math. Dokl. 4, 1624–1627 (1963)
4. Tikhonov, A.N., Goncharsky, A.V., Stepanov, V.V., Yagola, A.G.: Numerical Methods for the Solution of Ill-Posed Problems. Springer, Dordrecht (1995). <https://doi.org/10.1007/978-94-015-8480-7>
5. Bakushinsky, A., Goncharsky, A.: Ill-Posed Problems: Theory and Applications. Kluwer Academic Publishers. Springer, Dordrecht (1994). <https://doi.org/10.1007/978-94-011-1026-6>
6. Vinokurov, V. A.: Regularizability of Functions. In: Ill-posed problems in the Natural Sciences, pp. 52–70, Mir, Moscow (1987)
7. Lavrentiev, M.M., Romanov, V.G., Shishatskii, S.P.: Ill-Posed Problems of Mathematical Physics and Analysis. American Mathematical Society, Providence, (1986)
8. Ramm, A.G.: Non-uniqueness of the solution to an inverse problem in geophysics. Inverse problems 2, 123–125 (1986)
9. Engl, H.W., Kunish, K., Neubaer, A.: Convergence rate for Tikhonov regularization of non-linear ill-posed problem. Inverse problems 4, 532–540 (1989)
10. Groetch, C.W.: The theory of Tikhonov regularization for Fredholm equations of the first kind. SIAM Review 28, 116–118 (1986). <https://doi.org/10.1137/1028033>
11. Nashed, M.Z.: Ill-posed Problems: Theory and Practice. Reidel, Dordrecht (1985)
12. Klibanov, M.V., Timonov, A.A.: Carleman Estimates for Coefficient Inverse Problems and Numerical Applications. Walter de Gruyter GmbH (2004). <https://doi.org/10.1515/9783110915549>
13. Romanov, V.G., Kabanikhin, S.I. Inverse Problems for Maxwell's Equations. VSP, Utrecht, (1994)

14. Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Inverse problems of 3D ultrasonic tomography with complete and incomplete range data. *Wave Motion* 51(3), 389–404 (2014). <https://doi.org/10.1016/j.wavemoti.2013.10.001>
15. Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: Comparison of the capabilities of GPU clusters and general-purpose supercomputers for solving 3D inverse problems of ultrasound tomography. *Journal of Parallel and Distributed Computing* 133, 77–92 (2019). <https://doi.org/10.1016/j.jpdc.2019.06.008>
16. Natterer, F., Sielschott, H., Dorn, O., et al.: Frechet derivatives for some bilinear inverse problems. *SIAM J. Appl. Math.* 62, 2092–2113 (2002). <https://doi.org/10.1137/s0036139901386375>
17. Beilina, L., Klibanov, M.V., Kokurin, M.Y.: Adaptivity with relaxation for ill-posed problems and global convergence for a coefficient inverse problem. *J. Math. Sci.* 167, 279–325 (2010). <https://doi.org/10.1007/s10958-010-9921-1>
18. Goncharsky, A.V., Romanov, S.Y.: Iterative Methods for Solving Coefficient Inverse Problems of Wave Tomography in Models with Attenuation. *Inverse Probl.* 33(2), 025003 (2017). <https://doi.org/10.1088/1361-6420/33/2/025003>
19. Global Optimization: From Theory to Implementation. Edition by Liberti, L., Maculan, N. Springer (2006)
20. Gel'fand, I.M., Tsetlin, M.L.: Some methods of control for complex systems. *Russian Mathematical Surveys* 17, 95–117 (1962). <https://doi.org/10.1070/rm1962v01n01abeh001124>
21. Sulimov, A.V., Zheltkov, D.A., Oferkin, I.V., et al.: Tensor Train Global Optimization: Application to Docking in the Configuration Space with a Large Number of Dimensions. *Communications in Computer and Information Science (CCIS)*, vol. 793, pp. 151–167. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71255-0_12
22. Duric, N., Littrup, P., Li, C., et al.: Breast ultrasound tomography: bridging the gap to clinical practice. *Proc. SPIE*, 8320, 83200O (2012). <https://doi.org/10.1117/12.910988>
23. Jirik, R., Peterlik, I., Ruiter, N., et al.: Sound-speed image reconstruction in sparse-aperture 3D ultrasound transmission tomography. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control.* 59, 254–264 (2012). <https://doi.org/10.1109/tuffc.2012.2185>
24. Goncharsky, A.V., Romanov, S.Y., Seryozhnikov, S.Y.: A computer simulation study of soft tissue characterization using low-frequency ultrasonic tomography. *Ultrasonics* 67, 136–150 (2016). <https://doi.org/10.1016/j.ultras.2016.01.008>
25. Shen, Y., Shamout, F.E., Oliver, J.R., et al.: Artificial intelligence system reduces false-positive findings in the interpretation of breast ultrasound exams. *Nat. Commun.* 12, 5645 (2021). <https://doi.org/10.1038/s41467-021-26023-2>
26. Natterer, F.: Sonic imaging. In: Scherzer O. (eds) *Handbook of Mathematical Methods in Imaging*, pp. 1253–1278. Springer, New York (2015). https://doi.org/10.1007/978-1-4939-0790-8_37

27. Bakushinsky, A., Goncharsky, A., Romanov, S., Seatzu, S.: On the identification of velocity in seismics and in acoustic sounding. *Pubblicazioni IAGA, Series “Problemi non ben posti e inversi”* 71 (1994)
28. Bakushinskii, A.B., Kozlov, A.I., Kokurin, M.Yu.: One Inverse Problem for a Three-Dimensional Wave Equation. *Computational Mathematics and Mathematical Physics* 43(8), 1149–1158 (2003)
29. Bakushinskii, A.B., Leonov, A.S.: Low-cost numerical method for solving a coefficient inverse problem for the wave equation in three-dimensional space. *Computational Mathematics and Mathematical Physics* 58, 548–561 (2018). <https://doi.org/10.1134/s0965542518040073>
30. Klibanov, M.V., Li, J., Zhang, W.: Linear Lavrent’ev Integral Equation for the Numerical Solution of a Nonlinear Coefficient Inverse Problem. *SIAM J. Appl. Math.* 81(5), 1954–1978 (2021). <https://doi.org/10.1137/20M1376558>
31. Hamilton, B., Bilbao, S.: Fourth-order and optimised finite difference schemes for the 2-D wave equation. In: Proc. of the 16th Int. Conference on Digital Audio Effects (DAFx-13). pp. 363–395. Springer (2013)
32. Voevodin, V., Antonov, A., Nikitenko, D., et al.: Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. *Supercomputing Frontiers and Innovations* 6(2), 4–11 (2019). <https://doi.org/10.14529/jsfi190201>
33. Goncharsky, A., Seryozhnikov, S.: Supercomputer technology for ultrasound tomographic image reconstruction: mathematical methods and experiment. In: Voevodin, V., Sobolev, S. (eds) *Supercomputing. RuSCDays 2018. Communications in Computer and Information Science*, vol. 965, pp. 401–413. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05807-4_34
34. Romanov, S. Supercomputer simulation study of the convergence of iterative methods for solving inverse problems of 3D acoustic tomography with the data on a cylindrical surface // In: Voevodin, V., Sobolev, S. (eds) *Supercomputing. RuSCDays 2018. Communications in Computer and Information Science*, vol. 965, pp. 388–400. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05807-4_33
35. Rocki, K., Essendelft, D.V., Sharapov, I.: Fast stencil-code computation on a wafer-scale processor. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (2020)
36. Lechleiter, A., Schlasche, J.W.: Identifying Lame parameters from time-dependent elastic wave. *Inverse Problems in Science and Engineering* 25(1), 2–26 (2017). <https://doi.org/10.1080/17415977.2015.1132713>
37. He, J., Rao, J., Fleming, J.D., et al.: Numerical ultrasonic full waveform inversion (FWI) for complex structures in coupled 2D solid/fluid media. *Smart Materials and Structures* 30, 085044 (2021). <https://doi.org/10.1088/1361-665X/ac0f44>