fractal and fractional

MDPI

Article

Parallel Algorithm for Solving the Inverse Two-Dimensional Fractional Diffusion Problem of Identifying the Source Term

Elena N. Akimova [1,2], Murat A. Sultanov [3,*], Vladimir E. Misilov [1,4] and Yerkebulan Nurlanuly [3]

[1] Ural Branch of RAS, Krasovskii Institute of Mathematics and Mechanics, S. Kovalevskaya Street 16, Ekaterinburg 620108, Russia; aen15@yandex.ru
[2] Department of Information Technologies and Control Systems, Institute of Radioelectronics and Information Technology, Ural Federal University, Mira Street 19, Ekaterinburg 620002, Russia
[3] Department of Mathematics, Faculty of Natural Science, Khoja Akhmet Yassawi International Kazakh-Turkish University, Turkistan 160200, Kazakhstan; yerkebulan.nurlanuly@ayu.edu.kz
[4] Department of High Performance Computing Technologies, Institute of Natural Sciences and Mathematics, Ural Federal University, Mira Street 19, Ekaterinburg 620002, Russia; v.e.misilov@urfu.ru
* Correspondence: murat.sultanov@ayu.edu.kz

Abstract: This paper is devoted to the development of a parallel algorithm for solving the inverse problem of identifying the space-dependent source term in the two-dimensional fractional diffusion equation. For solving the inverse problem, the regularized iterative conjugate gradient method is used. At each iteration of the method, we need to solve the auxilliary direct initial-boundary value problem. By using the finite difference scheme, this problem is reduced to solving a large system of a linear algebraic equation with a block-tridiagonal matrix at each time step. Solving the system takes almost the entire computation time. To solve this system, we construct and implement the direct parallel matrix sweep algorithm. We establish stability and correctness for this algorithm. The parallel implementations are developed for the multicore CPU using the OpenMP technology. The numerical experiments are performed to study the performance of parallel implementations.

Keywords: time-fractional diffusion equation; Caputo fractional derivative; inverse problems; source term identification; finite-difference scheme; block-elimination method; parallel matrix sweep method; parallel computing

Citation: Akimova, E.N.; Sultanov, M.A.; Misilov, V.E.; Nurlanuly, Y. Parallel Algorithm for Solving the Inverse Two-Dimensional Fractional Diffusion Problem of Identifying the Source Term. *Fractal Fract.* **2023**, *7*, 801. https://doi.org/10.3390/fractalfract7110801

Academic Editors: Haci Mehmet Baskonus, Boying Wu and Xiuying Li

Received: 11 August 2023
Revised: 14 October 2023
Accepted: 30 October 2023
Published: 2 November 2023

1. Introduction

Numerous physical phenomena exhibit the characteristics of memory retention and nonlocal effects [1]. The formulation of mathematical frameworks to describe these phenomena often involves the utilization of fractional calculus [2]. Various fractional derivative operators possess distinct definitions and inherent properties. The fractional differential equation may be applied to a wide range of fields such as anomalous diffusion [3–5], viscoelasticity [6], ferroelectric media [7], fractional multi-pole and neuron modelling [8], and fractional Lèvy motion [9]. The review paper [10] presents a comprehensive survey on real-world applications of fractional calculus in various fields, namely, physics; control, signal and image processing; mechanics and dynamic systems; biology; environmental science; material studies; economics; and engineering.

The solution to direct and inverse problems for differential equations with fractional derivatives typically incurs substantial computational costs due to their nonlocal characteristics. Different numerical techniques are available for solving approximate initial-boundary problems for fractional differential equations [11–14], for example, the finite difference method. An established approach to enhance computational efficiency involves the utilization of parallel computing [15–17]. The problems for fractional differential equations may be solved using various original parallel algorithms [18,19].

Additionally, while forward problems are usually well developed, the inverse problems may present significant challenges to achieve the stable solutions [20]. Here, by

direct problems, we mean the classical initial boundary problem of finding the unknown function from equation and additional boundary and initial conditions. Inverse problems include identifying unknown coefficients of an equation or unknown boundary or initial conditions [21]. In this case, a priori information is used to ensure the uniqueness of the solution. Such tasks are often incorrect. Regularization methods are used to achieve correctness. Hence, the development and implementation of parallel algorithms aimed at solving inverse problems emerging from fractional differential equations is of great importance.

In [22], an algorithm for solving the inverse problem of identifying the space-dependent source is constructed on the base of regularized Landweber iterative method. An iterative algorithm on the basis of the conjugate gradient method is constructed in [23]. For smoothing the values, the authors used the Savitzky–Golay filter.

In [24], existence, uniqueness, and stability estimates are established for the inverse source problem of the recovery of a space-dependent source term for a generalized subdiffusion equation. The Tikhonov regularization method was proposed for solving the problem of reconstruction of a time-dependent source term in a time-fractional diffusion-wave equation [25]. Convergence estimates and parameted choice rules for Tikhonov regularization for the inverse source problem for a time fractional diffusion equation were proposed in [26].

In our previous works [27,28] parallel algorithms for solving direct and inverse problems for one-dimensional time-fractional diffusion equation are constructed on the basis of the parallel sweep method for solving the systems of linear algebraic equations with tridiagonal matrices.

The goal of our work is to construct and implement an efficient algorithm for solving the inverse problem of identifying the stationary source term for the two-dimensional time-fractional diffusion equation. To solve the inverse problem, we apply the regularized iterative conjugate gradient method. It requires solving the auxilliary direct subproblem at each iteration. By using the finite difference scheme, we reduce the direct problem to solving a series of systems of linear algebraic equations (SLAE) with block tridiagonal matrices. Note that solving these SLAEs for the two-dimensional fractional diffusion problem takes most of the computation time (up to 99% of the total computational time). We construct and establish the stability and correctness of the direct parallel matrix sweep method for solving such systems. We have developed a parallel algorithm and a parallel code for solving the inverse problem. The code is intended for multicore processors and is implemented using C++17 and OpenMP 4.0 extension. In order to evaluate the validity of developed numerical methods and the performance of parallel algorithms, numerical experiments are conducted. In the future, we plan to use our algorithm and code for solving applied problems.

Below is a breakdown of the article's structure. In Section 2, we formulate the direct and inverse problems and introduce the discretization and difference scheme for solving the direct initial-boundary value problem. We demonstrate the block-tridiagonal structure of the coefficient matrix of the SLAE. In Section 3, we describe the direct methods for solving SLAEs with the block-tridiagonal matrix. We establish the stability and correctness of the parallel matrix sweep algorithm. In Section 4, we describe the conjugate gradient algorithm for solving the inverse problem. Section 5 describes the development of the parallel code that implements the numerical algorithms described above. Section 6 presents the results of the performed numerical experiments. In Section 7, we discuss these results. Section 8 concludes our work.

2. Problem

2.1. Statement of the Problem

Consider the basis time-fractional elliptic partial differential equation in the following form:

$$\frac{\partial^\alpha U}{\partial t^\alpha} + \mathcal{L}U = \psi(\mathbf{x})\eta(\mathbf{x}, t), \quad \mathbf{x} \in \Omega, \quad 0 < t \leq T. \tag{1}$$

Here, $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \overline{\Omega} = \prod_{i=1}^{d} [0, \gamma_i]$, $0 < \alpha < 1$, $q(x,t) \neq 0$ and $\mathcal{L}$ is an elliptic operator

$$\mathcal{L}U = -\sum_{i=1}^{d} \frac{\partial}{\partial x_i}\left(k_i(\mathbf{x},t)\frac{\partial U}{\partial x_i}\right), \quad x_i \in (0,\gamma_i), \quad 0 < t \leq T.$$

The boundary condition is

$$U(\mathbf{x},t) = 0, \quad \mathbf{x} \in \delta\Omega. \tag{2}$$

The initial condition is

$$U(\mathbf{x},0) = g_0(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{3}$$

where $g_0(\mathbf{x})$ is the given functions.

For simplicity, we have formulated our fractional diffusion equation with homogeneous Dirichlet boundary conditions. Note that the problem with inhomogeneous boundary conditions may be reduced to a problem with homogeneous conditions. To do this, we need to represent function U as the sum of a some function U_0 that satisfies the inhomogeneous boundary conditions (for example, found by solving the Dirichlet problem for the Laplace's equation) plus a remainder function V that will satisfy the homogeneous conditions $U(\mathbf{x},t) = U_0(\mathbf{x},t) + V(\mathbf{x},t)$. The algorithms and approaches presented below may be utilized for more general formulations with the Neumann or mixed boundary conditions.

For this study, we consider the Caputo fractional derivative with order α in the form [29]

$$D^{\alpha} f(x) = \frac{1}{\Gamma(m-\alpha)} \int_0^x \frac{f^{(m)}(t)}{(x-t)^{\alpha-m+1}} dt,$$

with $\alpha \in (m-1, m)$, $m \in N$, $x > 0$.

Assuming that the solution $U(\mathbf{x},t)$ exists and satisfies the Dirichlet boundary conditions, for the case of $0 < \alpha < 1$ ($m = 1$) we can consider the following formula for the Caputo fractional partial derivative:

$$\frac{\partial^{\alpha} U(\mathbf{x},t)}{\partial t^{\alpha}} = \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{\partial U(\mathbf{x},s)}{\partial s} (t-s)^{-\alpha} ds. \tag{4}$$

The direct initial boundary problem consists in finding the unknown function $U(\mathbf{x},t)$ when all other components of Equation (1) are known.

In the present work, we study the inverse problem of restoring the space-dependent right-hand part $\psi(\mathbf{x})$. Thus, the problem consists in finding the pair of unknown functions $[U(\mathbf{x},t), \psi(\mathbf{x})]$. Additional information for the inverse problem is given in the form of final overdetermination

$$U(\mathbf{x},T) = \varphi(\mathbf{x}), \mathbf{x} \in \Omega. \tag{5}$$

Conditions for the uniqueness of the solution of this inverse problem for a general multi-dimensional equation are formulated in [30].

2.2. Discretization of Equation and Difference Scheme

In this paper, for simplicity, we will consider the case of $d = 2$ and $k_1 = k_2 = 1$. For an arbitrary elliptic operator, the general structure of the coefficient matrix of the SLAE remains the same, and the methods and algorithms presented below will be applicable.

Equation (1) will take the following form:

$$\frac{\partial^\alpha U(x_1,x_2,t)}{\partial t^\alpha} = \frac{\partial^2 U(x_1,x_2,t)}{\partial x_1^2} + \frac{\partial^2 U(x_1,x_2,t)}{\partial x_2^2} + \psi(x_1,x_2)\eta(x_1,x_2,t), \tag{6}$$

where $U(x_1,x_2,t)$ and $\psi(x_1,x_2)$ are the sought functions; $a(x_1,x_2), b(x_1,x_2), c(x_1,x_2)$, $\eta(x_1,x_2,t)$ are the given functions; and $0 < \alpha < 1$ is the the order of the fractional derivative.

The problem is considered for the area $\Omega : 0 \le x_1 \le \gamma_1, 0 \le x_2 \le \gamma_2$ and time interval $0 \le t \le T$.

To discretize Equation (6), we construct the partitioning on the solution domain. On intervals $[0,\gamma_1], [0,\gamma_2]$, we introduce the grid with n and N points. The steps are $h_1 = \Delta x_1 = \gamma_1/n$ and $h_2 = \Delta x_2 = \gamma_2/N$. For the time interval $[0,T]$, we use the uniform grid of $\widetilde{N}$ points. The step is $\tau = \Delta t = T/\widetilde{N}$. We denote the nodes as $x_{1,i_1} = i_1 h_1,\ x_{2,i_2} = i_2 h_2,\ i_1 \in \{0,1,\ldots,n\},\ i_2 \in \{0,1,\ldots,N\}$, and $t_j = j\tau, j \in \left\{0,1,\ldots,\widetilde{N}\right\}$. The values of discretized functions U and others are denoted as $U_{i_1,i_2,j} = U(x_{1,i_1}, x_{2,i_2}, t_j)$.

In this work, the Grunwald–Letnikov formula [2] is used for approximating the Caputo fractional derivative in Equation (6)

$$\begin{aligned} &D_t^\alpha U_{i_1,i_2,j} \cong \sigma_{\alpha,\tau} \sum_{\ell=1}^{j} w_\ell^{(\alpha)} (U_{i_1,i_2,j-\ell+1} - U_{i_1,i_2,j-\ell}), \\ &\sigma_{\alpha,\tau} = \frac{1}{\Gamma(1-\alpha)(1-\alpha)\tau^\alpha}, \quad w_\ell^{(\alpha)} = \ell^{1-\alpha} - (\ell-1)^{1-\alpha}. \end{aligned} \tag{7}$$

The implicit two-step finite difference scheme is used for approximating Equation (6). For the grid point (x_{i_1}, x_{i_2}) at time layer t_j, the difference equation has the form

$$\begin{aligned} &\sigma_{\alpha,\tau} \sum_{\ell=1}^{j} w_\ell^{(\alpha)} (U_{i_1,i_2,j-\ell+1} - U_{i_1,i_2,j-\ell}) = \\ &= \frac{U_{i_1-1,i_2,j} - 2U_{i_1,i_2,j} + U_{i_1+1,i_2,j}}{h_1^2} + \frac{U_{i_1,i_2-1,j} - 2U_{i_1,i_2,j} + U_{i_1,i_2+1,j}}{h_2^2} + \psi_{i_1,i_2}\eta_{i_1,i_2,j}\,. \end{aligned} \tag{8}$$

2.3. Constructing the SLAE

Let us apply the following transformations:

$$\begin{aligned} &\sigma_{\alpha,\tau}(U_{i_1,i_2,j} - U_{i_1,i_2,j-1}) + \sigma_{\alpha,\tau} \sum_{\ell=2}^{j} w_\ell^{(\alpha)} (U_{i_1,i_2,j-\ell+1} - U_{i_1,i_2,j-\ell}) = \\ &= r_{i_1,i_2} U_{i_1,i_2-1,j} + p_{i_1,i_2} U_{i_1-1,i_2,j} + (-2p_{i_1,i_2} - 2q_{i_1,i_2}) U_{i_1,i_2,j} + \\ &+ p_{i_1,i_2} U_{i_1+1,i_2,j} + r_{i_1,i_2} U_{i_1,i_2+1,j} + \psi_{i_1,i_2}\eta_{i_1,i_2,j}\,; \end{aligned}$$

$$\begin{aligned} &- r_{i_1,i_2} U_{i_1,i_2-1,j} - p_{i_1,i_2} U_{i_1-1,i_2,j} + q_{i_1,i_2} U_{i_1,i_2,j} - p_{i_1,i_2} U_{i_1+1,i_2,j} - r_{i_1,i_2} U_{i_1,i_2+1,j} = \\ &= \sigma_{\alpha,\tau} \left(U_{i_1,i_2,j-\ell} - \sum_{\ell=2}^{j} w_\ell^{(\alpha)} (U_{i_1,i_2,j-\ell+1} - U_{i_1,i_2,j-\ell}) \right) + \psi_{i_1,i_2}\eta_{i_1,i_2,j}\,, \end{aligned}$$

where

$$p_{i_1,i_2} = \frac{1}{h_1^2}, \quad r_{i_1,i_2} = \frac{1}{h_2^2}, \quad q_{i_1,i_2} = \sigma_{\alpha,\tau} + 2p_{i_1,i_2} + 2q_{i_1,i_2}.$$

Let us denote

$$\begin{aligned} &f_{i_1,i_2,j} = \sigma_{\alpha,\tau} \left(U_{i_1,i_2,j-\ell} - \sum_{\ell=2}^{j} w_\ell^{(\alpha)} (U_{i_1,i_2,j-\ell+1} - U_{i_1,i_2,j-\ell}) \right) + \psi_{i_1,i_2}\eta_{i_1,i_2,j}\,, \quad j > 1, \\ &f_{i_1,i_2,1} = \sigma_{\alpha,\tau} U_{i_1,i_2,0} + \psi_{i_1,i_2}\eta_{i_1,i_2,0}\,. \end{aligned} \tag{9}$$

The difference equation will take the form

$$-r_{i_1,i_2}U_{i_1,i_2-1,j} - p_{i_1,i_2}U_{i_1-1,i_2,j} + q_{i_1,i_2}U_{i_1,i_2,j} - p_{i_1,i_2}U_{i_1+1,i_2,j} - r_{i_1,i_2}U_{i_1,i_2+1,j} = f_{i_1,i_2,j}\,. \tag{10}$$

Note that the values $U_{0,i_2,j}$, $U_{n,i_2,j}$, $i_2 \in \{0,1,\ldots,N\}$ and $U_{i_1,0,j}$, $U_{i_1,N,j}$, $i_1 \in \{0,1,\ldots,n\}$ at the boundaries are given. Then, for all inner points (x_{i_1}, x_{i_2}), $i_1 \in \{1,2,\ldots,n-1\}$, $i_2 \in \{1,2,\ldots,N-1\}$, difference Equation (10) constitutes an SLAE

$$AY = F\,, \tag{11}$$

where A is the block-tridiagonal matrix of dimension $(n-2)(N-2) \times (n-2)(N-2)$

$$A = \begin{bmatrix} C_1 & -B_1 & & & & & \\ -A_2 & C_2 & -B_2 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & -A_{i_2} & C_{i_2} & -B_{i_2} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -A_{N-2} & C_{N-2} & -B_{N-2} \\ & & & & & -A_{N-1} & C_{N-1} \end{bmatrix},$$

$i_2 \in \{1,2,\ldots,N-1\}$.

Blocks A_{i_2}, B_{i_2}, C_{i_2} of dimension $(n-2) \times (n-2)$ are defined as

$$A_{i_2} = B_{i_2} = \begin{bmatrix} r_{1,i_2} & & & & & \\ & r_{2,i_2} & & & & \\ & & \ddots & & & \\ & & & r_{i_1,i_2} & & \\ & & & & \ddots & \\ & & & & & r_{n-2,i_2} \\ & & & & & & r_{n-1,i_2} \end{bmatrix},$$

$i_1 \in \{1,2,\ldots,n-1\}$,

$$C_{i_2} = \begin{bmatrix} q_{1,i_2} & -p_{1,i_2} & & & & & & \\ -p_{2,i_2} & q_{2,i_2} & -p_{2,i_2} & & & & & \\ & \ddots & \ddots & \ddots & & & & \\ & & -p_{i_1,i_2} & q_{i_1,i_2} & -p_{i_1,i_2} & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & -p_{n-2,i_2} & q_{n-2,i_2} & -p_{n-2,i_2} & \\ & & & & & -p_{n-1,i_2} & q_{n-1,i_2} \end{bmatrix},$$

$i_1 \in \{1, 2, \ldots, n-1\}$.

The sought vector consists of values $U_{i_1,i_2,j}$ collapsed in a row-major order

$$Y = \left[Y_1, Y_2, \ldots, Y_{N-1}\right]^\intercal,$$

$$Y_1 = \left[U_{1,1,j},\ U_{2,1,j}, \ldots,\ U_{n-2,1,j},\ U_{n-1,1,j}\right];$$

$$Y_2 = \left[U_{1,2,j},\ U_{2,2,j}, \ldots,\ U_{n-2,2,j},\ U_{n-1,2,j}\right];$$

$$\ldots$$

$$Y_{N-1} = \left[U_{1,N-1,j},\ U_{2,N-1,j}, \ldots,\ U_{n-1,N-1,j}\right].$$

Right-hand vector is constructed similarly, taking into account the boundary points

$$F = \left[F_1, F_2, \ldots, F_{N-1}\right]^\intercal;$$

$$F_1 = \Big[F_{1,1,j} + r_{1,1}U_{1,0,j} + p_{1,1}U_{0,1,j},\ F_{2,1,j} + r_{2,1}U_{2,0,j},\ \ldots,$$
$$F_{n-2,1,j} + r_{n-2,1}U_{n-2,0,j},\ F_{n-1,1,j} + r_{n-1,0}U_{n-2,0,j} + p_{n-1,1}U_{n,1,j}\Big];$$

$$F_2 = \left[F_{1,2,j} + p_{1,2}U_{0,2,j},\ F_{2,2,j},\ \ldots,\ F_{n-1,2,j},\ F_{n-1,2,j} + p_{n-1,2}U_{n,2,j}\right];$$

$$\ldots$$

$$F_{N-1} = \Big[F_{1,N-1,j} + r_{1,N-1}U_{1,N,j} + p_{1,N-1}U_{0,N-1,j},\ F_{2,N-1,j} + r_{2,N-1}U_{2,N,j},\ \ldots,$$
$$F_{n-1,N-1,j} + r_{n-1,N-1}U_{n-1,N,j} + p_{n-1,N-1}U_{n,N-1,j}\Big].$$

To numerically solve the initial boundary problem, we need to solve system (11) at each subsequent time level $j = 1, \ldots, \widetilde{N}$.

3. Numerical Methods for Solving the SLAE

To solve the SLAEs with various matrix structures, different numerical methods may be used. In this work, for solving the block-tridiagonal SLAE (11) we will use the block-elimination method [31] and parallel matrix sweep method [32].

3.1. Block-Elimination Method

Let us consider an SLAE with a block tridiagonal matrix in the following form:

$$\left\{\begin{array}{rcll} C_0\overline{Y}_0 - B_0\overline{Y}_1 & = & \overline{F}_0\,, & i=0,\\ -A_i\overline{Y}_{i-1} + C_i\overline{Y}_i - B_i\overline{Y}_{i+1} & = & \overline{F}_i\,, & i=1,2,\ldots,N-1,\\ -A_N\overline{Y}_{N-1} + C_N\overline{Y}_N & = & \overline{F}_N\,, & i=N, \end{array}\right. \tag{12}$$

where $\overline{Y}_i$ are the sought vectors of the n dimension; $\overline{F}_i$ are the given right-hand vectors of the n dimension; and A_i, B_i, C_i are the square matrices of the $n \times n$ dimension.

The direct block elimination (or matrix sweep) method is intended for solving the SLAE with block tridiagonal matrix. The auxilliary coefficients α_i (matrices of $n \times n$ dimension) and β_i (vectors of n dimension) are found by the reccurent formulae (the forward elimination phase)

$$\begin{array}{l} \alpha_0 = C_0^{-1}B_0, \quad \alpha_i = (C_i - A_i\alpha_i)^{-1}B_i, \quad i=1,2,\ldots,N-1,\\ \beta_0 = C_0^{-1}F_0, \quad \beta_i = (C_i - A_i\alpha_i)^{-1}(\overline{F_i} + A_i\beta_{i-1}), \quad i=1,2,\ldots,N. \end{array} \tag{13}$$

Solution vectors $\overline{Y_i}$ are found by formulae (the backward substitution phase)

$$\overline{Y_N} = \beta_N, \quad \overline{Y_i} = \alpha_i\overline{Y_{i+1}} + \beta_i, \quad i = N-1, N-2, \ldots, 1, 0. \tag{14}$$

Remark 1. *The algorithms* (13) *and* (14) *are correct if matrices* C_0 *and* $(C_i - A_i\alpha_i)$ *are nonsingular for* $i = 1,2,\ldots,N$. *The algorithm is* stable *if* $\|\alpha_i\| \le 1$ *for* $i = 1,2,\ldots,N$.

The stability condition for this algorithm is the following.

Lemma 1. *If matrices* C_i, $i = 0,1,\ldots,N$ *are nonsingular, matrices* A_i, B_i, $i = 1,\ldots,N-1$ *are non-null, and conditions*

$$\|C_0^{-1}B_0\| \le 1, \quad \|C_N^{-1}A_N\| \le 1, \quad \|C_i^{-1}A_i\| + \|C_i^{-1}B_i\| \le 1, \quad i = 1,\ldots,N-1, \tag{15}$$

are satisfied where at least one of the inequalities is strict; then, the algorithms (13) *and* (14) *are stable and correct.*

Remark 2. *The coefficients* α_i, β_i *are dependent on the previous ones* $\alpha_{i-1}, \beta_{i-1}$. *Thus, we cannot distribute these calculations to independent workers. Thus, the parallelization is limited to operations of matrix inversion and multiplication.*

3.2. Parallel Matrix Sweep Method

To construct a direct parallel algorithm, let us split the interval $i = 0,1,\ldots,N$ into L subintervals of length M such as $N = L \times M$. Consider the unknown values $\overline{Y}_K$, $K = 0, M, \ldots, N$ as parameters.

Now, let us construct the reduced SLAE for $\overline{Y}_K$. To do this, consider the following problems for the interval $(K, K+M)$

$$\left\{\begin{array}{lll} -A_i\overline{U}^1_{i-1} + C_i\overline{U}^1_i - B_i\overline{U}^1_{i+1} = 0\,, & \overline{U}^1_K = (10\ldots0)\,, & \overline{U}^1_{K+M} = (00\ldots0),\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\\ -A_i\overline{U}^n_{i-1} + C_i\overline{U}^n_i - B_i\overline{U}^n_{i+1} = 0\,, & \overline{U}^n_K = (00\ldots1)\,, & \overline{U}^n_{K+M} = (00\ldots0), \end{array}\right. \tag{16}$$

$$\begin{cases} -A_i\overline{V}_{i-1}^1 + C_i\overline{V}_i^1 - B_i\overline{V}_{i+1}^1 = 0\,, & \overline{V}_K^1 = (00\ldots0)\,, & \overline{V}_{K+M}^1 = (10\ldots0), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\ -A_i\overline{V}_{i-1}^n + C_i\overline{V}_i^n - B_i\overline{V}_{i+1}^n = 0\,, & \overline{V}_K^n = (00\ldots0)\,, & \overline{V}_{K+M}^n = (00\ldots1), \end{cases} \tag{17}$$

$$\{\ -A_i\overline{W}_{i-1} + C_i\overline{W}_i - B_i\overline{W}_{i+1} = \overline{F}_i\,, \quad \overline{W}_K = (00\ldots0)\,, \quad \overline{W}_{K+M} = (00\ldots0), \tag{18}$$

where $i = K+1, \ldots, K+M-1$.

Theorem 1. *If $\overline{U}_i^1, \ldots, \overline{U}_i^n$ are solutions of auxilliary problem (16); $\overline{V}_i^1, \ldots, \overline{V}_i^n$ are solutions of problem (17); $\overline{W}_i$ are solutions of problem (18); and $\overline{Y}_i$ are solutions of the basic problem (12) for interval $(K, K+M)$, then*

$$\overline{Y}_i = (\overline{U}_i^1\overline{U}_i^2\ldots\overline{U}_i^n)\overline{Y}_K + (\overline{V}_i^1\overline{V}_i^2\ldots\overline{V}_i^n)\overline{Y}_{K+M} + \overline{W}_i. \tag{19}$$

Proof. Consider system (12) for the inner subinterval $(K, K+M)$:

$$-A_i\overline{Y}_{i-1} + C_i\overline{Y}_i - B_i\overline{Y}_{i+1} = \overline{F}_i,\ \ i = K+1, \ldots, K+M-1. \tag{20}$$

This system contains the parameters $\overline{Y}_K$ and $\overline{Y}_{K+M}$.
Let us rewrite (20) in the form

$$\begin{cases} C_{K+1}\overline{Y}_{K+1} - B_{K+1}\overline{Y}_{K+2} = A_{K+1}\overline{Y}_K + \overline{F}_{K+1}, \\ -A_i\overline{Y}_{i-1} + C_i\overline{Y}_i - B_i\overline{Y}_{i+1} = \overline{F}_i\,, i = K+2, \ldots, K+M-2, \\ -A_{K+M-1}\overline{Y}_{K+M-2} + C_{K+M-1}\overline{Y}_{K+M-1} = B_{K+M-1}\overline{Y}_{K+M} + \overline{F}_{K+M-1}. \end{cases} \tag{21}$$

In this system (21), $A_{K+1}\overline{Y}_K$ and $B_{K+M-1}\overline{Y}_{K+M}$ have the following form:

$$\begin{aligned} A_{K+1}\overline{Y}_K &= \begin{pmatrix} A_{K+1}^{11} & \ldots & A_{K+1}^{1N} \\ \ldots & \ldots & \ldots \\ A_{K+1}^{N1} & \ldots & A_{K+1}^{NN} \end{pmatrix} \begin{pmatrix} Y_K^1 \\ \ldots \\ Y_K^N \end{pmatrix} = \\ &= \begin{pmatrix} A_{K+1}^{11} \\ \ldots \\ A_{K+1}^{N1} \end{pmatrix} Y_K^1 + \cdots + \begin{pmatrix} A_{K+1}^{1N} \\ \ldots \\ A_{K+1}^{NN} \end{pmatrix} Y_K^N = \\ &= \overline{A}_{K+1}^1 Y_K^1 + \overline{A}_{K+1}^2 Y_K^2 + \cdots + \overline{A}_{K+1}^N Y_K^N; \end{aligned} \tag{22}$$

$$\begin{aligned} B_{K+M-1}\overline{Y}_{K+M} &= \begin{pmatrix} B_{K+M-1}^{11} & \ldots & B_{K+M-1}^{1N} \\ \ldots & \ldots & \ldots \\ B_{K+M-1}^{N1} & \ldots & B_{K+M-1}^{NN} \end{pmatrix} \begin{pmatrix} Y_{K+M}^1 \\ \ldots \\ Y_{K+M}^N \end{pmatrix} = \\ &= \begin{pmatrix} B_{K+M-1}^{11} \\ \ldots \\ B_{K+M-1}^{N1} \end{pmatrix} Y_{K+M}^1 + \cdots + \begin{pmatrix} B_{K+M-1}^{1N} \\ \ldots \\ B_{K+M-1}^{NN} \end{pmatrix} Y_{K+M}^N = \\ &= \overline{B}_{K+M-1}^1 Y_{K+M}^1 + \overline{B}_{K+M-1}^2 Y_{K+M}^2 + \cdots + \overline{B}_{K+M-1}^N Y_{K+M}^N. \end{aligned}$$

Here, the line over the symbol denotes the vector column of corresponding matrix.
Taking into account the formulae (22), SLAEs (21) have the following form:

$$\begin{pmatrix} C_{K+1} & -B_{K+1} & & & \\ -A_{K+2} & C_{K+2} & -B_{K+2} & & \\ & \ddots & \ddots & \ddots & \\ & & -A_{K+M-1} & C_{K+M-1} & \end{pmatrix} \begin{pmatrix} \overline{Y}_{K+1} \\ \overline{Y}_{K+2} \\ \cdots \\ \overline{Y}_{K+M-1} \end{pmatrix} =$$

$$= \begin{pmatrix} \overline{F}_{K+1} \\ \overline{F}_{K+2} \\ \cdots \\ \overline{F}_{K+M-1} \end{pmatrix} + \begin{pmatrix} \overline{A}^1_{K+1} \\ 0 \\ \cdots \\ 0 \end{pmatrix} Y^1_K + \cdots + \begin{pmatrix} \overline{A}^N_{K+1} \\ 0 \\ \cdots \\ 0 \end{pmatrix} Y^N_K + \tag{23}$$

$$+ \begin{pmatrix} 0 \\ \cdots \\ 0 \\ \overline{B}^1_{K+M-1} \end{pmatrix} Y^1_{K+M} + \cdots + \begin{pmatrix} 0 \\ \cdots \\ 0 \\ \overline{B}^N_{K+M-1} \end{pmatrix} Y^N_{K+M}.$$

System (23) is equivalent to the following one:

$$\Lambda \overline{Y} = \overline{F} + \overline{A}^1 Y^1_K + \cdots + \overline{A}^N Y^N_K + \overline{B}^1 Y^1_{K+M} + \cdots + \overline{B}^N Y^N_{K+M}, \tag{24}$$

where Λ is the submatrix of the basic block tridiagonal matrix of system (12) that corresponds to the interval $(K, K+M)$.

If matrix Λ is inversible, then

$$\overline{Y} = \Lambda^{-1}\overline{F} + \Lambda^{-1}\overline{A}^1 Y^1_K + \cdots + \Lambda^{-1}\overline{A}^N Y^N_K + \Lambda^{-1}\overline{B}^1 Y^1_{K+M} + \cdots + \Lambda^{-1}\overline{B}^N Y^N_{K+M}.$$

From this, it follows that

$$\overline{Y} = \overline{U}^1 Y^1_K + \cdots + \overline{U}^N Y^N_K + \overline{V}^1 Y^1_{K+M} + \cdots + \overline{V}^N Y^N_{K+M} + \overline{W}, \tag{25}$$

where

$$\begin{gathered} \overline{U}^1 \text{ is a solution of } \Lambda\overline{U}^1 = \overline{A}^1, \ldots, \\ \overline{U}^N \text{ is a solution of } \Lambda\overline{U}^N = \overline{A}^N, \\ \\ \overline{V}^1 \text{ is a solution of } \Lambda\overline{V}^1 = \overline{B}^1, \ldots, \\ \overline{V}^N \text{ is a solution of } \Lambda\overline{V}^N = \overline{B}^N, \\ \overline{W} \text{ is a solution of } \Lambda\overline{W} = \overline{F}. \end{gathered} \tag{26}$$

Let us rewrite $\overline{U}^1, \ldots, \overline{U}^N$, $\overline{V}^1, \ldots, \overline{V}^N$, $\overline{W}$ in a more detailed manner.

$$\overline{U}^1 = \begin{pmatrix} \overline{U}^1_{K+1} \\ \cdots \\ \overline{U}^1_{K+M-1} \end{pmatrix}, \ldots, \overline{U}^N = \begin{pmatrix} \overline{U}^N_{K+1} \\ \cdots \\ \overline{U}^N_{K+M-1} \end{pmatrix}, \overline{V}^1 = \begin{pmatrix} \overline{V}^1_{K+1} \\ \cdots \\ \overline{V}^1_{K+M-1} \end{pmatrix}, \ldots,$$

$$\overline{V}^N = \begin{pmatrix} \overline{V}^N_{K+1} \\ \cdots \\ \overline{V}^N_{K+M-1} \end{pmatrix}, \overline{W} = \begin{pmatrix} \overline{W}_{K+1} \\ \cdots \\ \overline{W}_{K+M-1} \end{pmatrix}. \tag{27}$$

Taking into account (23) and (27), problems (26) are equivalent to following:

$$\begin{cases} C_{K+1}\overline{U}^1_{K+1} - B_{K+1}\overline{U}^1_{K+2} = \overline{A}^1_{K+1}, \\ \quad \cdots \\ -A_{K+M-1}\overline{U}^1_{K+M-2} + C_{K+M-1}\overline{U}^1_{K+M-1} = 0. \end{cases}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\begin{cases} C_{K+1}\overline{U}^N_{K+1} - B_{K+1}\overline{U}^N_{K+2} = \overline{A}^N_{K+1}; \\ \quad \cdots \\ -A_{K+M-1}\overline{U}^N_{K+M-2} + C_{K+M-1}\overline{U}^N_{K+M-1} = 0; \end{cases}$$

$$\begin{cases} C_{K+1}\overline{V}^1_{K+1} - B_{K+1}\overline{V}^1_{K+2} = 0; \\ \quad \dots \\ -A_{K+M-1}\overline{V}^1_{K+M-2} + C_{K+M-1}\overline{V}^1_{K+M-1} = \overline{B}^1_{K+M-1}; \end{cases} \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \begin{cases} C_{K+1}\overline{V}^N_{K+1} - B_{K+1}\overline{V}^N_{K+2} = 0; \\ \quad \dots \\ -A_{K+M-1}\overline{V}^N_{K+M-2} + C_{K+M-1}\overline{V}^N_{K+M-1} = \overline{B}^1_{K+M-1}; \end{cases}$$

$$\begin{cases} C_{K+1}\overline{W}_{K+1} - B_{K+1}\overline{W}_{K+2} = \overline{F}_{K+1}; \\ \quad \dots \\ -A_{K+M-1}\overline{W}_{K+M-2} + C_{K+M-1}\overline{W}_{K+M-1} = \overline{F}_{K+M-1}. \end{cases} \tag{28}$$

Systems (28) for intervals $(K, K+M)$, $K = 0, M, \dots, N$ are equivalent to problems (16)–(18).

Thus, the original solution (25) on interval $(K, K+M)$ has the form

$$\overline{Y}_i = (\overline{U}^1_i \overline{U}^2_i \dots \overline{U}^n_i)\overline{Y}_K + (\overline{V}^1_i \overline{V}^2_i \dots \overline{V}^n_i)\overline{Y}_{K+M} + \overline{W}_i, \;\; i = K+1, \dots, K+M-1.$$

□

Remark 3. *Matrices U_i, V_i and vector $\overline{W}_i$ can be obtained by the block elimination methods (13) and (14). For arbitrary L block elimination formulae for intervals $(K, K+M)$, $K = 0, M, \dots, N$ have the following form.*

The forward phase for Equations (16)–(18) is as follows:

$$\begin{aligned} &\alpha_{K+1} = C^{-1}_{K+1}B_{K+1}, \;\; \beta_{K+1} = C^{-1}_{K+1}A_{K+1}, \;\; \overline{\gamma}_{K+1} = C^{-1}_{K+1}\overline{F}_{K+1}, \\ &\alpha_i = [C_i - A_i\alpha_{i-1}]^{-1}B_i, \;\; \beta_i = [C_i - A_i\alpha_{i-1}]^{-1}A_i\beta_{i-1}, \\ &\overline{\gamma}_i = [C_i - A_i\alpha_{i-1}]^{-1}(\overline{F}_i + A_i\overline{\gamma}_{i-1}), \;\; i = K+2, \dots, K+M-1. \end{aligned} \tag{29}$$

The backward phase for Equations (16)–(18) is as follows:

$$\begin{aligned} &U_{K+M-1} = \beta_{K+M-1}, \;\; V_{K+M-1} = \alpha_{K+M-1}, \;\; \overline{W}_{K+M-1} = \overline{\gamma}_{K+M-1}, \\ &U_i = \alpha_i U_{i+1} + \beta_i, \;\; V_i = \alpha_i V_{i+1}, \;\; \overline{W}_i = \alpha_i \overline{W}_{i+1} + \overline{\gamma}_i, \quad i = K+M-2, \dots, K+1. \end{aligned} \tag{30}$$

When we substitute expression (19) for indices $K = 0, M, \dots, N$ into the basic system (12), we will obtain a reduced SLAE for the parametric unknown values (vectors $\overline{Y}_K$). This system has a similar structure to (12) but has a smaller size.

$$\begin{cases} [C_0 - B_0U_1]\overline{Y}_0 - [B_0V_1]\overline{Y}_M = \overline{F}_0 + B_0\overline{W}_1, \quad K = 0, \\ -[A_KU_{K-1}]\overline{Y}_{K-M} + [C_K - A_KV_{K-1} - B_KU_{K+1}]\overline{Y}_K - [B_KV_{K+1}]\overline{Y}_{K+M} = \\ = \overline{F}_K + A_K\overline{W}_{K-1} + B_K\overline{W}_{K+1}, \quad K = M, \dots, N-M, \\ -[A_NU_{N-1}]\overline{Y}_{N-M} + [C_N - A_NV_{N-1}]\overline{Y}_N = \overline{F}_N + A_N\overline{W}_{N-1}, \quad K = N, \end{cases} \tag{31}$$

where U_K and V_K are matrices of dimension $n \times n$.

Problem (31) is solved by the block-elimination method (13) and (14) in the single-threaded mode or on a single node of a cluster. Auxilliary problems (16)–(18) are solved independently for each of the L intervals. Thus, this workload can be distributed between L threads or processes. After obtaining $\overline{Y}_K$, the rest of the unknown values are found by formula (19). This work also can be performed independently for each of the L intervals.

The parallel matrix sweep algorithm for solving system (12) is presented in Listing 1.

Listing 1. Parallel matrix sweep algorithm for solving SLAE.

1. Find values $U_i, V_i, \overline{W}_i$ from problems (16)–(18) for inner points $i \in (K, K+M)$ of each subinterval $K = 0, M, 2M, \ldots, N$. Methods (29) and (30) are used to solve the subproblems independently on each of the subintervals.
2. Calculate the coefficients for reduced system (31). The coefficients may be calculated independently for each K, but to solve the resulting reduced system, we need to transfer these coefficients to a single process. This requires synchronization or gather-type communication.
3. Find values $\overline{Y}_K$ from the reduced system (31). Compared to the basic system (12), its dimension is much smaller. It is solved by the block elimination algorithms (13) and (14) in serial mode. The computed values $\overline{Y}_K$ must be transmitted to processors. This step requires synchronization or communication.
4. Use formula (19) to calculate the sought values $\overline{Y}_i$, $i \in (K, K+M)$. These computations may be performed independently for each subinterval K.

Thus, steps 1, 2, and 4 of this algorithm may be parallelized, while step 3 must be performed in serial mode.

To establish the stability (see Remark 1) of the parallel matrix sweep algorithm, let us prove the following theorems.

Theorem 2. *If original system (12) satisfies the condition*

$$\|C_i\| \geq \|A_i\| + \|B_i\| + \delta, \quad \delta > 0,$$

then reduced system (31) also satisfies this condition in the form

$$\|C_K - A_K V_{K-1} - B_K U_{K+1}\| \geq \|A_K U_{K-1}\| + \|B_K V_{K+1}\| + \delta.$$

Proof.

$$\begin{aligned}
&\|C_K - A_K V_{K-1} - B_K U_{K+1}\| \geq \|C_K\| - \|A_K V_{K-1}\| - \|B_K U_{K+1}\| \geq \\
\geq &\|C_K\| - \|A_K(I - U_{K-1})\| - \|B_K(I - V_{K+1})\| \geq \\
\geq &\|C_K\| - \|A_K\| + \|A_K U_{K-1}\| - \|B_K\| + \|B_K V_{K+1}\| \geq \\
\geq &\|A_K U_{K-1}\| + \|B_K V_{K+1}\| + \|C_K\| - \|A_K\| - \|B_K\| \geq \\
\geq &\|A_K U_{K-1}\| + \|B_K V_{K+1}\| + \delta,
\end{aligned}$$

since $\|U_K\| + \|V_K\| \leq 1$. □

Theorem 3. *If basic system (12) satisfies the stability conditions of the matrix sweep method (Lemma 1), then these conditions are sufficient for the stability of the matrix sweep method for reduced system (31) for $\overline{Y}_K$.*

Proof. Let us construct a proof by the mathematical induction method.

We will utilize the following statement [31]. If square matrix S satisfies $\|S\| \leq q \leq 1$, then matrices $(E - S)^{-1}$ and $\|(E - S)^{-1}\| \leq 1/(1 - q)$ must exist.

Let $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_K, \ldots, \tilde{\alpha}_N$ be the elimination coefficients for the matrix sweep method for system (31).

1. Let us demonstrate that $\|\tilde{\alpha}_1\| \leq 1$.

$$\|C_0^{-1} B_0 U_1\| \leq \|C_0^{-1} B_0\| \cdot \|U_1\| < 1,$$

therefore, there are $(E - C_0^{-1} B_0 U_1)^{-1}$ and

$$\|\tilde{\alpha}_1\| = \|(C_0 - B_0 U_1)^{-1} B_0 V_1\| \leq \|C_0^{-1}(E - C_0^{-1} B_0 U_1)^{-1} B_0 V_1\| \leq$$

$$\leq \|(E - C_0^{-1}B_0U_1)^{-1}\| \cdot \|C_0^{-1}B_0\| \cdot \|V_1\| \leq \frac{1}{1 - \|C_0^{-1}B_0\| \cdot \|U_1\|} \cdot \|C_0^{-1}B_0\| \cdot \|V_1\| \leq$$

$$\leq \frac{\|V_1\|}{1 - \|U_1\|} \leq \frac{\|V_1\|}{\|V_1\|} = 1, \quad \text{since} \quad \|U_1\| + \|V_1\| \leq 1.$$

2. Assume $\|\tilde{\alpha}_K\| \leq 1$. Let us demonstrate that $\|\tilde{\alpha}_{K+1}\| \leq 1$.

$$\|\tilde{\alpha}_{K+1}\| = \|(C_K - A_KV_{K-1} - B_KU_{K+1} - A_KU_{K-1}\tilde{\alpha}_K)^{-1} \cdot B_KV_{K+1}\|.$$

Consider,

$$\|(C_K^{-1}A_KV_{K-1} + C_K^{-1}B_KU_{K+1} + C_K^{-1}A_KU_{K-1}\tilde{\alpha}_K)^{-1}\| \leq$$

$$\leq \|C_K^{-1}A_K\| \cdot \|V_{K-1}\| + \|C_K^{-1}B_K\| \cdot \|U_{K+1}\| + \|C_K^{-1}A_K\| \cdot \|U_{K-1}\| \leq$$

$$\leq \|C_K^{-1}A_K\| + \|C_K^{-1}B_K\| \cdot \|U_{K+1}\| \leq 1 - \|C_K^{-1}B_K\| + \|C_K^{-1}B_K\| \cdot \|U_{K+1}\| \leq$$

$$\leq 1 - \|C_K^{-1}B_K\| \cdot (1 - \|U_{K+1}\|) \leq 1 - \|C_K^{-1}B_K\| \cdot \|V_{K+1}\| < 1,$$

since

$$\|U_{K+1}\| + \|V_{K+1}\| \leq 1.$$

Therefore, there are

$$(E - C_K^{-1}A_KV_{K-1} + C_K^{-1}B_KU_{K+1} + C_K^{-1}A_KU_{K-1}\tilde{\alpha}_K)^{-1} \quad \text{and}$$

$$\|\tilde{\alpha}_{K+1}\| \leq \|(E - C_K^{-1}A_KV_{K-1} + C_K^{-1}B_KU_{K+1} + C_K^{-1}A_KU_{K-1}\tilde{\alpha}_K)^{-1}\| \times$$

$$\times \|C_K^{-1}B_KV_{K+1}\| \leq \frac{1}{\|C_K^{-1}B_K\| \cdot \|V_{K+1}\|} \cdot \|C_K^{-1}B_K\| \cdot \|V_{K+1}\| = 1.$$

3. Let us demonstrate that $\|\tilde{\alpha}_N\| \leq 1$.

$$\|C_N^{-1}A_NV_{N-1}\| \leq \|C_N^{-1}A_N\| \cdot \|V_{N-1}\| < 1,$$

therefore, there are

$$(E - C_N^{-1}A_NV_{N-1})^{-1}.$$

Consider

$$\|\tilde{\alpha}_N\| = \|(C - A_NV_{N-1})^{-1}A_NU_{N-1}\| \leq$$

$$\leq \|(E - C_N^{-1}A_NV_{N-1})^{-1}\| \cdot \|C_N^{-1}A_N\| \cdot \|U_{N-1}\| \leq$$

$$\leq \frac{1}{\|C_N^{-1}A_N\| \cdot \|V_{N-1}\|} \cdot \|C_N^{-1}A_N\| \cdot \|U_{N-1}\| \leq \frac{\|U_{N-1}\|}{1 - \|V_{N-1}\|} \leq 1,$$

since $\|U_{N-1}\| + \|V_{N-1}\| \leq 1$. □

4. Numerical Method for Solving the Inverse Problems

To solve the inverse problems (1)–(3) and (5), we use the iterative conjugate gradient method [23,33]. Consider that additional information φ may contain a random perturbation $\varphi^\delta = \varphi \cdot (1 + \text{rand}(-\delta, \delta))$. To overcome this, we will regularize the inverse problems using the Lavrentyev scheme [34].

The resulting algorithm for solving the inverse problems is presented in Listing 2, where $\varepsilon > 0$ is the regularization parameter.

Listing 2. Regularized conjugate gradient algorithm for solving the inverse problems.

Initialization:

1. Set $s = 0$ as the iterative step.
2. Set the initial approximation $\psi_0(\mathbf{x})$, for example, $\psi_0(\mathbf{x}) = 0$.
3. Solve the initial boundary problems (1)–(3) by substituting the right-hand part ψ with ψ_0; obtain $U_T^0 = U(\mathbf{x}, T)\Big|_{\psi_0}$.
4. Calculate the initial residual $r_0(\mathbf{x}) = \varphi(\mathbf{x}) - (U_T^0 + \varepsilon\psi_0)$ and initial estimation $p_0(\mathbf{x}) = r_0(\mathbf{x})$, where $\varepsilon > 0$ is the regularization parameter.

Iterations:

5. Set $s = s + 1$.
6. Solve the initial boundary problems by substituting the right-hand part with $p_s(\mathbf{x})$; obtain $U_T^s = U(\mathbf{x}, T)\Big|_{p_s}$.
7. Calculate the coefficient $\alpha_s = (r_s, r_s)/(p_s, (U_T^s + \varepsilon p_s))$.
8. Calculate the estimation and residual for next step $\psi_{s+1} = \psi_s + \alpha_s p_s$, $r_{s+1} = r_s - \alpha_s(U_T^s + \varepsilon p_s)$.
9. Calculate the coefficient $\beta_s = (r_{s+1}, r_{s+1})/(r_s, r_s)$.
10. Calculate $p_{s+1} = r_{s+1} + \beta_s p_s$.
11. Check the stopping rule $\|r_s\|/\|\varphi\| < \mu$, $0 < \mu < 1$. If not met, go to step 5.

5. Parallel Implementation of Algorithms for Solving the Inverse Problem

The numerical solution to the problems related to the fractional differential equation is an expensive task that requires a lot of computing time.

The most time-consuming subroutine of the regularized conjugate gradient algorithm (see Listing 2) is solving the auxiliary initial boundary problem at each iteration. In turn, this procedure consists in forming and solving SLAEs (11) at each subsequent time step.

5.1. Efficient Computation of the Right-Hand Parts

Forming SLAE requires the calculation of the right-hand parts using formula (9). In our earlier work [27], when solving one-dimensional problems, the fraction of time spent to calculate the right-hand part was up to 70% of the total time. To optimize this procedure, we implemented the logarithmic memory approach. It consists of using the non-uniform time grid when computing the approximation of the fractional derivative. The fine time step is used for the latest history part. For the more distant history, successively larger time steps are used. This approach allowed us to reduce the computing time for the one-dimensional case by up to 1.5 times.

This approach may also be utilized for the ctwo-dimensional problem. For solving system (11), a modified variant of formula (9) takes the form

$$
\begin{aligned}
&f_{i_1,i_2,1} = \sigma_{\alpha,\tau} U_{i_1,i_2,0} + \psi_{i_1,i_2}\eta_{i_1,i_2,0}\,,\\
&f_{i_1,i_2,j} = \sigma_{\alpha,\tau} U_{i_1,i_2,j-1} - \sum_{(\ell,k)} \sigma_{\alpha,\tau}^{(\ell,k)} w_{\ell,k}^{(\alpha)} (U_{i_1,i_2,j-\ell+1} - U_{i_1,i_2,j-k}) + \psi_{i_1,i_2}\eta_{i_1,i_2,j}\,, \quad j > 1,\\
&\qquad (\ell,k) \in \Big\{(2,\, 2+\theta^0), (2+\theta^0,\, 2+\theta^1), (2+\theta^1,\, 2+\theta^2), \ldots, (2+\theta^{\lfloor \log_\theta j \rfloor}, j)\Big\},\\
&\qquad \sigma_{\alpha,\tau}^{(\ell,k)} = \frac{1}{\Gamma(1-\alpha)(1-\alpha)\theta^{k-\ell}\tau^\alpha}, \quad w_{\ell,k}^{(\alpha)} = (k)^{1-\alpha} - (\ell-1)^{1-\alpha}, \quad 0 < \alpha < 1,
\end{aligned}
\tag{32}
$$

where $\theta \in \mathbb{N}$ is the stretching coefficient and $\lfloor \log_\theta n \rfloor$ is the floor function (integer part). Note that approach has complexity $O(\widetilde{N} \cdot \log \widetilde{N})$ in contrast of $O(\widetilde{N}^2)$ of the uniform time grid.

In the next section, we will explore the usefulness of this approach for the case of a two-dimensional problem.

5.2. Parallel Implementation of the SLAE Solver

To speed up SLAE solving, we implement the parallel matrix sweep Algorithm (see Listing 1). For comparison, we also implemented the serial block-elimination methods (13) and (14).

The parallel algorithm for solving the inverse problem of finding the source term of the time-fractional diffusion equation was implemented for the multicore processor using OpenMP technology [35] and the Intel MKL library [36]. The parallelization is performed as follows.

1. The workload of calculating the right-hand parts (9) and (32) is distributed to OpenMP threads utilizing to the same subinterval decomposition that is used for parallel matrix sweep algorithm.
2. The matrix operations and the inversion of the matrix blocks are performed with MKL routines (`gemv`, `gemm`, `getrf`, and `getri`).
3. In the parallel matrix sweep algorithm (Listing 1), steps 1, 2, and 4 are performed by the individual OpenMP threads on their corresponding subintervals. To perform step 3, thread synchronization is required. This is performed by the '`#pragma omp barrier`' directive. Note that this synchronization requires additional time.

6. Numerical Experiments

In this section, we present the numerical experiments of solving the direct and inverse problems for the two-dimensional fractional diffusion equation. The experiments were performed using the developed code on the Intel i9-12900k CPU, which has 8 P-cores. The goal is to study the validity of the proposed numerical methods, as well as the efficiency of the parallel code.

6.1. Problem 1

Consider the two-dimensional equation

$$\frac{\partial^\alpha U(x_1,x_2,t)}{\partial t^\alpha} = \frac{\partial^2 U(x_1,x_2,t)}{\partial x_1^2} + \frac{\partial^2 U(x_1,x_2,t)}{\partial x_2^2} + \left(\frac{2}{\Gamma(3-\alpha)}t^{2-\alpha} + 2t^2\right)\sin(x_1)\sin(x_2) \tag{33}$$

with initial and boundary conditions

$$U(x_1,x_2,0) = 0,$$

$$U(x_1,0,0) = 0, \quad U(x_1,\pi,0) = 0,$$

$$U(0,x_2,0) = 0, \quad U(\pi,x_2,0) = 0,$$

and area

$$0 \le x_1, x_2 \le \gamma_1 = \gamma_2 = \pi,\ 0 \le t \le T = 1,$$

for order

$$0 < \alpha\ < 1.$$

Paper [37] presents the exact solution for this equation

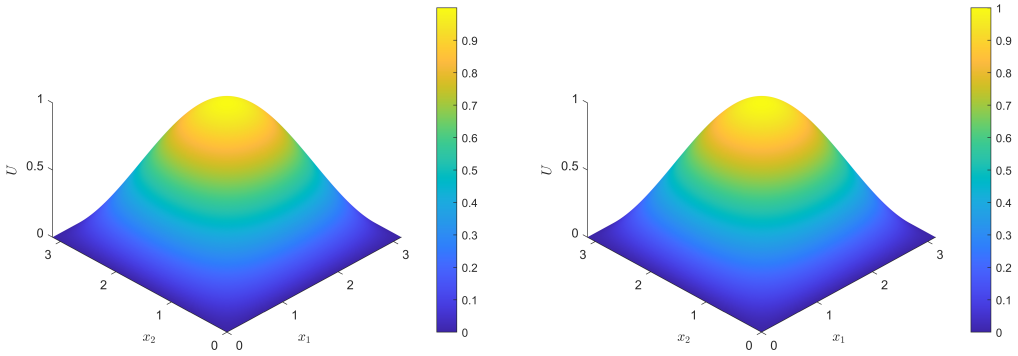
$$U(x_1,x_2,t) = t^2\sin(x_1)\sin(x_2).$$

6.1.1. Experiment 1

Experiment 1 consists of solving the forward (initial boundary) problem for Equation (33) on the various grids $n = N = \{128;\ 256;\ 512\}$, $\widetilde{N} = \{64;\ 128;\ 256\}$ and various parameters $\alpha = \{0.5;\ 0.8;\ 0.95\}$. It was solved using the difference scheme described in Section 2.2. For solving the SLAE, two methods were applied, namely, the classical serial block-elimination method (13) and (14) and parallel matrix sweep method (see Listing 1).

Figure 1 shows the exact solution $U(x_1,x_2,1) = t^2\sin(x_1)\sin(x_2)$ and approximate solution $\widetilde{U}(x_1,x_2,1)$ for Problem 1 obtained by the parallel matrix sweep algorithm for grid

size $n = N = 512$, $\widetilde{N} = 256$, and the order of fractional derivative $\alpha = 0.5$. The approximate solutions obtained by the matrix sweep and parallel matrix sweep methods coincide with each other up to the machine precision (10^{-15}, as we used the double precision format).



(**a**) (**b**)

Figure 1. Results of Experiment 1 for Problem 1: (**a**) exact solution $U(x_1, x_2, T)$; (**b**) approximate solution $\widetilde{U}(x_1, x_2, T)$ obtained by the parallel sweep algorithm.

Table 1 contains the relative error of the solutions $\left\|U - \widetilde{U}\right\| / \|U\|$ for various grid sizes and parameters α. The experiments show that taking a finer grid either for space or time reduces the relative error of the resulting solution. For the time grid, the rate of convergence is close to linear (increasing the number of grid points twofold reduces the error approximately by two times). As parameter α approaches 1, the error of the solution increases. To achieve higher accuracy, we need to use a finer grid.

Table 2 presents the total computing time T_L of solving the direct problem. For the parallel matrix sweep method, the computing time for various numbers L of OpenMP threads are presented. Total time T_L consists of time T_{SLAE} spent on solving the SLAEs plus time T_{Right} spent on computing the right-hand part for these SLAEs using formula (9). The table shows that for the case of two-dimensional problem solving SLAE with a more complex matrix structure (block-tridiagonal), the time spent on solving the SLAE is up to 600 times larger than the time spent on computing the right-hand part. Thus, utilizing the optimized approach for computing the fractional derivative is less relevant as it would bring a miniscule speedup.

Table 1. Results of Experiment 1 for Problem 1: relative error of solving the direct problem.

	Grid Size	$\widetilde{N} = 64$	$\widetilde{N} = 128$	$\widetilde{N} = 256$
$\alpha = 0.5$	$n = N = 128$	3.64×10^{-4}	1.48×10^{-4}	7.07×10^{-5}
	$n = N = 256$	3.45×10^{-4}	1.26×10^{-4}	4.93×10^{-5}
	$n = N = 512$	3.40×10^{-4}	1.21×10^{-4}	1.98×10^{-5}
$\alpha = 0.8$	$n = N = 128$	2.25×10^{-3}	9.44×10^{-4}	4.23×10^{-4}
	$n = N = 256$	2.13×10^{-3}	9.25×10^{-4}	4.04×10^{-4}
	$n = N = 512$	2.12×10^{-3}	9.20×10^{-4}	4.02×10^{-4}
$\alpha = 0.95$	$n = N = 128$	5.14×10^{-3}	2.48×10^{-3}	1.20×10^{-3}
	$n = N = 256$	5.12×10^{-3}	2.46×10^{-3}	1.18×10^{-3}
	$n = N = 512$	5.11×10^{-3}	2.45×10^{-3}	1.18×10^{-3}

Table 2. Results of Experiment 1 for Problem 1: computing time of solving the direct problem.

Method	Number L of OpenMP Threads	T_L (Minutes)	T_{SLAE}	T_{Right}
Matrix Sweep (13) and (14)	Serial	28.6	28.5	0.13
Parallel Matrix Sweep (Listing 1)	2	30.7	30.63	0.06
Parallel Matrix Sweep	4	16.3	16.26	0.03
Parallel Matrix Sweep	8	10.1	10.08	0.016
Parallel Matrix Sweep	16	10.5	10.48	0.017

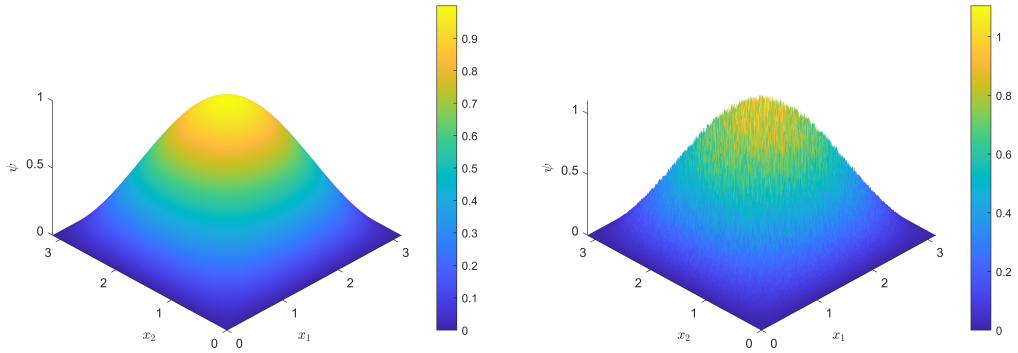
6.1.2. Experiment 2

Experiment 2 consists of solving the inverse problem for Equation (33). We assume that $\eta(t) = \left(\frac{2}{\Gamma(3-\alpha)}t^{2-\alpha} + 2t^2\right)$ and $\varphi(x_1, x_2) = \sin(x_1)\sin(x_2)$. Thus, we need to solve the inverse problem of finding unknown $[U(x_1, x_2, T), \psi(x_1, x_2)]$. For this experiment, we introduce the varying level δ of random perturbation to the *a priori* data $\varphi^\delta = \varphi \cdot (1 + \text{rand}(-\delta, \delta))$.

Remark 4. *Note that this level corresponds to an error in the infinity norm, i.e.,* $\delta \approx \|\varphi - \varphi^\delta\|_\infty / \|\varphi\|_\infty$. *In the rest of the paper, the norm is implied to be* L_2*-norm. In tables, we provide the corresponding* $\delta_2 = \|\varphi - \varphi^\delta\| / \|\varphi\|$.

The inverse problem was solved by the regularized conjugate gradient method Algorithm (see Listing 2). For solving the SLAEs, the parallel matrix sweep method was used. The grid size was $n = N = 256$, $\widetilde{N} = 256$.

Figure 2 shows the exact solution $\psi(x_1, x_2) = \sin(x_1)\sin(x_2)$ and the approximate solution $\widetilde{\psi}(x_1, x_2)$ for Problem 1 for the noise level $\delta = 0.02$.



(**a**) (**b**)

Figure 2. Results of Experiment 2 for Problem 1: (**a**) exact solution $\psi(x_1, x_2)$; (**b**) approximate solution $\widetilde{\psi}(x_1, x_2)$ obtained by the regularized conjugate gradient method with noise level $\delta = 0.02$.

Table 3 presents the results of Experiment 2 for varying levels of noise δ, δ_2. It contains the values of regularization parameter ε, the threshold μ for the stopping criterion (we used $\mu = \delta_2$ for experiments), number of iterations S, and the relative error of the resulting solution.

Table 3. Results of Experiment 2 for Problem 1: solving the inverse problem.

Noise Level δ	Noise Level δ_2	Regularization Parameter ε	Stopping Rule μ	Number of Iterations S	Error of Solution $\|\psi - \widetilde{\psi}\|/\|\psi\|$
0	0	0	0.001	3	5×10^{-5}
0.01	0.006	0.1	0.006	2	0.1
0.02	0.01	0.1	0.01	2	0.14
0.05	0.03	0.2	0.03	1	0.16

6.2. Problem 2

Consider the two-dimensional equation [22]

$$\frac{\partial^\alpha U(x_1,x_2,t)}{\partial t^\alpha} = \frac{\partial^2 U(x_1,x_2,t)}{\partial x_1^2} + \frac{\partial^2 U(x_1,x_2,t)}{\partial x_2^2} + \sin(x_1)\sin(x_2) + \sin(2x_1)\sin(2x_2) \tag{34}$$

with initial and boundary conditions

$$U(x_1,x_2,0) = 0,$$

$$U(x_1,0,0) = 0, \quad U(x_1,\pi,0) = 0,$$

$$U(0,x_2,0) = 0, \quad U(\pi,x_2,0) = 0,$$

and area

$$0 \le x_1, x_2 \le \pi,\ 0 \le t \le 1,$$

for order

$$0 < \alpha\ < 1.$$

The numerical experiment consists of solving the inverse problem for Equation (34). We assume that $\eta(x_1,x_2,t) = 1$. Additional data $\varphi(x_1,x_2) = U(x_1,x_2,1)$ are obtained by solving the direct problem substituting exact $\psi(x_1,x_2) = \sin(x_1)\sin(x_2) + \sin(2x_1)\sin(2x_2)$. They are shown in Figure 3.

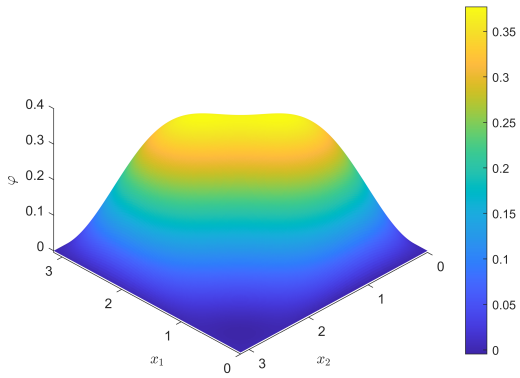


Figure 3. A priori data $\varphi(x_1,x_2)$ for Problem 2.

Table 4 presents the results of experiments for Problem 2 for varying levels of noise δ, δ_2. It contains the values of the regularization parameter ε, the threshold μ for the stopping criterion (we used $\mu = \delta_2$ for experiments), the number of iterations S, and the relative error of the resulting solution. The grid size was $n = N = 256$, $\widetilde{N} = 256$, and order $\alpha = 0.5$.

Table 4. Results of experiments for Problem 2: solving the inverse problem.

Noise Level δ	Noise Level δ_2	Regularization Parameter ε	Stopping Rule μ	Number of Iterations S	Error of Solution $\|\psi - \widetilde{\psi}\|/\|\psi\|$
0	0	0	0.001	3	3.2×10^{-4}
0.01	0.006	0.02	0.006	3	0.13
0.02	0.01	0.02	0.01	3	0.19
0.05	0.03	0.05	0.03	1	0.23

Figure 4 shows the approximate solutions $\widetilde{\psi}(x_1, x_2)$ for Problem 2 for various noise levels.

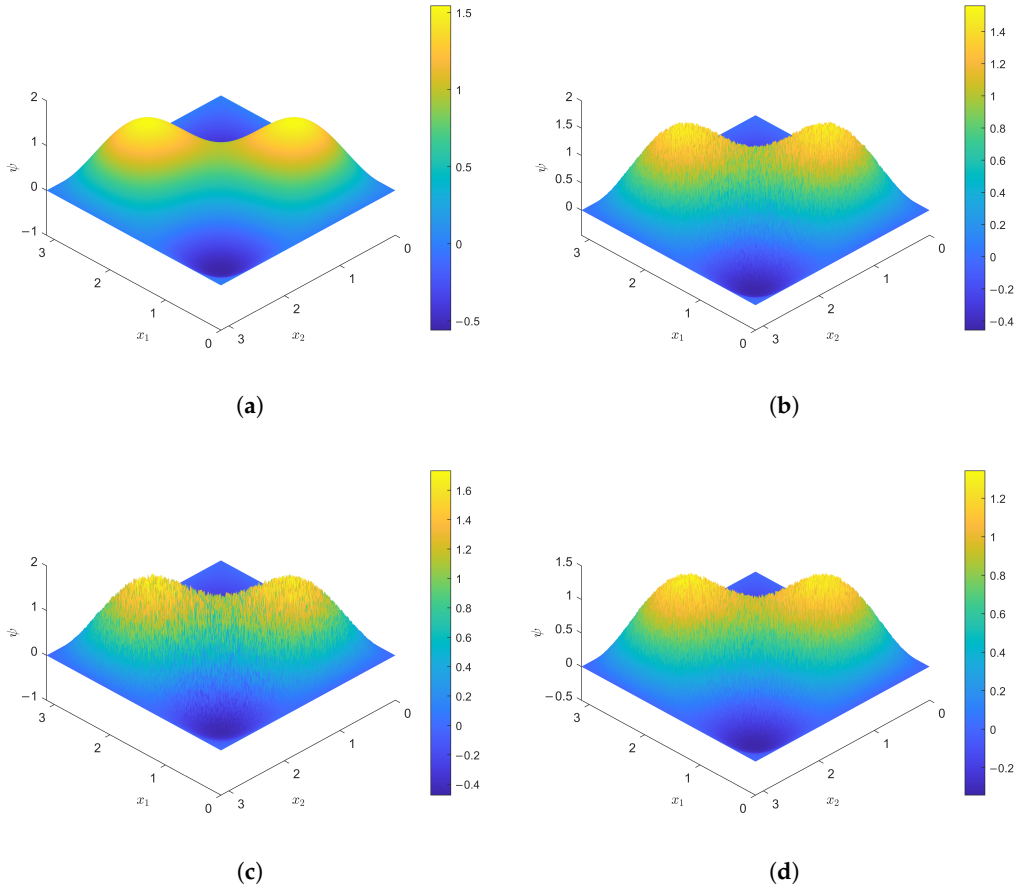


Figure 4. Results of experiments for Problem 2: approximate solution $\widetilde{\psi}(x_1, x_2)$ obtained by the regularized conjugate gradient method with noise level (**a**) $\delta = 0.00$; (**b**) $\delta = 0.01$; (**c**) $\delta = 0.02$; and (**d**) $\delta = 0.05$.

7. Discussion

According to the experiments, the relative error of the direct problem solution decreases with finer grid size. This indicates the experimental confirmation of convergence of the finite difference scheme.

In the case of two-dimensional problem solving, SLAE takes a significantly larger time (up to 600) than computing the right-hands part for SLAE. This makes the parallel implementation of the SLAE solver more important than the optimization of the procedures for computing the fractional derivative.

Experiments show that the parallel matrix sweep method for solving the SLAE has good parallel efficiency. The minimal computing time is achieved by using eight OpenMP threads on an eight-core processor.

The parallel code performance is mainly limited by memory bandwidth. Adding more than eight threads does not reduce the computing time. The largest speed up is only three-fold for 512×512 spatial grid. Figure 5 presents the roofline analysis performed by the Intel Advisor tool. Most subroutines of the parallel code lie primarily below and near the slanted line that represents DRAM bandwidth. This indicates that the code is memory-bound.

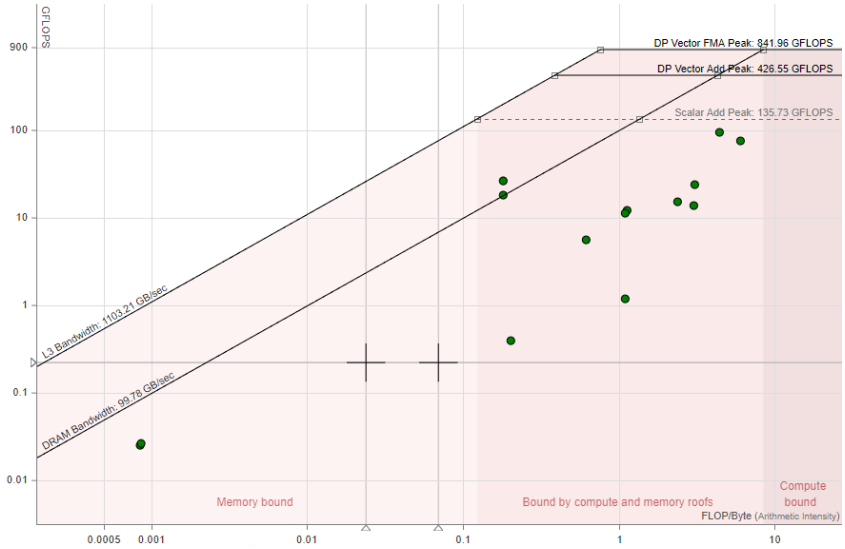


Figure 5. Roofline analysis for various subroutines (represented by dots) of parallel code for 16 OpenMP threads.

Several approaches can be used to overcome this limitation. Computing hardware with large memory bandwidth, such as graphics processors (GPU), can be used. Central processors with DDR4 or DDR5 RAM can achieve up to 100 GB/s, while modern GPUs have a bandwidth of 1000 GB/s or higher. Another option is to use massive distributed memory systems. Since each node works independently, the memory speed of individual node is effectively summed. Moreover, it enables larger problems to be solved with data that cannot be accommodated in the memory of a single computing node.

The experiments in Tables 3 and 4 show that for the model problems, the regularized conjugate gradient method allows us to solve the inverse problem even with noised data. The results are comparable with other works in terms of accuracy (for example, see [23], Table 3).

We also note that while this work is devoted to the case of the two-dimensional equation, the results may be extended to three-dimensional elliptic equations. The structure of matrix A in Equation (11) will remain block tridiagonal, but the inner structure of the blocks will be more complex.

8. Conclusions

In this work, we construct the parallel algorithm for solving the inverse problem of finding the space-dependent component of a source term in a two-dimensional fractional diffusion equation. The considered inverse problem is solved by the iterative conjugate gradient method. At each iteration, it is necessary to solve an auxiliary direct initial-boundary value problem. Applying the finite difference scheme, we reduce the initial-boundary value problem to solving an SLAE with block tridiagonal matrices at each subsequent time level. For the efficient solution of such SLAEs, we construct and implement the direct parallel matrix sweep method. Stability and correctness for the parallel matrix sweep method are established. In the two-dimensional case, computing the fractional derivative (the right-hand part of the SLAE) takes little time in comparison with solving the SLAE.

The algorithm is implemented for the multicore processors using the OpenMP technology. In the numerical experiments, we investigated the validity of numerical methods and the efficiency and speedup of the parallel algorithm. The utilization of the parallel sweep algorithm reduces the computing time by up to three times on a eight-core processor. Using Lavrentyev regularization method allows us to solve the inverse problem with a disturbed data.

In future, the authors plan to implement a similar approach to solving the retrospective inverse problem (identifying the initial value) for a fractional differential equation. The developed algorithms may be utilized for real applications. The parallel algorithms will be implemented on graphics processors.

Author Contributions: Conceptualization, E.N.A., M.A.S. and V.E.M.; methodology, E.N.A., M.A.S. and V.E.M.; validation, E.N.A., M.A.S., V.E.M. and Y.N.; formal analysis, E.N.A., M.A.S., V.E.M. and Y.N.; investigation, E.N.A., V.E.M. and Y.N.; resources, E.N.A., M.A.S. and V.E.M.; writing—original draft preparation, E.N.A., M.A.S. and V.E.M.; writing—review and editing, E.N.A., M.A.S., V.E.M. and Y.N.; supervision, E.N.A. and M.A.S.; project administration, V.E.M.; and funding acquisition, M.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: The second author (M.A.S.) and fourth author (Y.N.) were financially supported by the Ministry of Science and Higher Education of the Republic of Kazakhstan (project AP09258836). The first author (E.N.A.) and third author (V.E.M.) received no external funding.

Data Availability Statement: The data presented in this study are the model data. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Machado, J.T.; Galhano, A.; Trujillo, J. Science metrics on fractional calculus development since 1966. *Fract. Calc. Appl. Anal.* **2013**, *16*, 479–500. [CrossRef]
2. Podlubny, I. Fractional differential equations. *Math. Sci. Eng.* **1999**, *198*, 41–119.
3. Metzler, R.; Jeon, J.H.; Cherstvy, A.G.; Barkai, E. Anomalous diffusion models and their properties: Non-stationarity, non-ergodicity, and ageing at the centenary of single particle tracking. *Phys. Chem. Chem. Phys.* **2014**, *16*, 24128–24164. [CrossRef] [PubMed]
4. Tateishi, A.A.; Ribeiro, H.V.; Lenzi, E.K. The Role of Fractional Time-Derivative Operators on Anomalous Diffusion. *Front. Phys.* **2017**, *5*, 52. [CrossRef]
5. Yegenova, A.; Sultanov, M.; Brener, A. Nonlinear Wave Model for Transport Phenomena in Media with Non-local Effects. *Chem. Eng. Trans.* **2021**, *86*, 1201–1206. [CrossRef]
6. Li, X.; Han, X.; Wang, X. Numerical modeling of viscoelastic flows using equal low-order finite elements. *Comput. Methods Appl. Mech. Eng.* **2010**, *199*, 570–581. [CrossRef]
7. Maslovskaya, A.; Moroz, L. Time-fractional Landau–Khalatnikov model applied to numerical simulation of polarization switching in ferroelectrics. *Nonlinear Dyn.* **2023**, *111*, 4543–4557. [CrossRef]
8. Benson, D.A.; Wheatcraft, S.W.; Meerschaert, M.M. Application of a fractional advection-dispersion equation. *Water Resour. Res.* **2000**, *36*, 1403–1412. [CrossRef]
9. Laskin, N.; Lambadaris, I.; Harmantzis, F.; Devetsikiotis, M. Fractional Lévy motion and its application to network traffic modeling. *Comput. Netw.* **2002**, *40*, 363–375. [CrossRef]
10. Sun, H.; Zhang, Y.; Baleanu, D.; Chen, W.; Chen, Y. A new collection of real world applications of fractional calculus in science and engineering. *Commun. Nonlinear Sci. Numer. Simul.* **2018**, *64*, 213–231. [CrossRef]
11. Diethelm, K.; Ford, N.; Freed, A.; Luchko, Y. Algorithms for the fractional calculus: A selection of numerical methods. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 743–773. [CrossRef]
12. Baleanu, D.; Diethelm, K.; Scalas, E.; Trujillo, J.J. *Fractional Calculus: Models and Numerical Methods*; World Scientific: Singapore, 2012; Volume 3.
13. Li, C.; Zeng, F. *Numerical Methods for Fractional Calculus*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2019.
14. Sultanov, M.A.; Durdiev, D.K.; Rahmonov, A.A. Construction of an Explicit Solution of a Time-Fractional Multidimensional Differential Equation. *Mathematics* **2021**, *9*, 2052. [CrossRef]
15. Gong, C.; Bao, W.; Tang, G.; Jiang, Y.; Liu, J. A parallel algorithm for the two-dimensional time fractional diffusion equation with implicit difference method. *Sci. World J.* **2014**, *2014*, 219580. [CrossRef] [PubMed]
16. Akimova, E.N.; Misilov, V.E.; Sultanov, M.A. Regularized gradient algorithms for solving the nonlinear gravimetry problem for the multilayered medium. *Math. Methods Appl. Sci.* **2022**, *45*, 8760–8768. [CrossRef]

17. Li, X.; Su, Y. A parallel in time/spectral collocation combined with finite difference method for the time fractional differential equations. *J. Algorithms Comput. Technol.* **2021**, *15*, 17483026211008409. [CrossRef]
18. De Luca, P.; Galletti, A.; Ghehsareh, H.; Marcellino, L.; Raei, M. A GPU-CUDA framework for solving a two-dimensional inverse anomalous diffusion problem. *Parallel Comput. Technol. Trends* **2020**, *36*, 311.
19. Yang, X.; Wu, L. A New Kind of Parallel Natural Difference Method for Multi-Term Time Fractional Diffusion Model. *Mathematics* **2020**, *8*, 596. [CrossRef]
20. Berdyshev, A.S.; Sultanov, M.A. On Stability of the Solution of Multidimensional Inverse Problem for the Schrödinger Equation. *Math. Model. Nat. Phenom.* **2017**, *12*, 119–133. [CrossRef]
21. Samarskii, A.A.; Vabishchevich, P.N. *Numerical Methods for Solving Inverse Problems of Mathematical Physics*; Walter de Gruyter: Berlin, Germany, 2007; Volume 52.
22. Yang, F.; Ren, Y.P.; Li, X.X.; Li, D.G. Landweber iterative method for identifying a space-dependent source for the time-fractional diffusion equation. *Bound. Value Probl.* **2017**, *2017*, 163. [CrossRef]
23. Su, L.D.; Vasil'ev, V.I.; Jiang, T.S.; Wang, G. Identification of stationary source in the anomalous diffusion equation. *Inverse Probl. Sci. Eng.* **2021**, *29*, 3406–3422. [CrossRef]
24. Bazhlekova, E. An Inverse Source Problem for the Generalized Subdiffusion Equation with Nonclassical Boundary Conditions. *Fractal Fract.* **2021**, *5*, 63. [CrossRef]
25. Gong, X.; Wei, T. Reconstruction of a time-dependent source term in a time-fractional diffusion-wave equation. *Inverse Probl. Sci. Eng.* **2019**, *27*, 1577–1594. [CrossRef]
26. Nguyen, H.T.; Le, D.L.; Nguyen, V.T. Regularized solution of an inverse source problem for a time fractional diffusion equation. *Appl. Math. Model.* **2016**, *40*, 8244–8264. [CrossRef]
27. Sultanov, M.A.; Akimova, E.N.; Misilov, V.E.; Nurlanuly, Y. Parallel Direct and Iterative Methods for Solving the Time-Fractional Diffusion Equation on Multicore Processors. *Mathematics* **2022**, *10*, 323. [CrossRef]
28. Akimova, E.N.; Sultanov, M.A.; Misilov, V.E.; Nurlanuly, Y. Parallel sweep algorithm for solving direct and inverse problems for time-fractional diffusion equation. *Numer. Methods Program.* **2022**, *23*, 275–287. (In Russian) [CrossRef]
29. Zhang, Y. A finite difference method for fractional partial differential equation. *Appl. Math. Comput.* **2009**, *215*, 524–529. [CrossRef]
30. Slodička, M.; Šišková, K.; Bockstal, K.V. Uniqueness for an inverse source problem of determining a space dependent source in a time-fractional diffusion equation. *Appl. Math. Lett.* **2019**, *91*, 15–21. [CrossRef]
31. Samarskii, A.; Nikolaev, E. *Numerical Methods for Grid Equations, Volume I: Direct Methods*; Birkhäuser: Basel, Switzerland, 1989.
32. Akimova, E.N. Parallel Algorithms for Solving the Gravimetry, Magnetometry, and Elastisity Problems on Multiprocessor Systems with Distributed Memory. Doctor of Physical and Mathematical Sciences, Institute of Mathematics and Mechanics, Ural Branch of Russian Academy of Sciences, Ekaterinburg, Russia, 2009. (In Russian)
33. Saad, Y. *Iterative Methods for Sparse Linear Systems*; SIAM: Philadelphia, PA, USA, 2003.
34. Vasin, V.V.; Eremin, I.I. *Operators and Iterative Processes of Fejér Type: Theory and Applications*; De Gruyter: Berlin, Germany; New York, NY, USA, 2009. [CrossRef]
35. OpenMP Community. OpenMP Application Programming Interface Specification. Available online: https://www.openmp.org (accessed on 1 August 2023).
36. Intel Corporation. Accelerate Fast Math with Intel oneAPI Math Kernel Library. Available online: https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html (accessed on 1 August 2023).
37. Zhang, Y.N.; Sun, Z.Z. Alternating direction implicit schemes for the two-dimensional fractional sub-diffusion equation. *J. Comput. Phys.* **2011**, *230*, 8713–8728. [CrossRef]