energies

Article

Inverse Problem for a Two-Dimensional Anomalous Diffusion Equation with a Fractional Derivative of the Riemann–Liouville Type

Rafał Brociek [1,*], Agata Wajda [2] and Damian Słota [1]

[1] Department of Mathematics Applications and Methods for Artificial Intelligence, Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland; damian.slota@polsl.pl
[2] Department of Technologies and Installations for Waste Management, Faculty of Energy and Environmental Engineering, Silesian University of Technology, 44-100 Gliwice, Poland; agata.wajda@polsl.pl
* Correspondence: rafal.brociek@polsl.pl

Abstract: The article presents a method for solving the inverse problem of a two-dimensional anomalous diffusion equation with a Riemann–Liouville fractional-order derivative. In the first part of the present study, the authors present a numerical solution of the direct problem. For this purpose, a differential scheme was developed based on the alternating direction implicit method. The presented method was accompanied by examples illustrating its accuracy. The second part of the study concerned the inverse problem of recreating the model parameters, including the orders of the fractional derivative, in the anomalous diffusion equation. Equations of this type can be used to describe, inter alia, the heat conductivity in porous materials. The ant colony optimization algorithm was used to solve this problem. The authors investigated the impact of the distribution of measurement points, the use of different mesh sizes, and the input data errors on the obtained results.

Keywords: anomalous diffusion; inverse problem; fractional derivative; parameter identification

Citation: Brociek, R.; Wajda, A.; Słota, D. Inverse Problem for a Two-Dimensional Anomalous Diffusion Equation with a Fractional Derivative of the Riemann–Liouville Type. *Energies* **2021**, *14*, 3082. https://doi.org/10.3390/en14113082

Received: 19 April 2021
Accepted: 21 May 2021
Published: 26 May 2021

1. Introduction

In recent years, derivatives of fractional order have been frequently used. Many different definitions of such derivatives are available in the literature. In applications, the Caputo derivative and the Riemann–Liouville derivative are the most often used [1,2]. Fractional calculus is used, inter alia, in mathematical modeling of various problems in mechanics, physics, and biology [3–6], in control theory [7], in image processing [8], as well as in modeling the phenomena of heat transport [9–15]. Many mathematical models are based on differential equations containing fractional derivatives. Hence, the development of algorithms for solving differential equations with fractional derivatives is extremely important [16–20]. For example, in [16], the authors presented a solution of fractional pantograph differential equations. They used the operational matrix of the derivatives of the generalized Lucas polynomials and then transformed the problem into a system of equations. The paper also included the derivation of the numerical scheme with examples.

Direct problems for differential equations consist of solving a given equation (determining the state function) when all input data are known. Solving inverse problems, on the other hand, involves determining some of the input data on the basis of some information regarding the state function. In the literature, studies on inverse problems for differential equations with fractional derivatives are available. The first studies on these issues were by Murio [21,22]. In these papers, the one-dimensional problem was considered, and the mollification method was used to solve it. Zheng and Wei [23,24] presented a Fourier regularization and a convolution methods to recreate the boundary condition (function and its derivative) of a semi-infinite domain for the time-fractional inverse advection-dispersion equation. A similar issue is considered in the finite domain in [25], in this case a kernel-based meshless method was used. Xiong et al. [26] considered a two-dimensional case in the infinite domain. Yan and Yang [27] used an efficient Kansa-type method and presented numerical solution of two-dimensional time-fractional inverse diffusion problems. For the theoretical results of the stability and the uniqueness of the solution, a backward problem and inverse source problem for a multi-dimensional time-fractional diffusion equation were included in [28]. The authors of [29,30] managed the inverse problem for the two-dimensional time-fractional sideways heat equation in the infinite domain. Inverse problems for two-dimensional time-fractional diffusion equation were also considered in [29,31–33]. The study [34] considered the inverse problem for two-dimensional fractional partial differential equations. The method assumes the knowledge of the state function for a certain time interval at each point of the domain under consideration. A hybrid method based on modulating functions was used for the solution. The inverse problem for the heat equation in the two-dimensional domain with the Atangana–Baleanu fractional derivative was considered in [35]. In turn, the equation with the Hilfer derivative was considered in [36].

Research on one-dimensional issues prevails in the literature. Those on two-dimensional problems mainly regard the Caputo derivative in an infinite domain. There are a few articles on the inverse problem for the two-dimensional anomalous diffusion equation with the Riemann–Liouville derivative. Furthermore, the artificial intelligence algorithm was used in only one study in relation to time-fractional Schrödinger equations. The authors of this manuscript considered this rarely described case.

One of the main problems with using models with the fractional derivative is determining the order of the derivative. There are papers in which the authors determined it based on physical experiments [37]. To determine the order of the derivative, it may be useful to solve the inverse problem, which the authors present in this paper.

The article presents the consideration of the inverse problem for a two-dimensional anomalous diffusion equation with the Riemann–Liouville fractional derivative. The inverse problem in the study was to recreate the orders of two fractional derivatives and one of the parameters in the equation. Additional information, in this case, was the solution values at selected points of the domain. To solve the inverse problem, the direct problem should be repeatedly solved for the fixed values of the searched parameters. To solve the direct problem, the alternating direction implicit method was used [38], which allows for solving of multidimensional problem by the iterative solution of a number of one-dimensional problems. Using the given values of the solution and the values determined from the solution of the direct problem, a functional representing the error of the approximate solution was built. The argument for which the functional reached the minimum value was the sought solution for the inverse problem under consideration. The minimum of the functional was searched for using the ant colony optimization algorithm [39,40]. This is a probabilistic artificial intelligence algorithm inspired by the behavior of an ant colony. The authors of the study investigated the impact of the distribution of measurement points, the use of different mesh sizes, and the value of input data disturbances on the obtained results.

2. Space Fractional Diffusion Equation

The presented model in this section can be used to describe heat flow in a porous material with an irregular (fractal, chaotic) structure. In such a case, the heat flow can be modeled using fractional-order derivatives [10]. We considered a two-dimensional problem in which the derivative over space was a Riemann–Liouville type. Thus, the study considered the two-dimensional space fractional diffusion equation with the Riemann–Liouville derivative:

$$\begin{aligned} c\varrho\frac{\partial u(x,y,t)}{\partial t} &= \frac{\partial}{\partial x}\left(\lambda_{x1}(x,y)\frac{\partial^{\alpha}u(x,y,t)}{\partial x^{\alpha}} - \lambda_{x2}(x,y)\frac{\partial^{\alpha}u(x,y,t)}{\partial(-x)^{\alpha}}\right) \\ &+\frac{\partial}{\partial y}\left(\lambda_{y1}(x,y)\frac{\partial^{\beta}u(x,y,t)}{\partial y^{\beta}} - \lambda_{y2}(x,y)\frac{\partial^{\beta}u(x,y,t)}{\partial(-y)^{\beta}}\right) + f(x,y,t), \end{aligned} \tag{1}$$

specified in the following area $(x,y,t) \in \Omega \times [0,T]$, where $\Omega = [0,L_x] \times [0,L_y]$ and $\alpha,\beta \in (0,1)$. Initial-boundary conditions were attached to the above equation:

$$\begin{aligned} u(x,y,t)|_{\partial\Omega} &= 0, \;\; t \in (0,T], \\ u(x,y,t)|_{t=0} &= \varphi(x,y), \;\; (x,y) \in \Omega. \end{aligned} \tag{2}$$

It was also assumed that both constants c, ϱ and functions $\lambda_{x1}, \lambda_{x2}, \lambda_{y1}, \lambda_{y2} > 0$ are positive for all $(x,y) \in \Omega$. Equation (1) describes the phenomenon of anomalous diffusion, for example the phenomenon of heat conduction in porous materials [9–11]. In Equation (1), both the right- and left-hand Riemann–Liouville derivatives were used as fractional derivatives relative to space [1,2]:

$$\begin{aligned} \frac{\partial^\alpha u(x,y,t)}{\partial x^\alpha} &= \frac{1}{\Gamma(1-\alpha)} \frac{\partial}{\partial x} \int_0^x (x-\xi)^{-\alpha} u(\xi,y,t)\, d\xi, \\ \frac{\partial^\alpha u(x,y,t)}{\partial (-x)^\alpha} &= \frac{-1}{\Gamma(1-\alpha)} \frac{\partial}{\partial x} \int_x^{L_x} (\xi-x)^{-\alpha} u(\xi,y,t)\, d\xi, \end{aligned} \tag{3}$$

where $\alpha \in (0,1)$. Similar to the formula (3), we can define the fractional derivatives for a variable y and order $\beta \in (0,1)$.

Considering Equation (1) with the conditions (2) as a model describing the heat conductivity in porous materials, the following notations can be assumed:

- $\lambda_{x1}, \lambda_{x2}, \lambda_{y1}, \lambda_{y2}$—heat conductivity coefficients,
- c—specific heat of a medium,
- ϱ—density of a medium,
- u—a function describing the temperature distribution in space and time,
- f—an additional heat source.

3. Direct Problem

In this section, we present a way of solving Equation (1) numerically, and then, we introduce an example illustrating the accuracy of the method. For the discretization of the equation, the finite difference method supplemented with the appropriate discretization of Riemann–Liouville was used.

3.1. Numerical Method

Equation (1) can be written as follows:

$$\begin{aligned} c\varrho \frac{\partial u(x,y,t)}{\partial t} = &\left(\lambda_{x1}(x,y) \frac{\partial^{\alpha+1} u(x,y,t)}{\partial x^{\alpha+1}} + \lambda_{x2}(x,y) \frac{\partial^{\alpha+1} u(x,y,t)}{\partial (-x)^{\alpha+1}} \right) \\ &+ \left(\frac{\partial \lambda_{x1}(x,y)}{\partial x} \frac{\partial^\alpha u(x,y,t)}{\partial x^\alpha} - \frac{\partial \lambda_{x2}(x,y)}{\partial x} \frac{\partial^\alpha u(x,y,t)}{\partial (-x)^\alpha} \right) \\ &+ \left(\lambda_{y1}(x,y) \frac{\partial^{\beta+1} u(x,y,t)}{\partial y^{\beta+1}} + \lambda_{y2}(x,y) \frac{\partial^{\beta+1} u(x,y,t)}{\partial (-y)^{\beta+1}} \right) \\ &+ \left(\frac{\partial \lambda_{y1}(x,y)}{\partial y} \frac{\partial^\beta u(x,y,t)}{\partial y^\beta} - \frac{\partial \lambda_{y2}(x,y)}{\partial y} \frac{\partial^\beta u(x,y,t)}{\partial (-y)^\beta} \right) \\ &+ f(x,y,t). \end{aligned} \tag{4}$$

We commence with the discretization of the domain $[0,L_x] \times [0,L_y] \times [0,T]$. For this purpose, we introduce the following notations: $\Delta t = \frac{T}{N}$, $t^k = k\Delta t$, $k = 0,1,\ldots N$, $\Delta x = \frac{L_x}{M_x}$, $x_i = i\Delta x$, $i = 0,1,\ldots,M_x$, $\Delta y = \frac{L_y}{M_y}$, $y_j = j\Delta y$, $j = 0,1,\ldots,M_y$, where $N, M_x, M_y \in \mathbb{N}$, and

t_k, x_i, y_j are the mesh points. Using $u^k_{i,j}, f^k_{i,j}, \lambda^{i,j}_{1x}, \lambda^{i,j}_{2x}, \lambda^{i,j}_{1y}, \lambda^{i,j}_{2y}$, we denote the function values, respectively, $u, f, \lambda_{x1}, \lambda_{x2}, \lambda_{y1}$, and λ_{y2} in points (x_i, y_j, t^k).

We approximate the Riemann–Liouville derivative as follows [41] (the order of approximation $O((\Delta x)^2)$):

$$\left.\frac{\partial^\alpha u(x,y,t)}{\partial x^\alpha}\right|_{(x_i,y_j,t^k)} \approx \frac{1}{(\Delta x)^\alpha}\sum_{l=0}^{i+1}\omega^\alpha_l u(x_{i-l+1}, y_j, t^k), \tag{5}$$

$$\left.\frac{\partial^\alpha u(x,y,t)}{\partial (-x)^\alpha}\right|_{(x_i,y_j,t^k)} \approx \frac{1}{(\Delta x)^\alpha}\sum_{l=0}^{M_x-i+1}\omega^\alpha_l u(x_{i+l-1}, y_j, t^k), \tag{6}$$

where:

$$\omega^\alpha_0 = \frac{\alpha}{2}g^\alpha_0,\ \ \omega^\alpha_l = \frac{\alpha}{2}g^\alpha_l + \frac{2-\alpha}{2}g^\alpha_{l-1},\ \ l = 1,2,\ldots,$$
$$g^\alpha_0 = 1,\ \ g^\alpha_l = \left(1 - \frac{\alpha+1}{l}\right)g^\alpha_{l-1}\ \ l = 1,2,\ldots$$

The formulas for the fractional derivative with respect to the variable y look similar. The derivative with respect to time is approximated as follows (the order $O((\Delta t)^2)$):

$$\left.\frac{\partial u(x,y,t)}{\partial t}\right|_{(x_i,y_j,t^{k+\frac{1}{2}})} \approx \frac{u(x_i,y_j,t^{k+1}) - u(x_i,y_j,t^k)}{\Delta t}. \tag{7}$$

Taking into account the approximations (5)–(7), we can write:

$$\left.\frac{\partial^\alpha u(x,y,t)}{\partial x^\alpha}\right|_{(x_i,y_j,t^{k+\frac{1}{2}})} \approx \frac{1}{2(\Delta x)^\alpha}\left[\sum_{l=0}^{i+1}\omega^\alpha_l u(x_{i-l+1},y_j,t^{k+1}) + \sum_{l=0}^{i+1}\omega^\alpha_l u(x_{i-l+1},y_j,t^k)\right], \tag{8}$$

$$\left.\frac{\partial^{\alpha+1} u(x,y,t)}{\partial x^{\alpha+1}}\right|_{(x_i,y_j,t^{k+\frac{1}{2}})} \approx \frac{1}{2(\Delta x)^{\alpha+1}}\left[\sum_{l=0}^{i+1}\omega^{\alpha+1}_l u(x_{i-l+1},y_j,t^{k+1}) + \sum_{l=0}^{i+1}\omega^{\alpha+1}_l u(x_{i-l+1},y_j,t^k)\right], \tag{9}$$

$$\left.\frac{\partial^\alpha u(x,y,t)}{\partial (-x)^\alpha}\right|_{(x_i,y_j,t^{k+\frac{1}{2}})} \approx \frac{1}{2(\Delta x)^\alpha}\left[\sum_{l=0}^{M_x-i+1}\omega^\alpha_l u(x_{i-l+1},y_j,t^{k+1}) + \sum_{l=0}^{M_x-i+1}\omega^\alpha_l u(x_{i-l+1},y_j,t^k)\right], \tag{10}$$

$$\begin{aligned}\left.\frac{\partial^{\alpha+1} u(x,y,t)}{\partial (-x)^{\alpha+1}}\right|_{(x_i,y_j,t^{k+\frac{1}{2}})} \approx \frac{1}{2(\Delta x)^{\alpha+1}}&\left[\sum_{l=0}^{M_x-i+1}\omega^{\alpha+1}_l u(x_{i-l+1},y_j,t^{k+1})\right.\\ &\left.+\sum_{l=0}^{M_x-i+1}\omega^{\alpha+1}_l u(x_{i-l+1},y_j,t^k)\right].\end{aligned} \tag{11}$$

For simplicity, we introduce the following notation:

$$\delta^\alpha_x u^k_{i,j} = \frac{1}{2(\Delta x)^\alpha}\left[\lambda'^{\,i,j}_{x\,x1}\sum_{l=0}^{i+1}\omega^\alpha_l u^k_{i-l+1} - \lambda'^{\,i,j}_{x\,x2}\sum_{l=0}^{M_x-i+1}\omega^\alpha_l u^k_{i+l-1}\right], \tag{12}$$

$$\bar{\delta}^{\alpha+1}_x u^k_{i,j} = \frac{1}{2(\Delta x)^{\alpha+1}}\left[\lambda^{i,j}_{x1}\sum_{l=0}^{i+1}\omega^{\alpha+1}_l u^k_{i-l+1} + \lambda^{i,j}_{x2}\sum_{l=0}^{M_x-i+1}\omega^{\alpha+1}_l u^k_{i+l-1}\right]. \tag{13}$$

The approximation formulas for the derivative of the variable y are analogous. Then, the difference scheme of the order $O((\Delta t)^2 + (\Delta x)^2 + (\Delta y)^2)$ for Equation (4) takes the form:

$$\begin{aligned} c\varrho \frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\Delta t} &= \overline{\delta}_x^{\alpha+1} u_{i,j}^{k+1} + \delta_x^{\alpha} u_{i,j}^{k+1} + \overline{\delta}_x^{\alpha+1} u_{i,j}^{k} + \delta_x^{\alpha} u_{i,j}^{k} \\ &+ \overline{\delta}_y^{\beta+1} u_{i,j}^{k+1} + \delta_y^{\beta} u_{i,j}^{k+1} + \overline{\delta}_y^{\beta+1} u_{i,j}^{k} + \delta_y^{\beta} u_{i,j}^{k} + f_{i,j}^{k+\frac{1}{2}}. \end{aligned} \tag{14}$$

After transforming the above equation, we can write:

$$\begin{aligned} &(1 - \frac{\Delta t}{c\varrho}\overline{\delta}_x^{\alpha+1} - \frac{\Delta t}{c\varrho}\delta_x^{\alpha} - \frac{\Delta t}{c\varrho}\overline{\delta}_y^{\beta+1} - \frac{\Delta t}{c\varrho}\delta_y^{\beta}) u_{i,j}^{k+1} \\ &= (1 + \frac{\Delta t}{c\varrho}\overline{\delta}_x^{\alpha+1} + \frac{\Delta t}{c\varrho}\delta_x^{\alpha} + \frac{\Delta t}{c\varrho}\overline{\delta}_y^{\beta+1} + \frac{\Delta t}{c\varrho}\delta_y^{\beta}) u_{i,j}^{k} + \frac{\Delta t}{c\varrho} f_{i,j}^{k+\frac{1}{2}}, \end{aligned} \tag{15}$$

for $i = 1, 2, \ldots, M_x - 1$, $j = 1, 2, \ldots, M_y - 1$ and $k = 0, 1, \ldots, N - 1$.

We write the above difference scheme in matrix form. For this purpose, we define two matrices:

$$R_x(l) = (r_{i,j}^{x}(l))_{(M_x-1)\times(M_x-1)}, \quad l = 1, 2, \ldots, M_y - 1,$$

$$R_y(l) = (r_{i,j}^{y}(l))_{(M_y-1)\times(M_y-1)}, \quad l = 1, 2, \ldots, M_x - 1.$$

where:

$$r_{i,j}^{x}(l) = \begin{cases} \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha+1}} \lambda_{x1}^{i,l} \omega_{i-j+1}^{\alpha+1} + \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha}} \lambda_{x1}^{\prime x\; i,l} \omega_{i-j+l}^{\alpha}, & j < i-1, \\ \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha+1}} \left(\lambda_{x1}^{i,l}\omega_2^{\alpha+1} + \lambda_{x2}^{i,l}\omega_0^{\alpha+1}\right) + \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha}} \left(\lambda_{x1}^{\prime x\; i,l}\omega_2^{\alpha} - \lambda_{x2}^{\prime x\; i,l}\omega_0^{\alpha}\right), & j = i-1, \\ \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha+1}} \left(\lambda_{x1}^{i,l}\omega_1^{\alpha+1} + \lambda_{x2}^{i,l}\omega_1^{\alpha+1}\right) + \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha}} \left(\lambda_{x1}^{\prime x\; i,l}\omega_1^{\alpha} - \lambda_{x2}^{\prime x\; i,l}\omega_1^{\alpha}\right), & j = i, \\ \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha+1}} \left(\lambda_{x1}^{i,l}\omega_0^{\alpha+1} + \lambda_{x2}^{i,l}\omega_2^{\alpha+1}\right) + \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha}} \left(\lambda_{x1}^{\prime x\; i,l}\omega_0^{\alpha} - \lambda_{x2}^{\prime x\; i,l}\omega_2^{\alpha}\right), & j = i+1, \\ \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha+1}} \lambda_{x2}^{i,l} \omega_{j-i+1}^{\alpha+1} - \frac{-\Delta t}{2c\varrho(\Delta x)^{\alpha}} \lambda_{x2}^{\prime x\; i,l} \omega_{j-i+l}^{\alpha}, & j > i+1. \end{cases} \tag{16}$$

$$r_{i,j}^{y}(l) = \begin{cases} \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta+1}} \lambda_{y1}^{l,i} \omega_{i-j+1}^{\beta+1} + \frac{-\Delta t}{2c\varrho(\Delta y)^{\alpha}} \lambda_{y1}^{\prime y\; i,l} \omega_{i-j+l}^{\beta}, & j < i-1, \\ \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta+1}} \left(\lambda_{y1}^{l,i}\omega_2^{\beta+1} + \lambda_{y2}^{l,i}\omega_0^{\beta+1}\right) + \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta}} \left(\lambda_{y1}^{\prime y\; l,i}\omega_2^{\beta} - \lambda_{y2}^{\prime y\; l,i}\omega_0^{\beta}\right), & j = i-1, \\ \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta+1}} \left(\lambda_{y1}^{l,i}\omega_1^{\beta+1} + \lambda_{y2}^{l,i}\omega_1^{\beta+1}\right) + \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta}} \left(\lambda_{y1}^{\prime y\; l,i}\omega_1^{\beta} - \lambda_{y2}^{\prime y\; l,i}\omega_1^{\beta}\right), & j = i, \\ \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta+1}} \left(\lambda_{y1}^{l,i}\omega_0^{\beta+1} + \lambda_{y2}^{l,i}\omega_2^{\beta+1}\right) + \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta}} \left(\lambda_{y1}^{\prime y\; l,i}\omega_0^{\beta} - \lambda_{y2}^{\prime y\; l,i}\omega_2^{\beta}\right), & j = i+1, \\ \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta+1}} \lambda_{y2}^{l,i} \omega_{j-i+1}^{\beta+1} - \frac{-\Delta t}{2c\varrho(\Delta y)^{\beta}} \lambda_{y2}^{\prime y\; i,l} \omega_{j-i+l}^{\beta}, & j > i+1. \end{cases} \tag{17}$$

The next step is to define block matrices S and H. We start with the matrix S (dimension $\left[(M_y - 1) \cdot (M_x - 1)\right] \times \left[(M_y - 1) \cdot (M_x - 1)\right]$), which is a diagonal block matrix, i.e., it contains matrices $R_x(l)$, $l = 1, 2, \ldots, M_y - 1$ on the main diagonal and zeros in other places.

$$\begin{bmatrix} R_x(1) & 0 & \ldots & 0 \\ 0 & R_x(2) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & R_x(M_y - 1) \end{bmatrix} \tag{18}$$

Below is the form of matrix H (matrix H has the same dimension as matrix S). Matrix H is a block matrix where each block is a diagonal matrix of dimension $(M_x - 1) \times (M_x - 1)$.

$$\begin{bmatrix} r^y_{1,1}(1) & \cdots & 0 & & r^y_{1,M_y-1}(1) & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & r^y_{1,1}(M_x-1) & & 0 & \cdots & r^y_{1,M_y-1}(M_x-1) \\ & \vdots & & \ddots & & \cdots & \\ r^y_{M_y-1,1}(1) & \cdots & 0 & & r^y_{M_y-1,M_y-1}(1) & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & r^y_{M_y-1,1}(M_x-1) & & 0 & \cdots & r^y_{M_y-1,M_y-1}(M_x-1) \end{bmatrix} \tag{19}$$

We can write the difference scheme (15) in matrix form as follows:

$$(I+S+H)u^{k+1} = (I-S-H)u^k + \frac{\Delta t}{c\varrho} f^{k+\frac{1}{2}}, \quad k = 0,1,\ldots \tag{20}$$

where:

$$u^k = [u^k_{1,1}, u^k_{2,1}, \ldots, u^k_{M_x-1,1}, \ldots, u^k_{1,M_y-1}, u^k_{2,M_y-1}, \ldots u_{M_x-1,M_y-1}]^T,$$

$$f^{k+\frac{1}{2}} = [f^{k+\frac{1}{2}}_{1,1}, f^{k+\frac{1}{2}}_{2,1}, \ldots, f^{k+\frac{1}{2}}_{M_x-1,1}, \ldots, f^{k+\frac{1}{2}}_{1,M_y-1}, f^{k+\frac{1}{2}}_{2,M_y-1}, \ldots, f^{k+\frac{1}{2}}_{M_x-1,M_y-1}]^T.$$

The matrices appearing in the difference scheme (20) are quite large; as a consequence, the resulting system of equations will be time consuming to solve, which is shown in Section 3.2. Hence, we applied the alternating direction implicit method (ADIM) to the difference scheme (15), which significantly reduces the computation time. This will be an important issue in the case of solving the inverse problem, in which direct problem should be solved repeatedly. We write the scheme (15) in the form of the directional separation product:

$$\begin{aligned} &(1 - \frac{\Delta}{c\varrho}\overline{\delta}_x^{\alpha+1} - \frac{\Delta}{c\varrho}\delta_x^{\alpha})(1 - \frac{\Delta}{c\varrho}\overline{\delta}_y^{\beta+1} - \frac{\Delta}{c\varrho}\delta_y^{\beta})u^{k+1}_{i,j} \\ &= (1 + \frac{\Delta}{c\varrho}\overline{\delta}_x^{\alpha+1} + \frac{\Delta}{c\varrho}\delta_x^{\alpha})(1 + \frac{\Delta}{c\varrho}\overline{\delta}_y^{\beta+1} + \frac{\Delta}{c\varrho}\delta_y^{\beta})u^k_{i,j} + \frac{\Delta t}{c\varrho} f^{k+\frac{1}{2}}_{i,j}, \\ &i = 1,2,\ldots,M_x-1, \;\; j = 1,2,\ldots,M_y-1, \;\; k = 0,1,\ldots. \end{aligned} \tag{21}$$

Then, the above scheme can be broken in two parts and solved, respectively, first in direction x and, afterwards, in direction y. As a consequence, the resulting matrices for the system of equations will have significantly lower dimensions than in the case of the scheme (15). The solution algorithm consists of two successive steps:

- for each fixed y_j, we solve the scheme in the direction x. As a result, we obtain a temporary solution $\widetilde{u}^{k+1}_{i,j}$:

$$(1 - \tfrac{\Delta}{c\varrho}\overline{\delta}_x^{\alpha+1} - \tfrac{\Delta}{c\varrho}\delta_x^{\alpha})\widetilde{u}^{k+1}_{i,j} = (1 + \tfrac{\Delta}{c\varrho}\overline{\delta}_x^{\alpha+1} + \tfrac{\Delta}{c\varrho}\delta_x^{\alpha})(1 + \tfrac{\Delta}{c\varrho}\overline{\delta}_y^{\beta+1} + \tfrac{\Delta}{c\varrho}\delta_y^{\beta})u^k_{i,j} + \tfrac{\Delta t}{c\varrho} f^{k+\frac{1}{2}}_{i,j}, \tag{22}$$

- then, for each fixed x_i, we solve the scheme in direction y:

$$(1 - \frac{\Delta}{c\varrho}\overline{\delta}_y^{\beta+1} - \frac{\Delta}{c\varrho}\delta_y^{\beta})u^{k+1}_{i,j} = \widetilde{u}^{k+1}_{i,j}. \tag{23}$$

We can represent this process symbolically by means of Figure 1. For the boundary nodes and the initial condition, we assumed:

$$u^{k+1}_{0,j} = u^{k+1}_{M_x,j} = u^{k+1}_{i,0} = u^{k+1}_{i,M_y} = 0,$$

$$u^0_{i,j} = \varphi(i\Delta x, j\Delta y) = \varphi_{i,j}.$$

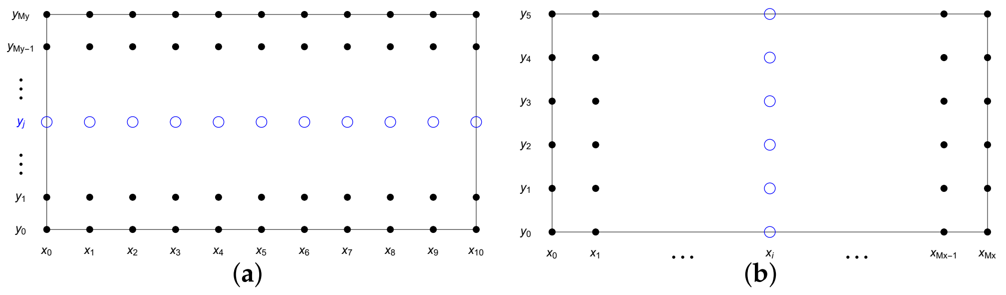

Figure 1. Numerical solution in the horizontal direction (for a fixed node y_j) (**a**) and the vertical direction (for a fixed node x_i) (**b**).

Furthermore, for the ADIM method, we write the appropriate matrix equations. First, for each $l = 1, 2, \ldots, M_x - 1$, we define auxiliary vectors u_l^*:

$$(I - R_y(l))u_l^k = u_l^*, \tag{24}$$

where $u_l^k = [u_{l,1}^k, u_{l,2}^k, \ldots, u_{l,M_y-1}^k]^T$, $u_l^* = [u_{l,1}^{*k}, u_{l,2}^{*k}, \ldots u_{l,M_y-1}^{*k}]^T$. Hence, we obtain an auxiliary matrix $U^{*k} = (u_{i,j}^{*k})$ of dimension $(M_x - 1) \times (M_y - 1)$. Then, the scheme (22) can be written in the following matrix form (for $p = 1, 2, \ldots, M_y - 1$):

$$(I + R_x(p))\widetilde{u}_p^k = (I - R_x(p))u_p^{**} + \frac{\Delta t}{c\varrho} f_p^{k+1}, \tag{25}$$

where the temporary solution takes the form $\widetilde{u}_p^k = [\widetilde{u}_{1,p}^k, \widetilde{u}_{2,p}^k, \ldots, \widetilde{u}_{M_x-1,p}^k]^T$, and $u_p^{**} = [u_{1,p}^{*k}, u_{2,p}^{*k}, \ldots, u_{M_x-1,p}^{*k}]^T$, $f_p^{k+\frac{1}{2}} = [f_{1,p}^{k+\frac{1}{2}}, f_{2,p}^{k+\frac{1}{2}}, \ldots f_{M_x-1,p}^{k+\frac{1}{2}}]^T$. In this step, we solved $M_y - 1$ systems of equations, each of $(M_x - 1) \times (M_x - 1)$ dimension. Afterwards, we present the scheme (23) in direction y in matrix form (for $l = 1, 2, \ldots, M_x - 1$):

$$(I + R_y(l))u_l^{k+1} = (I - R_y(l))\widetilde{u}_l^{*k}, \tag{26}$$

where $u_l^{k+1} = [u_{l,1}^{k+1}, u_{l,2}^{k+1}, \ldots, u_{l,M_y-1}^{k+1}]^T$ and $\widetilde{u}_l^{*k} = [\widetilde{u}_{l,1}^k, \widetilde{u}_{l,2}^k, \ldots, \widetilde{u}_{l,M_y-1}^k]^T$. From this step of the algorithm, we solved $M_x - 1$ systems of equations with dimensions $(M_y - 1) \times (M_y - 1)$ each. The Bi-CGSTAB [42,43] method was used to solve the equation systems, which also influenced the computation time. A comparison with the Gauss algorithm is presented in the next section. The proof of the convergence of the above difference schemes can be found in [38].

3.2. Numerical Results

In this section, we present the results of the numerical solution of the problem (1) and (2). The example assumes the following data:

$$\lambda_{x1} = \lambda_{x2} = \lambda_{y1} = \lambda_{y2} = 240, \;\; c = 900, \;\; \varrho = 2100,$$
$$\alpha = 0.8, \;\; \beta = 0.6, \;\; \varphi(x, y) = u(x, y, 0) = 0,$$

$$
\begin{aligned}
f(x,y,t) = \frac{3000000}{1309}\Bigg(&82467(x-2)^2x^2(y-1)^2y^3\cos\left(\frac{t}{100}\right) \\
&-\frac{1904\sqrt[5]{x}\left(25x^2-55x+22\right)(y-1)^2y^3\sin\left(\frac{t}{100}\right)}{\Gamma\left(\frac{1}{5}\right)} \\
&-\frac{1904\sqrt[5]{2-x}\left(25x^2-45x+12\right)(y-1)^2y^3\sin\left(\frac{t}{100}\right)}{\Gamma\left(\frac{1}{5}\right)} \\
&-\frac{220(x-2)^2x^2\left(125y^2-170y+51\right)y^{7/5}\sin\left(\frac{t}{100}\right)}{\Gamma\left(\frac{2}{5}\right)} \\
&-\frac{44(x-2)^2x^2(1-y)^{2/5}\left(625y^3-600y^2+90y+4\right)\sin\left(\frac{t}{100}\right)}{\Gamma\left(\frac{2}{5}\right)}\Bigg).
\end{aligned}
\tag{27}
$$

For the data defined above, the exact solution (1) and (2) is the function:

$$u(x,y,t) = 10,000(2-x)^2x^2(1-y)^2y^3\sin\left(\frac{t}{100}\right),$$

where the domain under consideration is as follows: $x \in [0,2],\ y \in [0,1],\ t \in [0,200]$.

Errors are marked as follows:

$$\Delta_{max} = \max(|u_{i,j}^k - ue_{i,j}^k|),$$

$$\Delta_{avg} = \frac{\sum\limits_k\sum\limits_i\sum\limits_j |u_{i,j}^k - ue_{i,j}^k|}{(M_x+1)(M_y+1)(N+1)},$$

where $i = 0,1,\ldots,M_x,\ j = 0,1,\ldots,M_y,\ k = 0,1,\ldots,N$, and $ue_{i,j}^k$ denotes exact values of the function u at the mesh points. First of all, we compared the results obtained with and without the ADIM for different meshes. We also provided computation times. As can be seen in Table 1, the results obtained with or without ADIM were similar; the differences were slight. However, the difference in the computation time was of considerable size; for example, for the mesh $100 \times 100 \times 100$, in the case of the ADIM, this time was about 9 s, while for the method without ADIM, this time was about 453 s. The difference was enormous, and considering that in the inverse problem, it is necessary to solve the direct problem repeatedly, the use of ADIM was essential.

Table 1. Comparison of the results and the times of the algorithm computation depending on the use of ADIM.

Mesh	$\mathbf{\Delta_{max}}$	$\mathbf{\Delta_{max}}$	$\mathbf{\Delta_{avg}}$	$\mathbf{\Delta_{avg}}$	**CPU Time (ms)**	**CPU Time (ms)**
$M_x \times M_y \times N$	ADIM	without ADIM	ADIM	without ADIM	ADIM	without ADIM
$20 \times 20 \times 20$	1.50453	1.50458	0.17421	0.17476	32	183
$40 \times 40 \times 40$	0.59740	0.59752	0.05186	0.05202	348	3929
$80 \times 80 \times 80$	0.18129	0.18131	0.01449	0.01453	4161	7266
$100 \times 100 \times 100$	0.12042	0.12043	0.00954	0.00956	9143	453,438

Another important issue influencing the computation time was the appropriate selection of the method to solve the system of equations. In the presented algorithm, the Bi-CGSTAB method [42,43] was used to solve a system of linear equations. We compared the computation times of the algorithm in the case of using Bi-CGSTAB and the Gaussian elimination method (Table 2). In the case of mesh $200 \times 200 \times 200$, the use of the Bi-CGSTAB caused the computation to be performed approximately 5.3 times faster than using the Gaussian elimination method.

Table 2. Comparison of the algorithm computation times depending on the method used to solve the system of equations.

Mesh	Time (ms) Bi-CGSTAB	Time (ms) Gaussian Elimination
$20 \times 20 \times 20$	32	33
$40 \times 40 \times 40$	348	440
$80 \times 80 \times 80$	4161	9046
$100 \times 100 \times 100$	9143	25,005
$200 \times 200 \times 200$	154,650	822,644

4. Inverse Problem

In this section, we present a solution of the inverse problem involving the reconstruction of the selected parameters of the model described by Equation (1). The considered inverse problem consisted of selecting the unknown input parameters of the model in such a way that the model output (function u from Equation (1)) had predetermined values at selected points of the domain. A lack of information regarding the searched input parameters was compensated by the values of the function u at selected points of the domain. Due to the fact that the inverse problems are ill-conditioned, they are classified as quite difficult issues.

4.1. Formulation of the Problem

We considered the following equation:

$$c\varrho\frac{\partial u(x,y,t)}{\partial t} = \frac{\partial}{\partial x}\left(\lambda(x,y)\frac{\partial^{\alpha} u(x,y,t)}{\partial x^{\alpha}} - \lambda(x,y)\frac{\partial^{\alpha} u(x,y,t)}{\partial(-x)^{\alpha}}\right)$$
$$+\frac{\partial}{\partial y}\left(\lambda(x,y)\frac{\partial^{\beta} u(x,y,t)}{\partial y^{\beta}} - \lambda(x,y)\frac{\partial^{\beta} u(x,y,t)}{\partial(-y)^{\beta}}\right) + f(x,y,t),$$

where:

$$\lambda = 240, \quad \varrho = 2100, \quad \varphi(x,y) = u(x,y,0) = 0,$$

and function $f(x,y,t)$ is given by the formula (27). We considered the differential equation in the domain $(x,y,t) \in \Omega \times [0,200]$, where $\Omega = [0,2] \times [0,1]$. The unknown parameters of the model that should be determined were c, α, β. However, the values of the function u at selected points of the domain were known. In order to test the stability of the algorithm, we considered:

- two different locations of the measuring points (see Figure 2):

$$\{P1(0.2,0.9), P2(0.2,0.5), P3(0.2,0.1), P4(0.5,0.9), P5(0.5,0.5), P6(0.5,0.1);$$
$$P7(1.5,0.9), P8(1.5,0.5), P9(1.5,0.1), P10(1.8,0.9), P11(1.8,0.5), P12(1.8,0.1)\};$$
$$\{K1(0.4,0.8), K2(0.4,0.5), K3(0.4,0.2), K4(1.0,0.5);$$
$$K5(1.6,0.8), K6(1.6,0.5), K7(1.6,0.2)\};$$

- three types of meshes ($M_x \times M_y \times N$): $160 \times 160 \times 250$, $100 \times 100 \times 200$, and $80 \times 80 \times 100$;
- different types of measurement data disturbances (errors with a normal distribution): $0\%, 2\%, 5\%, 10\%$.

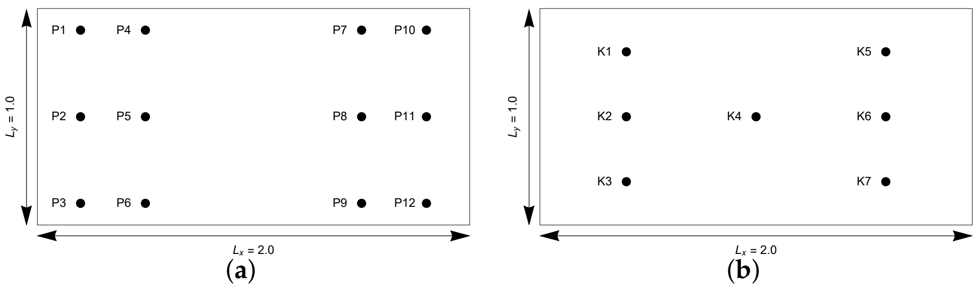

Figure 2. Two different arrangements of the measuring points.

In order to reconstruct the unknown parameters c, α, β, we developed a functional comparing the values of the numerical solution $u^k_{i,j}(c, \alpha, \beta)$ depending on the parameters sought with the measurement data $\overset{mk}{u}_{i,j}$:

$$J(c, \alpha, \beta) = \sum_{i,j}^{N_1} \sum_{k}^{N_2} \left(u^k_{i,j}(c, \alpha, \beta) - \overset{mk}{u}_{i,j} \right)^2, \tag{28}$$

where N_1 is the number of measurement points, and N_2 is the number of measurements. The values $u^k_{i,j}(c, \alpha, \beta)$ are obtained from the solution of the direct problem for fixed parameter values c, α, β.

4.2. *Objective Function Minimization*

In order to find minimum of the objective function (28), and then solve inverse problem, the Ant Colony Optimization (ACO) algorithm was used [39,40]. The ACO algorithm is inspired by behavior of an ant colony in nature. In the considered version of the algorithm, solutions, which in our case were vectors, are treated as the so-called pheromone spots. In the initial phase of the algorithm, these spots are randomly distributed in the considered domain. Then, they are sorted by quality, and each solution (pheromone spot) is assigned a probability of choice. The better the solution is, the higher the probability is. The probability is determined on the basis of the quality of the solution and the q parameter of the algorithm, which we can control. In the case of a low q, the best solution is preferred, and as q increases, the probability of choosing any solution becomes similar. As a result, we obtained an ordered set of solutions. Then, the iterative phase of the algorithm begins, in which each ant (which are M) selects a spot according to its probability and then transforms it to sample the neighborhood. In the considered algorithm, to transform the solution, we used the Gaussian distribution density function. As a result of the transformation, we obtained M new solutions (pheromone spots). Next, the solution set must be updated with new pheromone spots (solutions) by sorting all solutions according to quality, and then, M the worst solutions are deleted. We repeated the iterative process I times. Finally, we chose the best solution. The algorithm was adapted to parallel computations. A more detailed description of the algorithm can be found in [9].

5. Results

In the considered inverse problem, we determined the parameters α, β, c. The exact values of these parameters were respectively $0.8, 0.6, 900$. As we mentioned earlier, to test the stability of the algorithm, we assumed different arrangements of the measurement points, mesh sizes, and input data disturbances. For example, in Figure 3, we can see a comparison of the input data not disturbed by the error and disturbed by the 10% error from the measurement point $K4$.

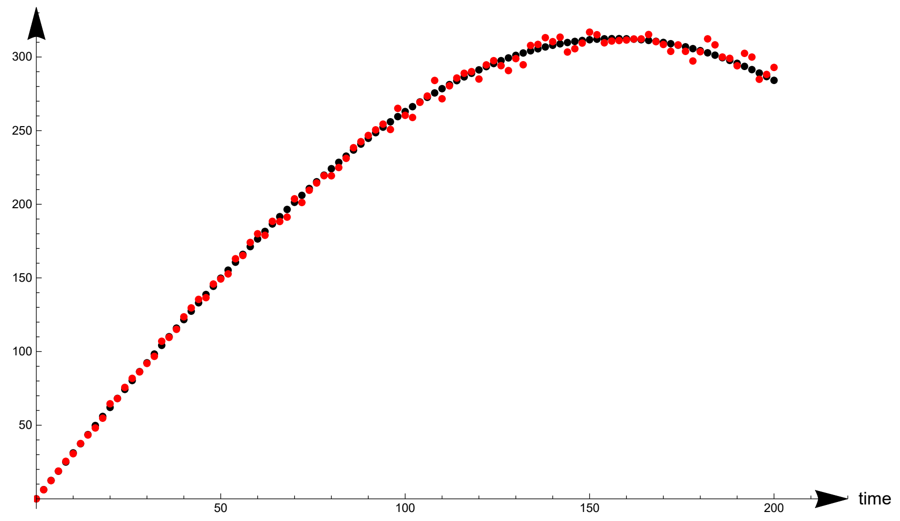

Figure 3. Disturbed (red dots) and exact (black dots) values of the function u at measuring point $K4$ (disturbed with the error 10%).

The following parameter values were adopted in the ant algorithm:

$$M = 32,\ L = 18,\ I = 30,\ \alpha \in (0, 0.99),\ \beta \in (0, 0.99),\ c \in [500, 1500].$$

Due to the probabilistic nature of the ant algorithm, we assumed five runs for the given settings, and then, the best one was selected from the obtained results. For this reason, in the tables with the results, the value of the standard deviation for the obtained values of the objective function was added. As can be seen, they were inconsiderable, which proved that the algorithm gave very similar results each time.

Table 3 shows the results of reconstructing the searched parameters for various meshes and the values of input data disturbances in the case of twelve measurement points. In each case, the errors of the results obtained were low and did not exceed 1%. For the input data without error, the results were the best. Even with 10% disturbance of the input data, the obtained results were satisfying, which proved the stability of the model and algorithm. Table 4 contains analogous data for the case of seven measuring points. As can be seen, the change in the arrangement of measurement points, as well as their number influenced the obtained results. In a few cases (for example, for the $100 \times 100 \times 200$ mesh and 10% disturbance), the errors in recreating selected parameters exceeded 1%, which did not happen in the case of twelve measuring points. It can therefore be concluded that the arrangement of the measuring points and their number had a significant impact on the results. At the same time, it should be noted that also in this case, the obtained errors of the reconstruction coefficient were at a satisfactory level. Only in five cases, the recreating error exceeded 1%.

We next verified the errors of the recreating function u in the measuring points for the recreated parameters α, β, c. In all cases, these errors were at a low level. A tendency can be seen that as input data noise increased, the recreating errors of the function u increased, although this was not always the case. A similar tendency applied to the mesh density; in general, the lowest reconstruction errors were obtained for the $160 \times 160 \times 250$ mesh and the highest for the $80 \times 80 \times 100$ mesh. Importantly, the average errors of the recreating function u turned out to be greater for the second arrangement of measurement points (seven measurement points) in ten of twelve cases (see Tables 5 and 6).

Table 3. Results of the calculations in the case of twelve measurements points P_i $(i = 1, 2, \ldots, 12)$: $\overline{\alpha}_i$—reconstructed value of x-directional order derivative α, $\overline{\beta}_i$—reconstructed value of y-directional order derivative β, $\overline{c}$—reconstructed value of parameter c, δ—relative error of reconstruction, J—value of the objective function, σ—standard deviation of the objective function.

Mesh Size	**Noise**	$\overline{\alpha}$	$\delta_{\overline{\alpha}}$ (%)	$\overline{\beta}$	$\delta_{\overline{\beta}}$ (%)	$\overline{c}$	$\delta_{\overline{c}}$(%)	J	σ_J
$160 \times 160 \times 250$	0%	0.80011	0.015	0.60011	0.018	899.98	0.002	0.00093	0.0047
	2%	0.79871	0.162	0.59952	0.079	899.67	0.035	441.96	0.0021
	5%	0.80171	0.213	0.60118	0.196	900.53	0.059	2840.82	0.0018
	10%	0.80107	0.134	0.60099	0.166	900.71	0.079	9988.16	0.0018
$100 \times 100 \times 200$	0%	0.80024	0.031	0.60026	0.044	899.96	0.004	0.00502	0.0126
	2%	0.80049	0.061	0.60104	0.174	900.35	0.039	360.43	0.0216
	5%	0.79768	0.289	0.59811	0.313	898.69	0.145	2204.71	0.0025
	10%	0.79216	0.979	0.59967	0.053	898.11	0.209	8451.98	0.0007
$80 \times 80 \times 100$	0%	0.80035	0.044	0.60038	0.064	899.96	0.004	0.00869	0.0003
	2%	0.80237	0.296	0.60115	0.191	900.78	0.087	196.75	0.0003
	5%	0.79833	0.207	0.60053	0.089	900.07	0.008	1177.09	0.0003
	10%	0.80066	0.083	0.60127	0.212	901.32	0.147	1109.12	0.0002

Table 4. Results of calculations in the case of seven measurements points K_i $(i = 1, 2, \ldots, 7)$: $\overline{\alpha}_i$—reconstructed value of x-directional order derivative α, $\overline{\beta}_i$—reconstructed value of y-directional order derivative β, $\overline{c}$—reconstructed value of parameter c, δ—relative error of reconstruction, J—value of the objective function, σ—standard deviation of the objective function.

Mesh Size	**Noise**	$\overline{\alpha}$	$\delta_{\overline{\alpha}}$ (%)	$\overline{\beta}$	$\delta_{\overline{\beta}}$ (%)	$\overline{c}$	$\delta_{\overline{c}}$(%)	J	σ_J
$160 \times 160 \times 250$	0%	0.80005	0.007	0.60006	0.011	899.99	0.001	0.00547	0.1445
	2%	0.80159	0.198	0.59922	0.128	900.32	0.036	1013.49	0.0207
	5%	0.79495	0.631	0.59595	0.673	902.61	0.288	5350.01	0.2056
	10%	0.78547	1.815	0.60141	0.234	900.41	0.045	23,672.65	0.1913
$100 \times 100 \times 200$	0%	0.80021	0.026	0.60021	0.035	900.01	0.001	0.03699	0.1074
	2%	0.79795	0.255	0.60029	0.049	900.15	0.017	724.43	0.0241
	5%	0.77951	2.561	0.60181	0.302	900.21	0.024	4993.79	0.0071
	10%	0.76984	3.769	0.60046	0.077	897.53	0.274	19,397.31	0.0315
$80 \times 80 \times 100$	0%	0.79957	0.053	0.60042	0.071	899.98	0.002	0.02526	0.0249
	2%	0.80634	0.792	0.60042	0.071	900.28	0.031	400.94	0.0131
	5%	0.81852	2.315	0.59791	0.349	899.80	0.022	3072.09	0.0204
	10%	0.80758	0.948	0.59324	1.126	901.03	0.115	2152.77	0.0118

Table 5. Errors of reconstruction function u in measurement points for twelve measurement points (Δ_{avg}—average absolute error, $\Delta_{\max}$—maximum absolute error, δ_{avg}—average relative error, $\delta_{\max}$—maximum relative error).

		$160 \times 160 \times 250$		
	0%	**2%**	**5%**	**10%**
Δ_{avg}(K)	4.45×10^{-4}	1.62×10^{-2}	2.64×10^{-2}	2.79×10^{-2}
$\Delta_{\max}$(K)	1.54×10^{-3}	1.04×10^{-1}	1.84×10^{-1}	1.89×10^{-1}
		$100 \times 100 \times 200$		
	0%	**2%**	**5%**	**10%**
Δ_{avg}(K)	1.16×10^{-3}	2.03×10^{-2}	5.77×10^{-2}	7.42×10^{-2}
$\Delta_{\max}$(K)	3.08×10^{-3}	1.28×10^{-1}	3.86×10^{-1}	3.86×10^{-1}
		$80 \times 80 \times 100$		
	0%	**2%**	**5%**	**10%**
Δ_{avg}(K)	2.11×10^{-3}	2.81×10^{-2}	1.21×10^{-2}	4.83×10^{-2}
$\Delta_{\max}$(K)	6.94×10^{-3}	2.01×10^{-1}	8.89×10^{-2}	2.84×10^{-1}

Table 6. Errors of reconstruction function u in measurement points for seven measurement points (Δ_{avg}—average absolute error, Δ_{max}—maximum absolute error, δ_{avg}—average relative error, δ_{max}—maximum relative error).

	0%	**2%**	**5%**	**10%**
160 × 160 × 250				
Δ_{avg}(K)	1.25×10^{-3}	1.77×10^{-2}	1.41×10^{-1}	6.78×10^{-2}
Δ_{max}(K)	4.71×10^{-3}	5.71×10^{-2}	3.67×10^{-1}	2.89×10^{-1}
100 × 100 × 200				
Δ_{avg}(K)	3.52×10^{-3}	1.29×10^{-2}	8.67×10^{-2}	2.16×10^{-1}
Δ_{max}(K)	1.36×10^{-2}	3.92×10^{-2}	5.31×10^{-1}	1.74
80 × 80 × 100				
Δ_{avg}(K)	4.45×10^{-3}	3.19×10^{-2}	9.21×10^{-2}	1.29×10^{-1}
Δ_{max}(K)	1.53×10^{-2}	3.37×10^{-1}	3.33×10^{-1}	5.96×10^{-1}

Figures 4 and 5 show the distribution of the reconstruction errors of the function u at selected points of the domain ($K4, K6$ and $P5, P11$) for the 100 × 100 × 200 mesh. The greater the noise on the input data was, the greater these errors were. For exact data, the reconstruction errors were at a stable, low level. In the case of point $K6$, it can be seen that the reconstruction error for input data disturbed with a 5% noise error at the end of the process was greater than the corresponding error for a noise of 10%.

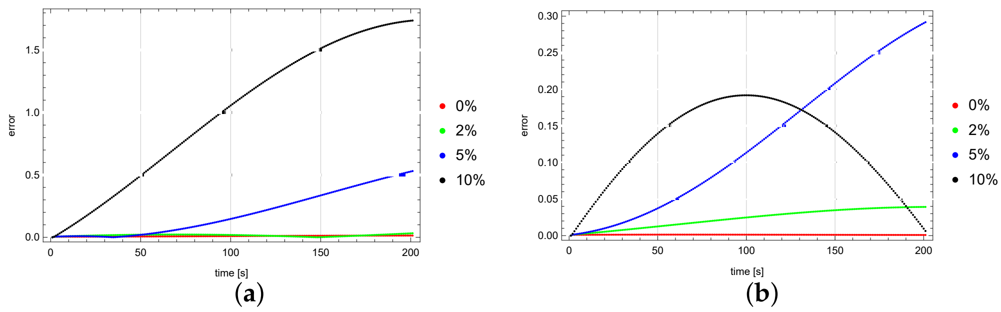

Figure 4. Errors in measurement points $K4$ (**a**) and $K6$ (**b**) in the case of a 100 × 100 × 200 mesh size.

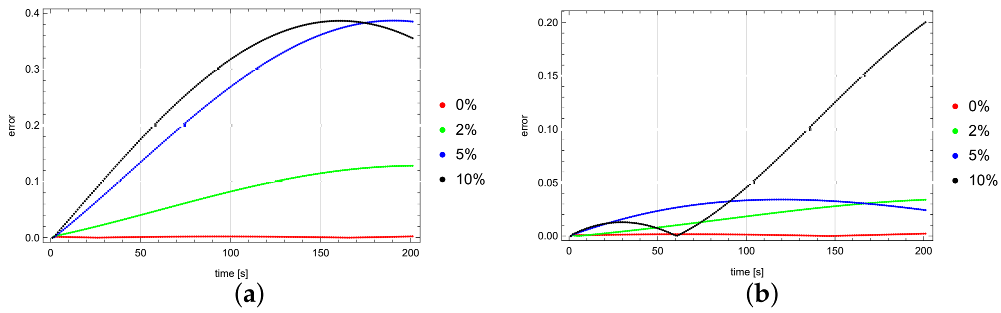

Figure 5. Errors in measurement points $P5$ (**a**) and $P11$ (**b**) in the case of a 100 × 100 × 200 mesh size.

Now, we present a way of achieving the solution by the ACO algorithm. Figures 6 and 7 show the values of the objective function in successive iterations of the algorithm for the case of the 100 × 100 × 200 mesh, seven measurement points, and various input data noise values. The values of the objective function for the first few iterations were relatively high; therefore, due to the scale, they are not marked in the figures. It can be seen that around 15 iterations, these values stabilized, and in the last iterations, the algorithm slightly improved the searched minimum. The selection of the parameters for the ant algorithm is quite important and was obtained based on the experience of the authors [40,44].

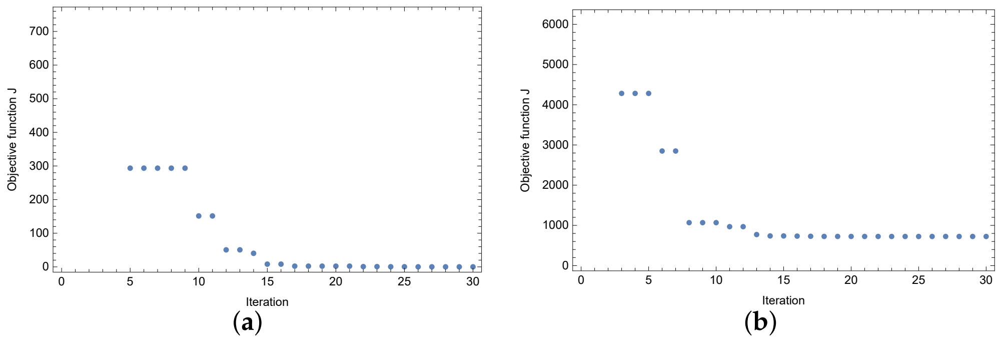

Figure 6. Values of the objective function in the case of 0% (**a**) and 2% (**b**) perturbation of the input data.

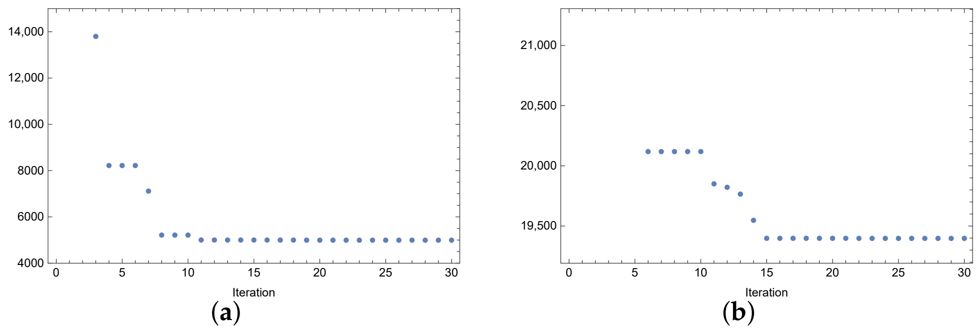

Figure 7. Values of the objective function in the case of 5% (**a**) and 10% (**b**) perturbation of the input data.

Finally, we present the influence of the recreated parameters on the values of the function u at the measuring point $K4$. For this purpose, we calculated the values of the function u at the point $K4$ for different values of the parameters α, β, c. Firstly, we determined the values of the parameters β, c, and we changed the values of the parameter $\alpha = 0.2, 0.4, 0.6, 0.8$. Then, we similarly executed with the parameter β at fixed values of the α, c. Finally, we set the α, β parameters, and the parameter c took the values $500, 700, \ldots, 1500$. As can be seen in Figures 8–10, a change in the value of the β parameter had a greater impact on the obtained values of the function u at point $K4$ than in the case of the α derivative order. Furthermore, introducing a change in the value of the parameter c had a significant impact on the function, especially in the case of lower values of the parameter c. It can also be observed that changing the α derivative parameter affected the u function to a lesser extent than the other parameters.

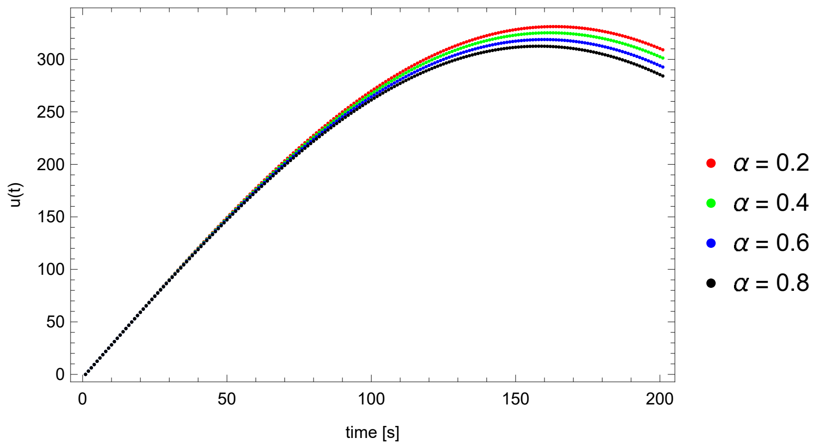

Figure 8. Values of the function u at point $K4$ for different values of the parameter α ($\beta = 0.6, c = 900$).

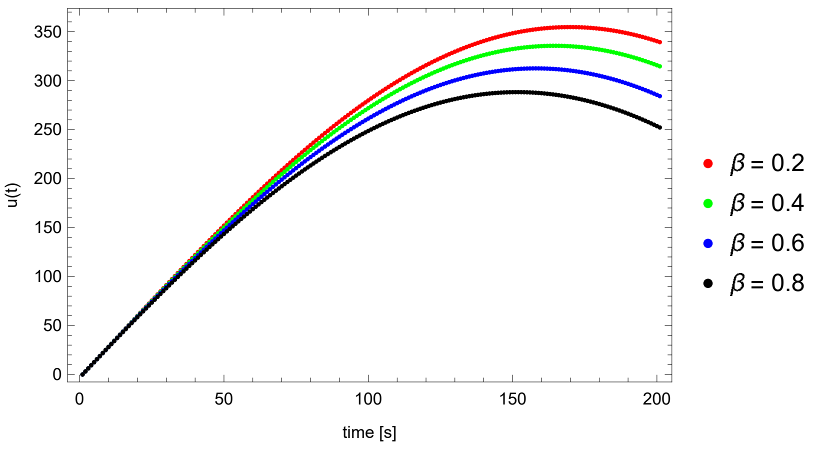

Figure 9. Values of the function u at point $K4$ for different values of the parameter β ($\alpha = 0.8, c = 900$).

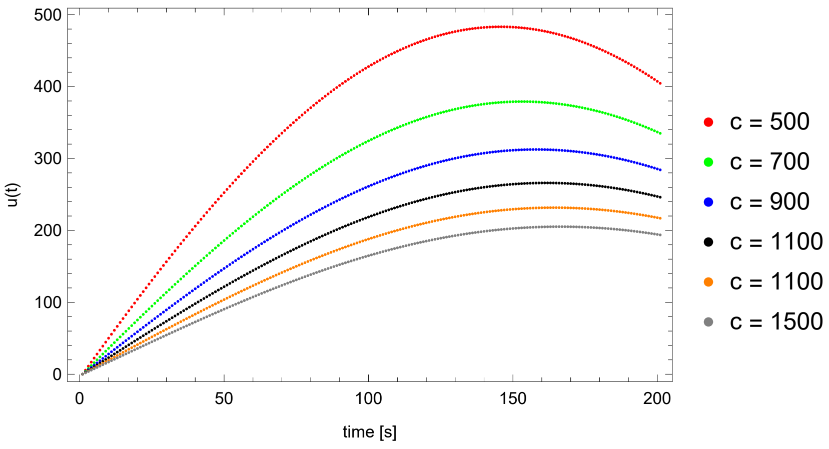

Figure 10. Values of the function u at point $K4$ for different values of the parameter c ($\beta = 0.6, \alpha = 0.8$).

6. Conclusions

The paper presented a numerical solution of a two-dimensional differential equation with a fractional derivative with respect to a space of the Riemann–Liouville type. This type of equation can describe the phenomenon of heat conduction in porous media, which the authors will also consider in further research. The presented numerical solution was also used to solve the inverse problem consisting of recreating the orders of the derivatives α, β and the parameter c. The presented results of the numerical experiment seemed satisfactory, and the algorithm was stable even for input data burdened with noise (measurement error).

The presented algorithm was characterized by faster calculations in relation to the classic differential scheme, resistance to falling into local minima compared to other minimization methods, low requirements for the objective function (it was not necessary to require continuity, differentiability, or convexity), and the stability against input data errors. The above-described properties can be treated as its advantages. The only downside was the computation time of the ACO algorithm.

In the further part of the research, the authors plan the following:

- continued tests of the proposed algorithm (reconstructing of more parameters);
- application of the model described in the study (differential equation) to model the heat flow process in porous materials;
- development of the algorithm to shorten the computation time;
- investigation of the influence of initial conditions (due to the fact that fractional derivatives contain the memory of past events), and thus the development of the model with the Caputo derivative to the time variable.

Author Contributions: Conceptualization, R.B. and D.S.; methodology, R.B.; software, R.B.; validation, R.B., D.S., and A.W.; formal analysis, D.S.; investigation, R.B., D.S., and A.W.; writing—original draft preparation, R.B., D.S., and A.W.; writing—review and editing, R.B., D.S., and A.W.; visualization, R.B.; funding acquisition, D.S. All authors read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

References

1. De Oliveira, E.C.; Tenreiro Machado, J.A. A Review of Definitions for Fractional Derivatives and Integral. *Math. Probl. Eng.* **2014**. [CrossRef]
2. Podlubny, I. *Fractional Differential Equations*; Academic Press: San Diego, CA, USA, 1999.
3. Carpinteri, A.; Mainardi, F. *Fractal and Fractional Calculus in Continuum Mechanics*; Springer: New York, NY, USA, 1997.
4. Hilfer, R. (Ed.) *Applications of Fractional Calculus in Physics*; World Scientific Publishing: Singapore, 2000.
5. Sun, H.G.; Zhang, Y.; Baleanu, D.; Chen, W.; Chen, Y.Q. A new collection of real world applications of fractional calculus in science and engineering. *Commun. Nonlinear Sci. Numer. Simul.* **2018**, *64*, 213–231. [CrossRef]
6. Sowa, M.; Majka, Ł. Ferromagnetic core coil hysteresis modeling using fractional derivatives. *Nonlinear Dyn.* **2020**, *101*, 775–793. [CrossRef]
7. Tenreiro Machado, J.A.; Silva, M.F.; Barbosa, R.S.; Jesus, I.S.; Reis, C.M.; Marcos, M.G.; Galhano, A.F. Some applications of fractional calculus in engineering. *Math. Probl. Eng.* **2010**, *2010*, 639801. [CrossRef]
8. Jindal, N.; Singh, K. Applicability of fractional transforms in image processing—Review, technical challenges and future trends. *Multimed. Tools Appl.* **2019**, *78*, 10673–10700. [CrossRef]
9. Brociek, R.; Słota, D.; Król, M.; Matula, G.; Kwaśny, W. Comparison of mathematical models with fractional derivative for the heat conduction inverse problem based on the measurements of temperature in porous aluminum. *Int. J. Heat Mass Transf.* **2019**, *143*, 118440. [CrossRef]
10. Voller, V.R. Anomalous heat transfer: Examples, fundamentals, and fractional calculus models. *Adv. Heat Transf.* **2018**, *50*, 338–380.
11. Sierociuk, D.; Dzieliński, A.; Sarwas, G.; Petras, I.; Podlubny, I.; Skovranek, T. Modelling heat transfer in heterogeneous media using fractional calculus. *Philos. Trans. R. Soc.* **2013**, *371*, 20120146. [CrossRef]
12. Zhuag, Q.; Yu, B.; Jiang, X. An inverse problem of parameter estimation for time-fractional heat conduction in a composite medium using carbon–carbon experimental data. *Physica* **2015**, *456*, 9–15. [CrossRef]
13. Błasik M. A Numerical Method for the Solution of the Two-Phase Fractional Lamé-Clapeyron-Stefan Problem. *Mathematics* **2020**, *8*, 2157. [CrossRef]
14. Mozafarifard, M.; Toghraie, D.; Sobhani, H. Numerical study of fast transient non-diffusive heat conduction in a porous medium composed of solid-glass spheres and air using fractional Cattaneo subdiffusion model. *Int. Commun. Heat Mass Transf.* **2021**, *122*, 105192. [CrossRef]
15. Anderson, J.; Moradi, S.; Rafiq, T. Non-Linear Langevin and Fractional Fokker–Planck Equations for Anomalous Diffusion by Lévy Stable Processes. *Entropy* **2018**, *20*, 760. [CrossRef] [PubMed]
16. Youssri, Y.H.; Abd-Elhameed, W.M.; Mohamed, A.S.; Sayed, S.M. Generalized Lucas Polynomial Sequence Treatment of Fractional Pantograph Differential Equation. *Int. J. Appl. Comput. Math.* **2021**, *7*, 27. [CrossRef]
17. Abd-Elhameed, W.M.; Tenreiro Machado, J.A.; Youssri, Y.H. Hypergeometric fractional derivatives formula of shifted Chebyshev polynomials: Tau algorithm for a type of fractional delay differential equations. *Int. J. Nonlinear Sci. Numer. Simul.* **2021**. [CrossRef]
18. Atta, A.G.; Moatimid, G.M.; Youssri, Y.H. Generalized Fibonacci Operational tau Algorithm for Fractional Bagley-Torvik Equation. *Prog. Fract. Differ. Appl.* **2020**, *6*, 215–224.
19. Abd-Elhameed, W.M.; Youssri, Y.H. Fifth-kind orthonormal Chebyshev polynomial solutions for fractional differential equations. *Comput. Appl. Math.* **2018**, *37*, 2897–2921. [CrossRef]
20. Abd-Elhameed, W.M.; Youssri, Y.H. Generalized Lucas polynomial sequence approach for fractional differential equations. *Nonlinear Dyn.* **2017**, *89*, 1341–1355. [CrossRef]
21. Murio, D.A. Stable numerical solution of a fractional-diffusion inverse heat conduction problem. *Comput. Math. Appl.* **2007**, *53*, 1492–1501. [CrossRef]
22. Murio, D.A. Time fractional IHCP with Caputo fractional derivatives. *Comput. Math. Appl.* **2008**, *56*, 2371–2381. [CrossRef]
23. Zheng, G.H.; Wei, T. A new regularization method for the time fractional inverse advection-dispersion problem. *Siam J. Numer. Anal.* **2011**, *49*, 1972–1990. [CrossRef]
24. Zheng, G.H.; Wei, T. A new regularization method for solving a time-fractional inverse diffusion problem. *J. Math. Anal. Appl.* **2011**, *378*, 418–431. [CrossRef]

25. Dou, F.F.; Hon, Y.C. Kernel-based approximation for Cauchy problem of the time-fractional diffusion equation. *Eng. Anal. Bound. Elem.* **2012**, *36*, 1344–1352. [CrossRef]
26. Xiong, X.; Zhoua, Q.; Honb, Y.C. An inverse problem for fractional diffusion equation in 2-dimensional case: Stability analysis and regularization. *J. Math. Anal. Appl.* **2012**, *393*, 185–199. [CrossRef]
27. Yan, L.; Yang, F. Efficient Kansa-type MFS algorithm for time-fractional inverse diffusion problems. *Comput. Math. Appl.* **2014**, *67*, 1507–1520. [CrossRef]
28. Sakamoto, K.; Yamamoto, M. Initial value/boundary value problems for fractional diffusion-wave equations and applications to some inverse problems. *J. Math. Anal. Appl.* **2011**, *382*, 426–447. [CrossRef]
29. Wang, C.; Ling, L.; Xiong, X.; Li, M. Regularization for 2-D Fractional Sideways Heat Equations. *Numer. Heat Transf. Part Fundam.* **2015**, *68*, 418–433. [CrossRef]
30. Liu, S.; Feng, L. An Inverse Problem for a Two-Dimensional Time-Fractional Sideways Heat Equation. *Math. Probl. Eng.* **2020**, *2020*, 5865971. [CrossRef]
31. Kirane, M.; Malik, S.A.; Al-Gwaiz, M.A. An inverse source problem for a two dimensional time fractional diffusion equation with nonlocal boundary conditions. *Math. Methods Appl. Sci.* **2013**, *36*, 1056–1069. [CrossRef]
32. Shivanian, E.; Jafarabadi, A. The numerical solution for the time-fractional inverse problem of diffusion equation. *Eng. Anal. Bound. Elem.* **2018**, *91*, 50–59. [CrossRef]
33. Song, X.; Zheng, G.H.; Jiang, L. Identification of the reaction coefficient in time fractional diffusion equations. *J. Comput. Appl. Math.* **2019**, *345*, 295–309. [CrossRef]
34. Aldoghaither, A.; Laleg-Kirati, T.M. Parameter and differentiation order estimation for a two dimensional fractional partial differential equation. *J. Comput. Appl. Math.* **2020**, *369*, 112570. [CrossRef]
35. Djennadi, S.; Shawagfeh, N.; Arqub, O.A. Well-posedness of the inverse problem of time fractional heat equation in the sense of the Atangana–Baleanu fractional approach. *Alex. Eng. J.* **2020**, *59*, 2261–2268. [CrossRef]
36. Salman, A.; Malik, S.A. An inverse source problem for a two parameter anomalous diffusion equation with nonlocal boundary conditions. *Comput. Math. Appl.* **2017**, *73*, 2548–2560.
37. Moradi, S.; Anderson, J.; Romanelli, M.; Kim, H.-T. Global scaling of the heat transport in fusion plasmas. *Phys. Rev. Res.* **2020**, *2*, 013027. [CrossRef]
38. Yang, S.; Liu, F.; Feng, L.; Turner, I.W. Efficient numerical methods for the nonlinear two-sided space-fractional diffusion equation with variable coefficients. *Appl. Numer. Math.* **2020**, *157*, 55–68. [CrossRef]
39. Socha, K.; Dorigo, M. Ant colony optimization for continuous domains. *Eur. J. Oper. Res.* **2008**, *185*, 1155–1173. [CrossRef]
40. Brociek, R.; Chmielowska, A.; Słota, D. Comparison of the probabilistic ant colony optimization algorithm and some iteration method in application for solving the inverse problem on model with the Caputo type fractional derivative. *Entropy* **2020**, *22*, 555. [CrossRef]
41. Tian, W.Y.; Zhou, H.; Deng, W.H. A class of second order difference approximations for solving space fractional diffusion equations. *Math. Comput.* **2015**, *84*, 1703–1727. [CrossRef]
42. Barrett, R.; Berry, M.; Chan, T.F.; Demmel, J.; Donato, J.; Dongarra, J.; Eijkhout, V.; Pozo, R.; Romine, C.; der Vorst, H.V. *Templates for the Solution of Linear System: Building Blocks for Iterative Methods*; SIAM: Philadelphia, PA, USA, 1994.
43. Der Vorst, H.V. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *Siam J. Sci. Stat. Comput.* **1992**, *13*, 631–644. [CrossRef]
44. Brociek, R.; Słota, D. A method for solving the time fractional heat conduction inverse problem based on ant colony optimization and artificial bee colony algorithms. *Commun. Comput. Inf. Sci.* **2017**, *756*, 351–361. [CrossRef]