Journal of Mathematical Modeling
Vol. , No. , 2026, pp. -. Research Article

JMM

Parameters optimization for the fractional advection–dispersion equation using Crank–Nicolson and particle swarm optimization methods

Safar Irandoust-Pakchin[1,2], Golamreza Zaki[1], Mohammad Hossein Derakhshan[1*], Somaiyeh Abdi-Mazraeh[1]

[1] *Department of Applied Mathematics, Faculty of Mathematics, Statistics and Computer Sciences, University of Tabriz, Tabriz, Iran.*
[2] *Department of Mathematics, Firat University, 23119 Elazig, Turkiye.*

Email(s): s.irandoust@tabrizu.ac.ir, zaki@tabrizu.ac.ir, m.h.derakhshan.20@gmail.com, s.abdim@tabrizu.ac.ir

Abstract. In this paper, we consider the time–fractional advection–dispersion equation with the Caputo fractional derivative of order $0 < \alpha \leq 1$, dispersion coefficient $\lambda > 0$, and average fluid velocity $\mu \geq 0$. For the direct problem, the initial and boundary conditions are prescribed, and a numerical method based on the Crank–Nicolson scheme is employed. For the inverse problem, the goal is to identify the optimal values of λ, μ, and α. To achieve this, the particle swarm optimization method is used to minimize the fitting error so that the numerical solution matches the observed data at the selected time levels. Several numerical examples are presented to demonstrate the efficiency of the proposed approach and to validate the theoretical analysis.

Keywords: Crank–Nicolson method, time–fractional advection–dispersion equation, particle swarm optimization method, inverse initial value problem

AMS Subject Classification 2010: 26A33, 33E12, 34A08, 34K37, 35R11, 60G22

1 Introduction

In recent decades, fractional partial differential equations (FPDEs) have attracted significant attention from many researchers. Obtaining exact solutions for such equations is often extremely difficult or even impossible, and therefore numerical methods frequently provide the only feasible approach. Consequently, a wide range of numerical schemes has been developed to approximate solutions of these

*Corresponding author
Received: 30 January 2026/ Revised: 14 May 2026/ Accepted: 15 May 2026
DOI: 10.22124/jmm.2026.32871.2989

equations [4, 7, 10]. The time–fractional advection–dispersion equation, which is a well-known type of FPDE, has been extensively studied in the literature [7, 17]. In some works, the fractional derivative is applied solely with respect to the temporal variable [11, 15, 16], whereas in others it is taken with respect to the spatial variable [12, 20]. There also exist studies in which fractional differentiation is considered in both time and space variables simultaneously [5, 17].

The one-dimensional time-fractional advection-diffusion equation is commonly written as

$$ {}^{C}D_t^{\alpha} u(x,t) + v\frac{\partial u(x,t)}{\partial x} = D\frac{\partial^2 u(x,t)}{\partial x^2}, \qquad 0 < \alpha \leq 1, $$

where ${}^{C}D_t^{\alpha}$ denotes the Caputo fractional derivative with respect to time, v is the advection velocity, and D is the diffusion coefficient. Physically, this equation describes transport processes in media that possess memory. In classical transport, the time derivative of integer order implies that the system responds instantaneously to external forcing and depends only on its current state. In contrast, the fractional time derivative introduces a nonlocal dependence in time, meaning that the present evolution of the system depends on its entire history. This memory effect originates from microscopic mechanisms such as trapping, sticking, and waiting phenomena that delay the motion of particles. The advection term represents directed transport caused by an external flow field, such as groundwater flow carrying dissolved contaminants, air currents moving aerosols, or blood flow transporting biochemical species. The diffusion term represents random thermal motion of particles, which tends to spread the concentration from regions of high concentration to regions of low concentration. In homogeneous media, this spreading grows linearly in time; however, in heterogeneous or porous environments, the spreading is significantly slower. The fractional time derivative captures the physical reality of anomalous transport, particularly subdiffusion, where the mean squared displacement of particles scales as t^{α} with $0 < \alpha < 1$. This behavior reflects the fact that particles can remain immobilized for long, randomly distributed waiting times before moving again. As a result, even under a constant flow field, the effective drift of the particle cloud is slower than in classical advection–diffusion. At the microscopic level, this model is equivalent to a continuous-time random walk in which the spatial step lengths have a normal distribution, while the waiting times follow a heavy-tailed power-law distribution. The fractional derivative naturally emerges from the macroscopic limit of such stochastic processes. In practical applications, this equation is used to model contaminant transport in groundwater, solute migration in fractured rocks, ion transport in electrochemical systems, heat conduction in complex materials, diffusion of macromolecules in biological cells, and drug delivery in biological tissues. In these systems, experimental observations often show slower transport and long-memory effects that cannot be explained by classical models, but are well captured by the time–fractional advection–diffusion equation.

In recent years, considerable effort has been devoted to determine the appropriate order of the fractional derivative in modeling various phenomena [2, 3, 8, 14]. When all other parameters are fixed, different fractional orders lead to different numerical outcomes [22]. Therefore, an accurate determination of the fractional order is essential. Identifying the fractional order or other model parameters requires additional information about the underlying system. When such information is available, numerical solutions obtained for different parameter values are compared with the given data, and an error measure is computed. The appropriate parameter set is then determined by minimizing this error. Several studies have addressed the problem of parameter identification. For example, Joshi and coauthors proposed a novel approximation method for a two–dimensional phase transfer problem in a moving domain with a heat–generation parameter [9]. Abbas and collaborators investigated the influence of Marangoni convec-

tion and heat generation on Darcy–Forchheimer three–dimensional flow of a Maxwell fluid [1]. In [19], Pashapour and coauthors introduced a combined finite volume and physics–informed neural network approach for parameter estimation.

In this article, the fractional order of the derivative α is also considered as a parameter and the proposed algorithm has the ability to estimate three parameters dispersion coefficient λ, average fluid velocity μ and fractional order of the derivative α simultaneously. For success of the search operation, firstly for each parameter, searching interval is guessed. In other words, the search operation is performed only in those intervals. The search operation is done by using the evolutionary methods.

Particle Swarm Optimization (PSO) is a stochastic optimization technique inspired by the collective behavior of social animals, which was first proposed by Kennedy and Eberhart in 1995. The PSO algorithm mimics the cooperative and adaptive behavior observed in nature, such as the flocking of birds, schooling of fish, swarming of insects, or herding of mammals. In these natural systems, individuals adjust their movements based on their own experience as well as the experiences of other members of the group, enabling the swarm to efficiently locate resources such as food. Similarly, in PSO, a population of candidate solutions, called particles, explores the search space, continually updating their positions based on both personal and global best experiences. This cooperative learning process allows the swarm to converge toward optimal or near-optimal solutions over successive iterations [21]. Since its introduction, PSO has been widely applied to a variety of complex optimization problems across multiple domains. For instance, Dastranj employed PSO for the optimal design of a fractional-order PID controller, demonstrating its effectiveness in tuning controller parameters for improved system performance [6]. Zhang proposed an adaptive algorithm for fractional image enhancement that combines rough set theory with PSO to improve image quality and feature extraction [23]. Additionally, PSO has been successfully applied to solve inverse problems in electromagnetic devices, where the algorithm helps to determine unknown parameters from observed measurements, as reported in [13]. These diverse applications illustrate the versatility and robustness of PSO as a powerful optimization tool for both engineering and computational problems.

To solve the inverse problem, an efficient numerical procedure for the corresponding direct problem is first required. Once the direct solver is established, the unknown parameters can be identified by evaluating the fitting error for different parameter values. In this work, a Crank–Nicolson-type scheme is employed for the numerical solution of the direct problem, while the PSO method is utilized to solve the associated inverse optimization problem. The determination of the dispersion coefficient λ, the average fluid velocity μ, and the fractional order α constitutes a three-dimensional optimization problem. The objective is to minimize an error functional measuring the discrepancy between the computed and observed data. In general, this minimization problem may not admit a unique solution. Moreover, since the error functional is not available in explicit form, classical gradient-based optimization techniques, such as the steepest descent and quasi-Newton methods, are generally not applicable. The resulting inverse problem is highly nonlinear and computationally challenging. Evolutionary optimization techniques are therefore more suitable for this class of problems. Although stochastic optimization methods may produce slightly different results in repeated runs due to their random nature, reliable approximations can still be obtained by performing multiple simulations and selecting the best solution according to the error criterion.

The remainder of this paper is organized as follows. In the next section, the direct and inverse problems are formulated. Section 3 presents the discretization of the direct problem together with the derivation of the proposed numerical algorithm. Stability conditions and convergence analysis of the

method are established in Section 4. In Section 5, a PSO based approach is introduced for solving the inverse problem. Finally, several numerical examples are provided to demonstrate the efficiency and accuracy of the proposed method through comparisons with observed data.
We consider the following time–fractional advection–diffusion equation:

$$\begin{cases} \frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = \lambda \frac{\partial^2 u(x,t)}{\partial x^2} - \mu \frac{\partial u(x,t)}{\partial x} + f(x,t), \quad 0 \le t, \quad 0 \le x \le 1, \quad 0 < \alpha \le 1, \\ B.C.: u(0,t) = 0, \quad u(1,t) = 0, \quad 0 \le t \le 1, \\ I.C: u(x,0) = g(x), \quad 0 \le x \le 1, \end{cases} \tag{1.1}$$

where f and g are given functions, $\lambda > 0$ and $\mu \ge 0$ are real constants, and the fractional partial derivative is in the Caputo sense:

$$\frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = \begin{cases} \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{u_t(x,\tau)}{(t-\tau)^\alpha} d\tau, & 0 < \alpha < 1, \\ u_t(x,t), & \alpha = 1. \end{cases} \tag{1.2}$$

By setting $\mu = 0$ in Eq. (1.1), it can be seen that the heat equation is a special case of the advection-diffusion equation.
Suppose that the fractional order α and the parameters λ and μ, or some of them, are unknown. Instead, several additional conditions related to the problem are available. The objective is to determine appropriate values of λ, μ, and α such that the numerical solution obtained from the proposed numerical scheme is consistent with the given additional data. In other words, the aim is to identify the parameters λ, μ, and the fractional order α in the following problem:

$$\begin{cases} \frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = \lambda \frac{\partial^2 u(x,t)}{\partial x^2} - \mu \frac{\partial u(x,t)}{\partial x} + f(x,t), \quad 0 \le t, \quad 0 \le x \le 1, \quad 0 < \alpha \le 1, \\ B.C.: \quad u(0,t) = 0, \quad u(1,t) = 0, \quad 0 \le t \le 1, \\ I.C.: \quad u(x,0) = g_0(x), \\ A.C.: \quad u(x,\hat{t}_k) = h_k(x), \quad 0 \le x \le 1, \ \hat{t}_k \in \{t_1, t_2, ..., t_N\}, k = 1, 2, ..., s, \end{cases} \tag{1.3}$$

where $0 < t_1 < t_2 < ... < t_N = 1$ are the time steps.

Eq. (1.3) without the additional conditions (A.C.) represents the direct problem (1.1). When the parameters λ, μ, and α are known, the direct problem can be solved numerically, providing approximate values of $u(x,t)$ at the time levels corresponding to the additional conditions.

Let $\Lambda_{\alpha,\lambda,\mu}$ denote the numerical procedure used to solve problem (1.1) for given values of the parameters λ, μ, and α, and let $\tilde{u}_{\alpha,\lambda,\mu}$ represent the corresponding numerical approximation. It should be noted that $\tilde{u}_{\alpha,\lambda,\mu}(x,\hat{t}_k)$ is obtained as a discrete vector with M components, whereas the exact solution $u(x,\hat{t}_k) = h_k(x)$ is a continuous function. Consequently, in order to compare the numerical and exact solutions, an interpolation of the discrete values $\tilde{u}_{\alpha,\lambda,\mu}(x_i,\hat{t}_k)$, $i = 1,\ldots,M$ must be constructed. Using the L_2 norm, the discrepancy between these quantities can be expressed as

$$E(\alpha,\lambda,\mu) = \|\tilde{u}_{\alpha,\lambda,\mu} - h\|_{L_2} = \sum_{k=1}^{s} \int_0^1 \left|\tilde{u}_{\alpha,\lambda,\mu}(x,\hat{t}_k) - h_k(x)\right|^2 dx. \tag{1.4}$$

Alternatively, by discretizing h and employing the discrete L_2 norm, the error measure can be written as

$$e(\alpha,\lambda,\mu) = \|\tilde{u}_{\alpha,\lambda,\mu} - h\|_{l_2} = \sum_{k=1}^{s} \sum_{i=1}^{M} \left[\tilde{u}_{\alpha,\lambda,\mu}(x_i,\hat{t}_k) - h_k(x_i)\right]^2. \tag{1.5}$$

The determination of suitable values for the parameters α, λ, and μ is then formulated as the following constrained optimization problem:

$$\begin{cases} \min\limits_{\alpha,\lambda,\mu} \ e(\alpha,\lambda,\mu), \\ \text{s.t.} \quad 0 < \alpha_l \le \alpha \le \alpha_u \le 1, \\ \qquad\quad 0 < \lambda_l \le \lambda \le \lambda_u, \\ \qquad\quad 0 \le \mu_l \le \mu \le \mu_u, \end{cases} \tag{1.6}$$

where α_l, λ_l, and μ_l denote the prescribed lower bounds, and α_u, λ_u, and μ_u denote the prescribed upper bounds for the parameters α, λ, and μ, respectively. In this study, the constrained minimization problem (1.6) is solved using the PSO method.

2 Problem statement

In this section, we construct a higher-order numerical scheme for approximating the solution of the time–fractional advection-dispersion Eq. (1.1). To facilitate the discretization procedure, we introduce uniform partitions of the spatial and temporal domains as

$$\begin{cases} x_i = i\Delta_x, & \Delta_x = \frac{1}{M}, \quad i = 0,1,\ldots,M, \\ t_j = j\Delta_t, & \Delta_t = \frac{1}{N}, \quad j = 0,1,\ldots,N. \end{cases} \tag{2.1}$$

By employing the L_1 approximation, the Caputo fractional derivative of $u(x,t)$ with respect to t can be discretized as

$$\begin{aligned} \frac{\partial^\alpha u(x,t_j)}{\partial t^\alpha} &= \frac{1}{\Gamma(1-\alpha)} \int_0^{t_j} \frac{u_\tau(x,\tau)}{(t_j-\tau)^\alpha}\, d\tau \\ &= \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{j-1} \int_{k\Delta_t}^{(k+1)\Delta_t} \frac{u_\tau(x,\tau)}{(t_j-\tau)^\alpha}\, d\tau \\ &= \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{j-1} \frac{u(x,t_{k+1})-u(x,t_k)}{\Delta_t} \int_{k\Delta_t}^{(k+1)\Delta_t} (t_j-\tau)^{-\alpha}\, d\tau + T_{\Delta_t}^j \\ &= \frac{(\Delta_t)^{-\alpha}}{\Gamma(2-\alpha)} \sum_{k=0}^{j-1} \Big((j-k)^{1-\alpha} - (j-k-1)^{1-\alpha} \Big) \big(u(x,t_{k+1}) - u(x,t_k) \big) + T_{\Delta_t}^j \\ &= \sum_{k=0}^{j-1} b_k^j \big(u(x,t_{k+1}) - u(x,t_k) \big) + T_{\Delta_t}^j \\ &= -b_0^j u(x,t_0) + \sum_{k=1}^{j-1} (b_{k-1}^j - b_k^j) u(x,t_k) + b_{j-1}^j u(x,t_j) + T_{\Delta_t}^j, \end{aligned} \tag{2.2}$$

where

$$\sigma = \frac{(\Delta_t)^{-\alpha}}{\Gamma(2-\alpha)}, \qquad T_{\Delta_t}^j \le C_u \Delta_t^{2-\alpha},$$

$$b_k^j = \sigma\Big((j-k)^{1-\alpha} - (j-k-1)^{1-\alpha}\Big), \qquad k = 0,1,\ldots,j-1, \tag{2.3}$$

and C_u denotes a positive constant depending on the function u.

Using Eq. (1.5), Eq. (2.2) can be briefly written as

$$\begin{aligned}\frac{\partial^\alpha u(x,t_j)}{\partial t^\alpha} &= \sum_{k=0}^{j} d_k^j u(x,t_k) + T_{\Delta_t}^j \\ &\cong \sum_{k=0}^{j} d_k^j \tilde{u}^k(x) \\ &= \sum_{k=0}^{j} d_k^j \sum_{l=-2}^{M+2} c_l^k s_l(x),\end{aligned} \tag{2.4}$$

where

$$d_0^j = -b_0^j, \quad d_j^j = b_{j-1}^j = \sigma, \quad d_k^j = b_{k-1}^j - b_k^j, \quad k = 1,2,...,j-1. \tag{2.5}$$

Furthermore $\sum_{k=0}^{j} d_k^j = 0$ and $\sum_{k=0}^{j-1} d_k^j = \sigma$.

On the other hand, for space discretization, one can write

$$\lambda \tilde{u}_{xx}^j(x) - \mu \tilde{u}_x^j(x) = \sum_{l=-2}^{M+2} (c_l^j(\lambda s_l''(x) - \mu s_l'(x)). \tag{2.6}$$

Replacing Eqs. (2.4) and (2.6) in Eq. (1.1) gives

$$\sum_{k=0}^{j} d_k^j \sum_{l=-2}^{M+2} c_l^k s_l(x) = \sum_{l=-2}^{M+2} c_l^j(\lambda s_l''(x) - \mu s_l'(x)) + f^j(x). \tag{2.7}$$

According to the collocation technique, it can be resulted that

$$\sum_{k=0}^{j} d_k^j \sum_{l=-2}^{M+2} c_l^k s_l(x_i) = \sum_{l=-2}^{M+2} c_l^j(\lambda s_l''(x_i) - \mu s_l'(x_i)) + f^j(x_i). \tag{2.8}$$

Now, it can be derived that

$$\begin{aligned}&\sum_{k=0}^{j} d_k^j (c_{i-2}^k \frac{1}{120} + c_{i-1}^k \frac{26}{120} + c_i^k \frac{66}{120} + c_{i+1}^k \frac{26}{120} + c_{i+2}^k \frac{1}{120}) \\ &= c_{i-2}^j (\frac{\lambda}{6(\Delta_x)^2} - \frac{-\mu}{24\Delta_x}) + c_{i-1}^j (\frac{2\lambda}{6(\Delta_x)^2} - \frac{-10\mu}{24\Delta_x}) + c_i^j (\frac{-6\lambda}{6(\Delta_x)^2}) \\ &+ c_{i+1}^j (\frac{2\lambda}{6(\Delta_x)^2} - \frac{10\mu}{24\Delta_x}) + c_{i+2}^j (\frac{\lambda}{6(\Delta_x)^2} - \frac{\mu}{24\Delta_x}) + f_i^j.\end{aligned} \tag{2.9}$$

Assuming $\zeta_0 = \frac{1}{120}$, $\zeta_1 = \frac{1}{24\Delta_x}$, $\zeta_2 = \frac{1}{6(\Delta_x)^2}$, $\zeta_3 = \frac{1}{2(\Delta_x)^3}$, $d_j^j = \sigma$, one can write Eq. (2.9) as

$$c_{i-2}^j(\lambda\zeta_2 + \mu\zeta_1 - \sigma\zeta_0) + c_{i-1}^j(2\lambda\zeta_2 + 10\mu\zeta_1 - 26\sigma\zeta_0) + c_i^j(-6\lambda\zeta_2 - 66\sigma\zeta_0)$$

$$
\begin{aligned}
&+c_{i+1}^{j}(2\lambda\zeta_2-10\mu\zeta_1-26\sigma\zeta_0)+c_{i+2}^{j}(\lambda\zeta_2-\mu\zeta_1-\sigma\zeta_0)\\
&=\zeta_0\sum_{k=0}^{j-1}d_k^j(c_{i-2}^k+26c_{i-1}^k+66c_{i-2}^k+26c_{i-2}^k+c_{i-2}^k)-f_i^j.
\end{aligned}
\tag{2.10}
$$

To put it briefly, it can be said that

$$
a_1c_{i-2}^j+a_2c_{i-1}^j+a_3c_i^j+a_4c_{i+1}^j+a_5c_{i+2}^j=\zeta_0\sum_{k=0}^{j-1}d_k^j(c_{i-2}^k+26c_{i-1}^k+66c_i^k+26c_{i+1}^k+c_{i+2}^k)-f_i^j, \tag{2.11}
$$

where

$$
\begin{aligned}
a_1&=(\lambda\zeta_2+\mu\zeta_1-\sigma\zeta_0),\\
a_2&=(2\lambda\zeta_2+10\mu\zeta_1-26\sigma\zeta_0),\\
a_3&=(-6\lambda\zeta_2-66\sigma\zeta_0),\\
a_4&=(2\lambda\zeta_2-10\mu\zeta_1-26\sigma\zeta_0),\\
a_5&=(\lambda\zeta_2-\mu\zeta_1-\sigma\zeta_0).
\end{aligned}
$$

Considering the boundary conditions, one con derive that

$$
c_{-2}^j+26c_{-1}^j+66c_0^j+26c_1^j+c_2^j=0, \tag{2.12}
$$

$$
c_{M-2}^j+26c_{M-1}^j+66c_M^j+26c_{M+1}^j+c_{M+2}^j=0. \tag{2.13}
$$

From Eqs. (2.11) for $0\le i\le M$ and (2.11)-(2.13), a linear system of $M+3$ equations in $M+5$ unknowns is obtained. To solve this system uniquely, two auxiliary equations are needed. For this purpose, by taking the derivative of relation (2.7) with respect to the variable x, it can be written as

$$
\sum_{k=0}^{j}d_k^j\sum_{l=-2}^{M+2}c_l^k s_l'(x)=\sum_{l=-2}^{M+2}c_l^j(\lambda s_l'''(x)-\mu s_l''(x))+f'^j(x). \tag{2.14}
$$

Furthermore, for $i=0$, it can be resulted that

$$
\begin{aligned}
&\zeta_1\sum_{k=0}^{j-1}d_k^j(-c_{-2}^k-10c_{-1}^k+10c_1^k+c_2^k)\\
&=c_{-2}^j(-\lambda\zeta_3-\mu\zeta_2+\sigma\zeta_1)+c_{-1}^j(2\lambda\zeta_3-2\mu\zeta_2+10\sigma\zeta_1)+c_0^j(6\mu\zeta_2)\\
&\quad+c_1^j(-2\lambda\zeta_3-2\mu\zeta_2-10\sigma\zeta_1)+c_2^j(1\lambda\zeta_3-\mu\zeta_2-\sigma\zeta_1)+f'^j(0)
\end{aligned}
\tag{2.15}
$$

and for $i=M$, it can be shown that

$$
\begin{aligned}
&\zeta_1\sum_{k=0}^{j-1}d_k^j(-c_{M-2}^k-10c_{M-1}^k+10c_{M+1}^k+c_{M+2}^k)\\
&=c_{M-2}^j(-\lambda\zeta_3-\mu\zeta_2+\sigma\zeta_1)+c_{M-1}^j(2\lambda\zeta_3-2\mu\zeta_2+10\sigma\zeta_1)+c_M^j(6\mu\zeta_2)\\
&\quad+c_{M+1}^j(-2\lambda\zeta_3-2\mu\zeta_2-10\sigma\zeta_1)+c_{M+2}^j(1\lambda\zeta_3-\mu\zeta_2+\sigma\zeta_1)+f'^j(1).
\end{aligned}
\tag{2.16}
$$

Assuming that

$$\begin{aligned}
\hat{a}_1 &= (-\lambda\zeta_3 - \mu\zeta_2 + \sigma\zeta_1),\\
\hat{a}_2 &= (2\lambda\zeta_3 - 2\mu\zeta_2 + 10\sigma\zeta_1),\\
\hat{a}_3 &= (0\lambda\zeta_3 + 6\mu\zeta_2 - 0\sigma\zeta_1),\\
\hat{a}_4 &= (-2\lambda\zeta_3 - 2\mu\zeta_2 - 10\sigma\zeta_1),\\
\hat{a}_5 &= (1\lambda\zeta_3 - \mu\zeta_2 - \sigma\zeta_1),
\end{aligned}$$

Eqs. (2.15)-(2.16) can be written as

$$\begin{aligned}
&\hat{a}_1 c_{-2}^j + \hat{a}_2 c_{-1}^j + \hat{a}_3 c_0^j + \hat{a}_4 c_1^j + \hat{a}_5 c_2^j\\
&= \zeta_1 \sum_{k=0}^{j-1} d_k^j (-c_{-2}^k - 10c_{-1}^k + 10c_1^k + c_2^k) - f'^j(0), \qquad (2.17)\\
&\hat{a}_1 c_{M-2}^j + \hat{a}_2 c_{M-1}^j + \hat{a}_3 c_M^j + \hat{a}_4 c_{M+1}^j + \hat{a}_5 c_{M+2}^j\\
&= \zeta_1 \sum_{k=0}^{j-1} d_k^j (-c_{M-2}^k - 10c_{M-1}^k + 10c_{M+1}^k + c_{M+2}^k) - f'^j(1). \qquad (2.18)
\end{aligned}$$

From Eqs. (2.11) for $0 \le i \le M$, and (2.12)-(2.13) as well as Eqs. (2.17)-(2.18) a linear system of $M+5$ equations with $M+5$ unknowns $c_{-2}^j, c_{-1}^j, ..., c_{M+1}^j, c_{M+2}^j$ is obtained. In the matrix form this system can be written as

$$\begin{aligned}
AC^j &= \zeta_0 (d_{j-1}^j BC^{j-1} + d_{j-2}^j BC^{j-2} + ... + d_0^j BC^0) - F^j\\
&= \zeta_0 B (d_{j-1}^j C^{j-1} + d_{j-2}^j C^{j-2} + ... + d_0^j C^0) - F^j,
\end{aligned} \qquad (2.19)$$

or

$$C^j = A^{-1}(\zeta_0 B \sum_{k=0}^{j-1} d_k^j C^k - F^j), \qquad (2.20)$$

where

$$\begin{aligned}
C^j &= [c_{-2}^j, c_{-1}^j, c_0^j, ..., c_M^j, c_{M+1}^j, c_{M+2}^j]^T,\\
F^j &= [f(x_{-2}, t_j), f(x_{-1}, t_j), f(x_0, t_j), ..., f(x_M, t_j), f(x_{M+1}, t_j), f(x_{M+2}, t_j)]^T,
\end{aligned}$$

as well as A and B are the quasi five-diagonal $(M+5)\times(M+5)$ matrices as follows:

$$A = \begin{bmatrix}
1 & 26 & 66 & 26 & 1 & 0 & \cdots & 0\\
\hat{a}_1 & \hat{a}_2 & \hat{a}_3 & \hat{a}_4 & \hat{a}_5 & 0 & \cdots & 0\\
a_1 & a_2 & a_3 & a_4 & a_5 & 0 & \cdots & 0\\
0 & a_1 & a_2 & a_3 & a_4 & a_5 & \cdots & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots\\
0 & \cdots & a_1 & a_2 & a_3 & a_4 & a_5 & 0\\
0 & \cdots & 0 & a_1 & a_2 & a_3 & a_4 & a_5\\
0 & \cdots & 0 & \hat{a}_1 & \hat{a}_2 & \hat{a}_3 & \hat{a}_4 & \hat{a}_5\\
0 & \cdots & 0 & 1 & 26 & 66 & 26 & 1
\end{bmatrix},$$

$$B=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -\zeta_1/\zeta_0 & -10\zeta_1/\zeta_0 & 0 & 10\zeta_1/\zeta_0 & \zeta_1/\zeta_0 & 0 & \cdots & 0 \\ 1 & 26 & 66 & 26 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 26 & 66 & 26 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 26 & 66 & 26 & 1 & 0 \\ 0 & \cdots & 0 & 1 & 26 & 66 & 26 & 1 \\ 0 & \cdots & 0 & -\zeta_1/\zeta_0 & -10\zeta_1/\zeta_0 & 0 & 10\zeta_1/\zeta_0 & \zeta_1/\zeta_0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

It is worth mentioning that to obtain the $C^0=[c^0_{-2},c^0_{-1},c^0_0,...,c^0_M,c^0_{M+1},c^0_{M+2}]^T$, the initial condition $g(x)=\sum_{k=-2}^{M+2}c^0_k s_k(x)$ is used, and the following relation is derived:

$$C^0=A_0^{-1}G_0, \tag{2.21}$$

where

$$A_0=\begin{bmatrix} -1 & -10 & 0 & 10 & 1 & 0 & \cdots & 0 \\ 1 & 2 & -6 & 2 & 1 & 0 & \cdots & 0 \\ 1 & 26 & 66 & 26 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 26 & 66 & 26 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 26 & 66 & 26 & 1 & 0 \\ 0 & \cdots & 0 & 1 & 26 & 66 & 26 & 1 \\ 0 & \cdots & 0 & 1 & 2 & -6 & 2 & 1 \\ 0 & \cdots & 0 & -1 & -10 & 0 & 10 & 1 \end{bmatrix},$$

$$G^0=[\zeta_2 g''(x_0),\zeta_1 g'(x_0),\zeta_0 g(x_0),...,\zeta_0 g(x_M),\zeta_1 g'(x_0),\zeta_2 g''(x_M)]^T.$$

3 Direct problem and discretization

The Crank-Nicolson type method for solving problem (1.1) in the special case $\lambda=1$ and $\mu=0$ was introduced in [11], where it was shown that the scheme is stable for $\alpha\geq 1-\ln(2)/\ln(3)\approx 0.3691$. Moreover, the method achieves a convergence of $O\left(\Delta_t^{2-\alpha}+\Delta_x^2\right)$. In the present work, we extend this method to a more general setting $\lambda>0$ and $\mu\geq 0$, and we introduce a Crank-Nicolson type scheme for solving problem (1.1) under these broader conditions. For the discretization, we employ the uniform spatial and temporal grids defined by

$$\begin{cases} x_i=i\Delta_x, & i=0,1,\ldots,M, \quad \Delta_x=1/M, \\ t_j=j\Delta_t, & j=0,1,\ldots,N, \quad \Delta_t=1/N, \end{cases} \tag{3.1}$$

where Δ_x and Δ_t denote the spatial and temporal step sizes, respectively. Following [11], the Caputo fractional derivative with respect to time is discretized at the midpoint $t_{j+1/2}=t_j+\Delta_t/2$ as

$$\frac{\partial^\alpha u(x_i,t_{j+1/2})}{\partial t^\alpha}=\frac{1}{\Gamma(1-\alpha)}\int_0^{t_{j+1/2}} u_t(x_i,\tau)(t_{j+1/2}-\tau)^{-\alpha}\,d\tau$$

$$= \frac{1}{\Gamma(1-\alpha)} \left[\int_0^{t_j} u_t(x_i,\tau)(t_{j+1/2}-\tau)^{-\alpha} d\tau + \int_{t_j}^{t_{j+1/2}} u_t(x_i,\tau)(t_{j+1/2}-\tau)^{-\alpha} d\tau \right] \tag{3.2}$$

$$= \frac{1}{\Gamma(1-\alpha)} \left[\sum_{k=1}^{j} \int_{t_{k-1}}^{t_k} u_t(x_i,\tau)\left((j+\tfrac{1}{2})\Delta_t-\tau\right)^{-\alpha} d\tau + \int_{t_j}^{t_{j+1/2}} u_t(x_i,\tau)\left((j+\tfrac{1}{2})\Delta_t-\tau\right)^{-\alpha} d\tau \right].$$

If one use the approximation formula

$$u_t(x_i,\tau) = \frac{u(x_i,t_{j+1}) - u(x_i,t_j)}{\Delta_t} + O(\Delta_t),$$

then Eq. (3.2) is written as follows:

$$\frac{\partial^\alpha u(x_i,t_{j+1/2})}{\partial t^\alpha} = \frac{{\Delta_t}^{-\alpha}}{\Gamma(2-\alpha)} \left\{ \frac{\mathrm{u_{i,j}} - \mathrm{u_{i,j-1}}}{2^{1-\alpha}} + \sum_{k=1}^{\mathrm{j}} (\mathrm{u_{i,k}} - \mathrm{u_{i,k-1}})\,\gamma_k \right\} + \mathrm{R_1} + \mathrm{R_2}, \tag{3.3}$$

where

$$\gamma_k = \left(j-k+\tfrac{3}{2}\right)^{1-\alpha} - \left(j-k+\tfrac{1}{2}\right)^{1-\alpha},$$
$$R_1 = \tfrac{1}{\Gamma(1-\alpha)} \textstyle\sum_{k=1}^{j} \int_{(k-1)\Delta_t}^{k\Delta_t} (\tau - t_{k-1/2})\, u_{tt}(x_i,\xi_k) \left((j+1/2)\Delta_t - \tau\right)^{-\alpha} d\tau,$$
$$R_2 = \tfrac{1}{\Gamma(2-\alpha)} \tfrac{1}{2^{1-\alpha}} O\left({\Delta_t}^{2-\alpha}\right),$$

and we see that

$$R_1 \le \frac{{\Delta_t}^{2-\alpha}}{\Gamma(1-\alpha)} \max_{1\le i\le N} |u_{tt}(x_i,\xi_k)|.$$

If we use the following notation

$$\sigma = \tfrac{\Delta_t^{-\alpha}}{\Gamma(2-\alpha)}, \quad \omega_j = \sigma\left[(j+1/2)^{1-\alpha} - (j-1/2)^{1-\alpha}\right],$$

then we can rewrite Eq. (3.3) as

$$\begin{aligned} \frac{\partial^\alpha u(x_i,t_{j+1/2})}{\partial t^\alpha} = {} & \omega_1 u_{i,j} + \sum_{k=1}^{\mathrm{j-1}} (\omega_{\mathrm{j-k+1}} - \omega_{\mathrm{j-k}})\, u_{i,k} - \omega_{\mathrm{j}} u_{i,0} \\ & + \sigma \frac{u_{i,j+1} - u_{i,j}}{2^{1-\alpha}} + \mathrm{O}\left(\Delta_{\mathrm{t}}^{2-\alpha}\right). \end{aligned} \tag{3.4}$$

On the other hand

$$\begin{aligned} u_x(x_i,t_{j+1/2}) = {} & \frac{1}{(4\Delta_x)} \left\{ u_{i+1,j+1} - u_{i-1,j+1} \right\} \\ & + \frac{1}{(4\Delta_x)} \left\{ u_{i+1,j} - u_{i-1,j} \right\} + \mathrm{O}\left(\Delta_x^2\right) \end{aligned} \tag{3.5}$$

and

$$\begin{aligned} u_{xx}(x_i,t_{j+1/2}) = {} & \frac{1}{2(\Delta_x)^2} \left\{ u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1} \right\} \\ & + \frac{1}{2(\Delta_x)^2} \left\{ u_{i-1,j} - 2u_{i,j} + u_{i+1,j} \right\} + \mathrm{O}\left(\Delta_{\mathrm{x}}^2\right). \end{aligned} \tag{3.6}$$

By substituting Eqs. (3.4), (3.5), and (3.6) into Eq. (1.1), we obtain

$$\omega_1 \tilde{u}_{i,j} + \sum_{k=1}^{j-1} (\omega_{j-k+1} - \omega_{j-k}) \tilde{u}_{i,k} - \omega_j \tilde{u}_{i,0} + \sigma \frac{\tilde{u}_{i,j+1} - \tilde{u}_{i,j}}{2^{1-\alpha}} = \frac{\lambda}{2(\Delta_x)^2} (\tilde{u}_{i-1,j+1} - 2\tilde{u}_{i,j+1} + \tilde{u}_{i+1,j+1})$$
$$+ \frac{\lambda}{2(\Delta_x)^2} (\tilde{u}_{i-1,j} - 2\tilde{u}_{i,j} + \tilde{u}_{i+1,j}) - \frac{\mu}{4\Delta_x} (\tilde{u}_{i+1,j+1} - \tilde{u}_{i-1,j+1}) - \frac{\mu}{4\Delta_x} (\tilde{u}_{i+1,j} - \tilde{u}_{i-1,j}) + f(x_i, t_{j+1/2}). \quad (3.7)$$

Setting

$$r = \frac{\lambda}{2(\Delta_x)^2}, \qquad s = \frac{\mu}{4\Delta_x}, \qquad a = \frac{\sigma}{2^{1-\alpha}}, \quad (3.8)$$

the above system can be rearranged as

$$\begin{aligned}(-r-s)\tilde{u}_{i-1,j+1} + (a+2r)\tilde{u}_{i,j+1} + (-r+s)\tilde{u}_{i+1,j+1} = (r+s)\tilde{u}_{i-1,j} + (-\omega_1 + a - 2r)\tilde{u}_{i,j} \\ + (r-s)\tilde{u}_{i+1,j} - \sum_{k=1}^{j-1} (\omega_{j-k+1} - \omega_{j-k})\tilde{u}_{i,k} \\ - \omega_j \tilde{u}_{i,0} + f(x_i, t_{j+1/2}).\end{aligned} \quad (3.9)$$

In matrix form, this equation can be written as

$$A\tilde{U}^{j+1} = B\tilde{U}^{j} - \sum_{l=0}^{j-1} d_l^j \tilde{U}^{l} + F^{j+1/2}, \quad (3.10)$$

where

$$\begin{aligned}\tilde{U}^{j} &= \begin{bmatrix}\tilde{u}_{1,j} & \tilde{u}_{2,j} & \cdots & \tilde{u}_{n-1,j}\end{bmatrix}^T, \\ F^{j+1/2} &= \begin{bmatrix}f_{1,j+1/2} & f_{2,j+1/2} & \cdots & f_{n-1,j+1/2}\end{bmatrix}^T, \\ A &= aI + rT_1 + sT_2, \\ B &= (-\omega_1 + a)I - rT_1 - sT_2 = (-\omega_1 + 2a)I - A, \\ d_0^j &= -\omega_j, \qquad d_l^j = \omega_{j-l+1} - \omega_{j-l}, \quad l = 1, 2, \ldots, j.\end{aligned} \quad (3.11)$$

Here, I denotes the $N \times N$ identity matrix. The matrix T_1 is an $N \times N$ tridiagonal matrix whose diagonal entries are equal to 2, and whose super- and sub-diagonal entries are equal to -1, that is,

$$T_1 = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}.$$

Similarly, the matrix T_2 is an $N \times N$ tridiagonal matrix having zero diagonal entries, sub-diagonal entries equal to -1, and super-diagonal entries equal to 1:

$$T_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 0 & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix}.$$

Proposition 1. *If $\lambda > (\Delta_x \mu)/2$, then the matrix A is strictly diagonally dominant, and consequently, A is nonsingular.*

Proof. Assume that $r > s$. Then

$$|-r-s|+|-r+s| = r+s+|-r+s| = r+s+(r-s) = 2r < 2r+a = |2r+a|.$$

Thus, the diagonal entry of A strictly dominates the sum of the magnitudes of the off-diagonal entries, which confirms strict diagonal dominance. □

Proposition 1 guarantees the solvability of Eq. (3.10). Therefore, if $\lambda > (\Delta_x \mu)/2$, then Eq. (3.10) can be rewritten as

$$\tilde{U}^{j+1} = A^{-1}\left(B\tilde{U}^{j} - \sum_{l=0}^{j-1} d_l^j \tilde{U}^{l} + F^{j+1/2} \right). \tag{3.12}$$

It should be observed that the values $\tilde{u}_{i,0}$ are determined directly from the initial condition of the problem. Equivalently, the vector $\tilde{U}^0$ is known. Hence, when $\lambda > (\Delta_x \mu)/2$, the recursive relation (3.12) is well defined for all time steps.

4 Stability and convergence

We analyze the stability of the difference scheme by applying the Fourier method. Let $\tilde{u}_{i,j}$ denote the numerical approximation, and define the error $\rho_{i,j} = \tilde{u}_{i,j} - u_{i,j}$ for $i = 1,2,\ldots,M$ and $j = 1,2,\ldots,N$. Then we obtain the following roundoff-error equation:

$$\begin{aligned}(-r-s)\rho_{i-1,j+1} + (a+2r)\rho_{i,j+1} + (-r+s)\rho_{i+1,j+1} = {} & (r+s)\rho_{i-1,j} + (-\omega_1 + a - 2r)\rho_{i,j} + (r-s)\rho_{i+1,j} \\ & - \sum_{k=1}^{j-1} \left(\omega_{j-k+1} - \omega_{j-k}\right) \rho_{i,k} \\ & - \omega_j \rho_{i,0} + f(x_i, t_{j+1/2}).\end{aligned} \tag{4.1}$$

We now define the grid function

$$\rho^{j}(x) = \begin{cases} \rho_{i,j}, & x_i - \Delta_x/2 < x < x_i + \Delta_x/2, \\ 0, & 0 \le x < \Delta_x/2 \ \text{ or } \ 1 - \Delta_x/2 < x \le 1. \end{cases} \tag{4.2}$$

The function $\rho^j(x)$ admits the Fourier series representation

$$\rho^j(x) = \sum_{l=-\infty}^{\infty} d_j(l)\, e^{\mathrm{i}2\pi l x}, \tag{4.3}$$

where

$$d_j(l) = \int_0^1 \rho^j(x)\, e^{-\mathrm{i}2\pi l x}\, dx, \qquad j = 1,2,\ldots,N,$$

and $\mathrm{i} = \sqrt{-1}$ denotes the imaginary unit. Based on this representation, we consider a mode of the form

$$\rho_{i,j} = d_j\, e^{\mathrm{i}i\Delta_x\beta}, \qquad \beta = 2\pi l.$$

Substituting this expression into Eq. (4.1), we obtain

$$\begin{aligned}
&(-r-s)d_{j+1}e^{\mathrm{i}(i-1)\Delta_x\beta} + (a+2r)d_{j+1}e^{\mathrm{i}i\Delta_x\beta} + (-r+s)d_{j+1}e^{\mathrm{i}(i+1)\Delta_x\beta} \\
&= (r+s)d_j e^{\mathrm{i}(i-1)\Delta_x\beta} + (-\omega_1 + a - 2r)d_j e^{\mathrm{i}i\Delta_x\beta} + (r-s)d_j e^{\mathrm{i}(i+1)\Delta_x\beta} \\
&\quad - \sum_{k=1}^{j-1} \left(\omega_{j-k+1} - \omega_{j-k}\right) d_k\, e^{\mathrm{i}i\Delta_x\beta} - \omega_j d_0\, e^{\mathrm{i}i\Delta_x\beta}.
\end{aligned} \tag{4.4}$$

After simplification, we obtain

$$\begin{aligned}
d_{j+1}\Big(-r\left[e^{\mathrm{i}\Delta_x\beta} + e^{-\mathrm{i}\Delta_x\beta}\right] + s\left[e^{\mathrm{i}\Delta_x\beta} - e^{-\mathrm{i}\Delta_x\beta}\right] &+ a + 2r\Big) = d_j\Big(r\left[e^{\mathrm{i}\Delta_x\beta} + e^{-\mathrm{i}\Delta_x\beta}\right] \\
&+ s\left[e^{\mathrm{i}\Delta_x\beta} - e^{-\mathrm{i}\Delta_x\beta}\right] + (-\omega_1 + a - 2r)\Big) \\
&+ \sum_{m=1}^{j-1} (\omega_{j-m+1} - \omega_{j-m}) d_m - \omega_j d_0.
\end{aligned} \tag{4.5}$$

Finally, using the identities

$$e^{\mathrm{i}\theta} + e^{-\mathrm{i}\theta} = 2\cos\theta, \qquad e^{\mathrm{i}\theta} - e^{-\mathrm{i}\theta} = 2\mathrm{i}\sin\theta,$$

we arrive at

$$\begin{aligned}
d_{j+1}\left(-2r\cos(\Delta_x\beta) + 2s\mathrm{i}\sin(\Delta_x\beta) + a + 2r\right) &= d_j\left(2r\cos(\Delta_x\beta) - 2s\mathrm{i}\sin(\Delta_x\beta) + (-\omega_1 + a - 2r)\right) \\
&+ \sum_{m=1}^{j-1} \left(\omega_{j-m+1} - \omega_{j-m}\right) d_m - \omega_j d_0.
\end{aligned} \tag{4.6}$$

Proposition 2. *If $3^\alpha > \frac{3}{2}$, then the sequence $\{d_j\}$ satisfies $|d_j| \le |d_0|$ for all $j = 1,2,\ldots,N$, where d_j denotes the Fourier coefficient appearing in the solution of Eq. (4.6).*

Proof. We use mathematical induction. For $j=0$, we have

$$|d_1| = \left|\frac{2r\cos(\Delta_x\beta) - 2s\sin(\Delta_x\beta) + (-\omega_1 + a - 2r)}{-2r\cos(\Delta_x\beta) + 2s\sin(\Delta_x\beta) + (a+2r)}\right| |d_0|$$
$$= \left|\frac{-2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) + a}{2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a}\right| |d_0| \le |d_0|.$$

Now assume that

$$|d_n| \le |d_0|, \qquad n = 1,2,\ldots,j.$$

We prove the inequality for d_{j+1}. From Eq. (4.6) we have

$$|d_{j+1}| \le \left(\frac{\left|-2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) + (-\omega_1 + a)\right|}{2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a}\right) |d_0|$$
$$+ \left(\frac{\omega_1}{2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a}\right) |d_0|. \tag{4.7}$$

If

$$-2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) + (-\omega_1 + a) \ge 0,$$

then

$$|d_{j+1}| \le \frac{|-2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) + a|}{2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a} |d_0| \le |d_0|. \tag{4.8}$$

If instead this expression is negative, then

$$|d_{j+1}| \le \frac{2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + (2\omega_1 - a)}{2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a} |d_0|. \tag{4.9}$$

This yields $|d_{j+1}| \le |d_0|$ provided that $\omega_1 \le a$. □

Theorem 1. *The finite difference scheme* (3.10) *is stable whenever* $3^{1-\alpha} < 2$.

Proof. Assume that $3^{1-\alpha} < 2$. Then, by Proposition 2 and the Parseval identity, it follows that

$$\|\rho^j\|_2 \le \|\rho^0\|_2, \qquad j = 1,2,\ldots,N,$$

which shows that the proposed difference scheme is conditionally stable. □

Let the truncation error at $(x_i, t_{j+1/2})$ be denoted by R_i^j. From Eqs. (3.4), (3.5), and (3.6), it follows that

$$|R_i^j| \le C_{i,j}^1 \left(\Delta_t^{2-\alpha} + \Delta_x^2\right), \qquad \text{or simply} \qquad |R_i^j| \le C_1 \left(\Delta_t^{2-\alpha} + \Delta_x^2\right),$$

where $C_1 = \max_{i,j} C_{i,j}^1$.

The error equation may thus be written as

$$(-r-s)\rho_{i-1,j+1} + (a+2r)\rho_{i,j+1} + (-r+s)\rho_{i+1,j+1}$$
$$-(r+s)\rho_{i-1,j} - (-\omega_1 + a - 2r)\rho_{i,j} - (r-s)\rho_{i+1,j}$$
$$-\sum_{k=1}^{j-1}(\omega_{j-k+1} - \omega_{j-k})\rho_{i,k} + \omega_j\rho_{i,0} = R_i^j. \tag{4.10}$$

Define the grid function

$$R^j(x) = \begin{cases} R_i^j, & x_i - \frac{\Delta_x}{2} < x < x_i + \frac{\Delta_x}{2}, \\ 0, & 0 \le x < \frac{\Delta_x}{2} \text{ or } 1 - \frac{\Delta_x}{2} < x \le 1. \end{cases} \tag{4.11}$$

The function $R^j(x)$ admits the Fourier series expansion

$$R^j(x) = \sum_{l=-\infty}^{\infty} \xi_j(l)\, e^{\mathrm{i}2\pi l x}, \tag{4.12}$$

where

$$\xi_j(l) = \int_0^1 R^j(x)\, e^{-\mathrm{i}2\pi l x}\, dx, \qquad j = 1, 2, \ldots, N.$$

Substituting the Fourier mode

$$\rho_{i,j} = \xi^j e^{\mathrm{i}i\Delta_x\beta}, \qquad \beta = 2\pi l,$$

into Eq. (4.10), we obtain

$$\begin{aligned} &(-r-s)\xi^{j+1}e^{\mathrm{i}(i-1)\Delta_x\beta} + (a+2r)\xi^{j+1}e^{\mathrm{i}i\Delta_x\beta} + (-r+s)\xi^{j+1}e^{\mathrm{i}(i+1)\Delta_x\beta} \\ &= (r+s)\xi^j e^{\mathrm{i}(i-1)\Delta_x\beta} + (-\omega_1 + a - 2r)\xi^j e^{\mathrm{i}i\Delta_x\beta} + (r-s)\xi^j e^{\mathrm{i}(i+1)\Delta_x\beta} \\ &\quad + \sum_{k=1}^{j-1} (\omega_{j-k+1} - \omega_{j-k})\, \xi^k e^{\mathrm{i}i\Delta_x\beta} - \omega_j \xi^0 e^{\mathrm{i}i\Delta_x\beta} + R_i^j. \end{aligned} \tag{4.13}$$

After cancelling the common factor $e^{\mathrm{i}i\Delta_x\beta}$, we obtain

$$\begin{aligned} &(-r-s)\xi^{j+1}e^{-\mathrm{i}\Delta_x\beta} + (a+2r)\xi^{j+1} + (-r+s)\xi^{j+1}e^{\mathrm{i}\Delta_x\beta} \\ &= (r+s)\xi^j e^{-\mathrm{i}\Delta_x\beta} + (-\omega_1 + a - 2r)\xi^j + (r-s)\xi^j e^{\mathrm{i}\Delta_x\beta} \\ &\quad + \sum_{k=1}^{j-1} (\omega_{j-k+1} - \omega_{j-k})\xi^k - \omega_j \xi^0 + R_i^j. \end{aligned} \tag{4.14}$$

Using

$$e^{\mathrm{i}\theta} + e^{-\mathrm{i}\theta} = 2\cos\theta, \qquad e^{\mathrm{i}\theta} - e^{-\mathrm{i}\theta} = 2\mathrm{i}\sin\theta,$$

Eq. (4.14) becomes

$$\begin{aligned} \xi^{j+1}\big(-2r\cos(\Delta_x\beta) + 2s\mathrm{i}\sin(\Delta_x\beta) + a + 2r\big) &= \xi^j\big(2r\cos(\Delta_x\beta) - 2s\mathrm{i}\sin(\Delta_x\beta) - \omega_1 + a - 2r\big) \\ &\quad + \sum_{k=1}^{j-1} (\omega_{j-k+1} - \omega_{j-k})\, \xi^k - \omega_j \xi^0 + R_i^j. \end{aligned} \tag{4.15}$$

Rearranging, we obtain the recurrence relation

$$\xi^{j+1} = \frac{2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) - \omega_1 + a}{-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a}\, \xi^j + \frac{\sum_{k=1}^{j-1} (\omega_{j-k+1} - \omega_{j-k})\xi^k - \omega_1 \xi^j + \eta}{-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a}, \tag{4.16}$$

where η denotes the Fourier coefficient of the truncation error.

Proposition 3. *Let ξ^j be the solution of the recurrence relation (4.16). Then there exists a constant $C_2 > 0$ such that*

$$|\xi^j| \le C_2|\eta_1|, \qquad j = 1,2,\dots,N.$$

Proof. From Eq. (4.16),

$$\begin{aligned}|\xi^{j+1}| \le{}& \frac{|2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) - \omega_1 + a|}{|-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a|}|\xi^j| \\ &+ \frac{1}{|-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a|}\left(\left|\sum_{k=1}^{j-1}(\omega_{j-k+1} - \omega_{j-k})\right||\xi^k| + |\eta|\right).\end{aligned} \tag{4.17}$$

Using the induction hypothesis $|\xi^k| \le C_2|\eta_1|$, we obtain

$$|\xi^{j+1}| \le \left(\frac{|2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) - \omega_1 + a|}{|-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a|} + \frac{\omega_1 - \omega_j + 1}{|-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a|}\right)C_2|\eta_1|.$$

If

$$2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) - \omega_1 + a \ge 0,$$

then

$$\frac{2r(1-\cos(\Delta_x\beta)) - 2s\sin(\Delta_x\beta) - \omega_j + a + 1}{-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a} \le 1,$$

and therefore $|\xi^{j+1}| \le C_2|\eta_1|$. In the opposite case, we obtain

$$|\xi^{j+1}| \le \frac{-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + 2\omega_1 - \omega_j - a + 1}{-2r(1-\cos(\Delta_x\beta)) + 2s\sin(\Delta_x\beta) + a}C_2|\eta_1,$$

and the numerator does not exceed the denominator provided that

$$3^\alpha > \frac{3}{2},$$

which completes the proof. □

5 Inverse problem

To solve the optimization problem (1.6), we first compute the numerical solution of Eq. (1.1) by the proposed numerical method. Denote this approximation at the observation time $t = \hat{t}$ by

$$\tilde{u}_{\lambda,\mu,\alpha}(x,\hat{t}),$$

where $\hat{t}$ corresponds to the time level associated with the additional condition. Using this numerical solution, we evaluate the error functional defined in Eq. (1.5). If the resulting error is not sufficiently small, the parameters λ, μ, and α are updated by means of the PSO algorithm.

In the first iteration of PSO algorithm, each particle is assigned random initial values of the parameters λ, μ, and α, and Eq. (1.1) is solved for each such parameter set. The corresponding value of the objective functional in (1.6) is then computed. Among all particles, the one yielding the smallest error

determines the best global position of the swarm, denoted by G_{best}. This quantity is subsequently used to update the position and velocity of each particle. In later iterations, the update process depends not only on G_{best}, but also on the best position previously visited by each particle, denoted by P_{best}.

Let $(\lambda_i^j, \mu_i^j, \alpha_i^j)$ denote the position of the ith particle at the jth iteration, and let e_i^j be the corresponding value of the objective functional. If the swarm size is denoted by $\mathcal{N}_{\text{pop}}$, then the global best position at iteration j is defined by

$$G_{\text{best}}^j = \underset{1\le k\le \mathcal{N}_{\text{pop}}}{\arg\min}\ e_k^j = (\lambda^{j^*}, \mu^{j^*}, \alpha^{j^*}).$$

Similarly, the personal best position of the ith particle up to iteration j is defined by

$$P_{\text{best}}^i = \underset{1\le k\le j}{\arg\min}\, e_i^k = (\lambda_{i^*}, \mu_{i^*}, \alpha_{i^*}).$$

The position of the ith particle is then updated according to

$$(\lambda_i^{j+1}, \mu_i^{j+1}, \alpha_i^{j+1}) = (\lambda_i^j, \mu_i^j, \alpha_i^j) + v_i^{j+1},$$

where v_i^{j+1} denotes the updated velocity of the ith particle. This velocity is computed by

$$v_i^{j+1} = v_i^j + c_1\,\text{rand}\left(P_{\text{best}}^i - (\lambda_i^j, \mu_i^j, \alpha_i^j)\right) + c_2\,\text{rand}(t)\left(G_{\text{best}}^j - (\lambda_i^j, \mu_i^j, \alpha_i^j)\right).$$

In the numerical experiments, the swarm size is taken to be 30, and the maximum number of iterations is denoted by Iter. For further details on the PSO algorithm, we refer the reader to [18]. The steps of the proposed algorithm are summarized in Algorithm 1.

Algorithm 1: PSO algorithm for solving the inverse problem

Input:
  (i) Specify the functions $f(x,t)$, $g(x)$, and $h(x)$.
  (ii) Choose admissible intervals for the parameters λ, μ, and α.
  (iii) Set the number of unknown parameters equal to 3 and the swarm size equal to 30.
Initialize the particles randomly within the admissible intervals.
for each particle in the swarm **do**
  Solve Eq. (1.1).
  Evaluate the objective functional defined in Eq. (1.6).
end for
Compute the personal best position P_{best}^i for each particle.
Determine the global best position G_{best}.
Update the velocity and position of each particle according to the PSO rules.
if the stopping criterion is not satisfied **then**
  Go to the evaluation step.
end if
Output: Report the particle corresponding to the minimum value of the objective functional.

6 Numerical results

In this section, we present several numerical experiments to assess the accuracy and robustness of the proposed algorithm in solving the inverse time–fractional advection–dispersion equation. When the parameters λ, μ, and α are prescribed, the governing equation together with the initial and boundary conditions can be solved as a direct problem using the numerical method developed in the previous sections. In contrast, the inverse problem consists of determining the parameters λ, μ, and α such that the numerical solution of the direct problem best matches the given additional data. To minimize the resulting mismatch error, we employ PSO algorithm.

In all direct simulations, the spatial and temporal step sizes are set to $\Delta x = 0.005$ and $\Delta t = 0.005$, respectively. In the inverse problem, we use the default settings of the PSO algorithm, including a maximum of $\texttt{MaxIter} = 600$ iterations and $\texttt{MaxTime} = \infty$.

Example 1. Consider the following inverse fractional advection–dispersion problem:

$$\begin{cases} \partial_t^{\alpha} u = \lambda u_{xx} - \mu u_x + \dfrac{6}{\Gamma(3.3)} t^{2.3} x(1-x) + 10(1+t^3) + 3(1-2x)(1+t^3), \\ \text{I.C.}: \quad u(x,0) = x(1-x), \\ \text{B.C.}: \quad u(0,t) = u(1,t) = 0, \\ \text{A.C.}: \begin{cases} u(x,0.25) = \dfrac{65}{64} x(1-x), & u(x,0.50) = \dfrac{9}{8} x(1-x), \\ u(x,0.75) = \dfrac{91}{64} x(1-x), & u(x,1) = 2x(1-x). \end{cases} \end{cases} \tag{6.1}$$

The exact solution of the corresponding direct problem is

$$u(x,t) = x(1-x)(1+t^3)$$

which yields the true parameter values

$$\lambda = 5, \qquad \mu = 3, \qquad \alpha = 0.7.$$

To enhance reliability, the PSO algorithm was executed 10 times under identical settings. For each run, the optimal parameters and the associated objective function error $e_{\lambda,\mu,\alpha}$ were recorded. These results are summarized in Table 6.1, and their variation across the 10 runs is illustrated in Figure 6.1.

Among these runs, the smallest error occurs in the 10th execution, which therefore yields the best recovered parameters. The evolution of the objective function evaluations in this run is depicted in Figure 6.2. As illustrated, the algorithm terminates after 1110 evaluations, with the minimal error achieved at the 1109th evaluation.

The best parameter estimates obtained for the inverse problem are

$$\lambda^* = 4.99782441780948, \qquad \mu^* = 3.00144614870887, \qquad \alpha^* = 0.803793907803762.$$

Figure 6.3 compares the numerical solutions of the direct problem using these reconstructed parameters with the exact solution for $t = 0.25$, $t = 0.50$, $t = 0.75$, and $t = 1$. The close agreement between the two confirms the accuracy of the reconstructed parameters. In the figure, the numerical solutions are marked with red circles.

Table 6.1: Optimal values of λ, μ, and α obtained from 10 independent runs of the PSO algorithm for Example 1, together with the number of iterations (*Iter*), the number of objective function evaluations (*feval*), and the final error $e_{\lambda,\mu,\alpha}$

Run	λ	μ	α	*Iter*	*feval*	$e_{\lambda,\mu,\alpha}$
1	4.9982	2.9987	0.8007	37	1110	8.2037e-09
2	4.9979	3.0006	0.8053	31	930	8.1917e-09
3	4.9985	2.9952	0.79349	32	960	1.3159e-08
4	4.9983	2.9983	0.80089	33	990	8.2367e-09
5	4.9976	3.0041	0.808	33	990	9.0815e-09
6	4.9966	3.0058	0.81109	35	1050	1.3214e-08
7	4.9984	3.0012	0.79975	34	1020	8.1798e-09
8	5.0016	2.9989	0.76419	32	960	2.3995e-08
9	5.0001	3.0025	0.78773	36	1080	1.4463e-08
10	4.9978	3.0014	0.80379	37	1110	8.1712e-09

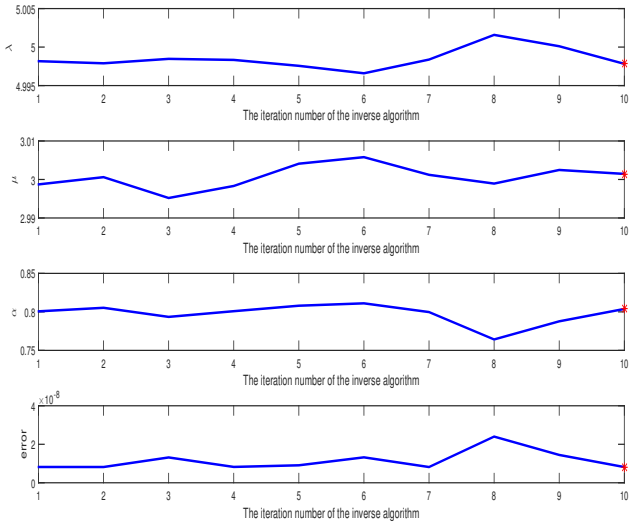

Figure 6.1: Optimal values of λ, μ, and α, together with their corresponding errors $e_{\lambda,\mu,\alpha}$, obtained from 10 runs of the PSO algorithm for Example 1

The error corresponding to these optimal values is

$$e_{\lambda,\mu,\alpha} = 8.17124082614059 \times 10^{-9}.$$

Additional executions of the PSO algorithm may yield even more accurate reconstructions.

To demonstrate the advantages of using PSO algorithm in comparison with the genetic algorithm

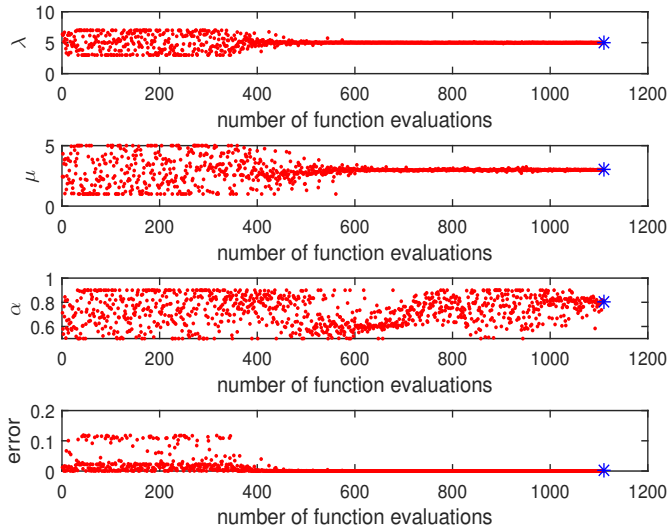

Figure 6.2: Progress of the PSO algorithm in the 10th run, showing the evolution of the parameters λ, μ, and α and the corresponding error values $e_{\lambda,\mu,\alpha}$

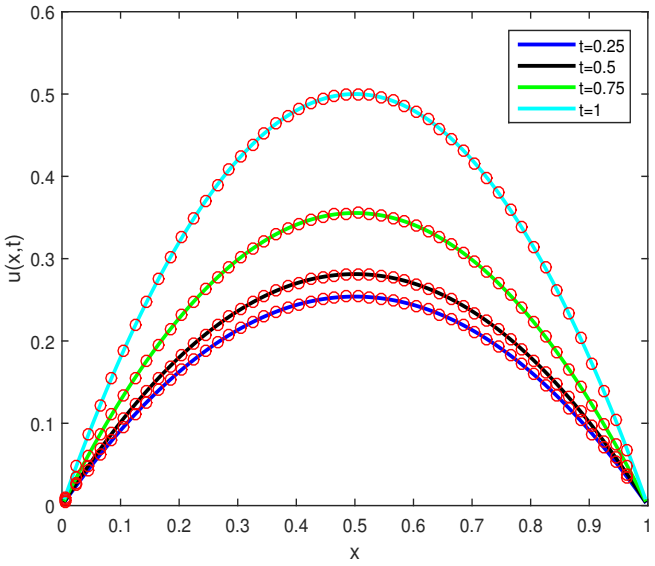

Figure 6.3: Comparison of the numerical solution $\tilde{u}(x,t)$ obtained by the direct method (3.12) using the optimal parameters estimated from the inverse problem (1.6), with the exact solution $u(x,t)$ at four different time levels for Example 1

(GA), we perform a numerical experiment in which

$$f(x,t)=\frac{6}{\Gamma(3.5)}t^{2.5}x(1-x)+2(1+t^3),\qquad \text{A.C.}:\ u(x,1)=2x(1-x).$$

Table 6.2: Optimal values of λ and α obtained from 10 independent runs of the PSO algorithm, together with the number of iterations (*Iter*), the number of objective function evaluations ($feval$), and the corresponding errors $e_{\lambda,\mu,\alpha}$ for Example 1

Run	λ	μ	α	*Iter*	$feval$	$e_{\lambda,\mu,\alpha}$
1	1.0051	0	0.51903	28	840	3.9326e-09
2	1.0160	0	0.3721	22	660	3.7669e-08
3	1.0114	0	0.4406	22	660	3.3343e-08
4	1.0026	0	0.55026	23	690	1.2082e-09
5	0.99761	0	0.61184	27	810	8.5424e-10
6	1.0002	0	0.58072	31	930	6.8913e-12
7	1.0102	0	0.45097	22	660	1.5560e-08
8	1.0192	0	0.32663	22	660	5.4583e-08
9	0.99386	0	0.65452	28	840	5.7520e-09
10	1.0265	0	0.2194	25	750	1.1183e-07

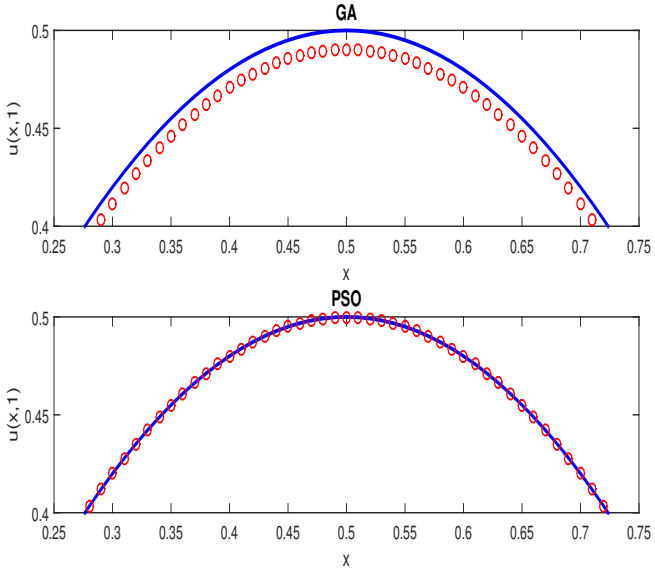

Figure 6.4: Comparison of the exact solution with the numerical solutions obtained using the proposed method with PSO and the GA-based method in [22] for Example 1

The numerical results obtained from 10 independent runs of the PSO algorithm are reported in Table 6.2. Among these runs, the smallest error is achieved in the sixth execution, yielding

$$\lambda = 1.0002, \qquad \alpha = 0.58072,$$

after 930 objective function evaluations. To compare the performance of PSO with that of GA, the approximate solutions obtained by both methods are plotted together with the exact solution in Figure 6.4. Figure 6.5 displays the corresponding squared errors.

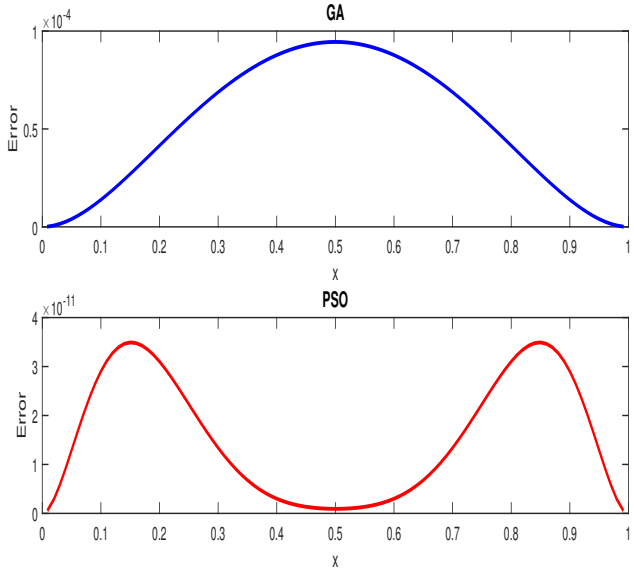

Figure 6.5: Comparison of the squared errors associated with the solutions obtained by PSO and GA for Example 1

As shown in Figure 6.5, the squared error associated with the GA approach is approximately one order of magnitude larger than that obtained by PSO. Furthermore, PSO requires only 930 evaluations of the objective function, whereas GA requires 9867 evaluations. These observations clearly demonstrate the superior efficiency and accuracy of the PSO-based approach compared with the GA-based method reported in [22].

Example 2. Consider the following inverse time-fractional advection-dispersion problem:

$$\begin{cases} \partial_t^{\alpha} u = \lambda u_{xx} - \mu u_x + \frac{6}{\Gamma(3.5)} t^{2.5} \sin(\pi x) + 4\pi^2 (1+t^3)\sin(\pi x) + 3\pi(1+t^3)\cos(\pi x), \\ B.C.: \quad u(0,t) = u(1,t) = 0, \\ I.C.: \quad u(x,0) = \sin(\pi x), \\ A.C.: \begin{cases} u(x,0.25) = \frac{65}{64}\sin(\pi x), & u(x,0.5) = \frac{9}{8}\sin(\pi x), \\ u(x,0.75) = \frac{91}{64}\sin(\pi x), & u(x,1) = 2\sin(\pi x). \end{cases} \end{cases} \tag{6.2}$$

The exact solution of the corresponding direct problem is given by

$$u(x,t) = (1+t^3)\sin(\pi x).$$

Accordingly, the exact parameter values in the inverse problem are

$$\lambda = 4, \qquad \mu = 3, \qquad \alpha = 0.5.$$

To further assess the reliability and stability of the proposed approach, the PSO algorithm is executed 10 times under the same conditions, and the best set of parameters is selected from these runs. The results are reported in Table 6.3 and Figure 6.6.

Table 6.3: Estimated values of λ, μ, and α obtained from 10 independent runs of the PSO algorithm, together with the number of iterations (*Iter*), the number of objective function evaluations ($feval$), and the corresponding errors $e_{\lambda,\mu,\alpha}$ for Example 2

Run	λ	μ	α	*Iter*	$feval$	$e_{\lambda,\mu,\alpha}$
1	3.9997	2.9980	0.56623	43	1290	3.0914e-08
2	3.9995	3.0002	0.56800	38	1140	2.2234e-08
3	3.9993	2.9999	0.57083	37	1110	2.2983e-08
4	3.9994	3.0000	0.56865	33	1590	2.2178e-08
5	3.9994	3.0003	0.56835	37	1110	2.2342e-08
6	3.9995	3.0002	0.56789	38	1140	2.2291e-08
7	3.9991	3.0012	0.57474	37	1110	3.0615e-08
8	3.9994	2.9997	0.56847	47	1410	2.2418e-08
9	3.9995	3.0001	0.56846	40	1200	2.2195e-08
10	3.9993	2.9988	0.57351	36	1080	3.3127e-08

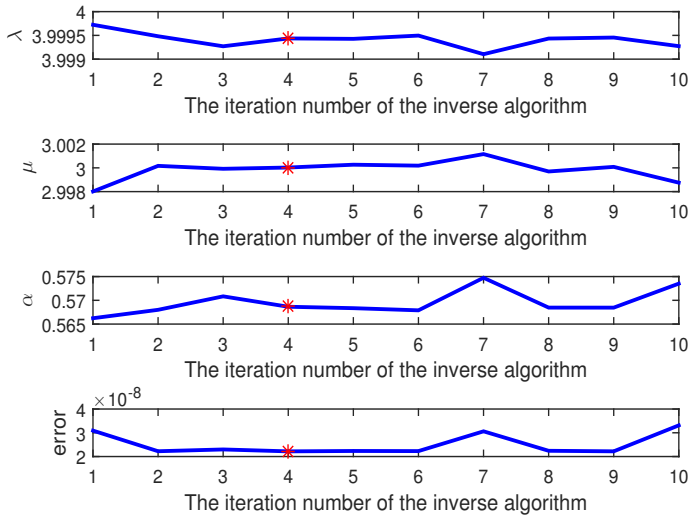

Figure 6.6: Estimated values of λ, μ, and α, together with the corresponding errors $e_{\lambda,\mu,\alpha}$, obtained from 10 independent runs of the PSO algorithm for Example 2

As shown in Table 6.3, the smallest error is obtained in the fourth run. Therefore, the corresponding parameter values are selected as the best estimates. Figure 6.7 illustrates the convergence history of the PSO algorithm in this run. It can be observed that the algorithm terminates after 1590 objective function evaluations, while the minimum error is attained at the 1508th evaluation. The best estimated values for the inverse problem are

$$\lambda^* = 3.99943580833686, \qquad \mu^* = 3.0000270185468, \qquad \alpha^* = 0.568645407778716.$$

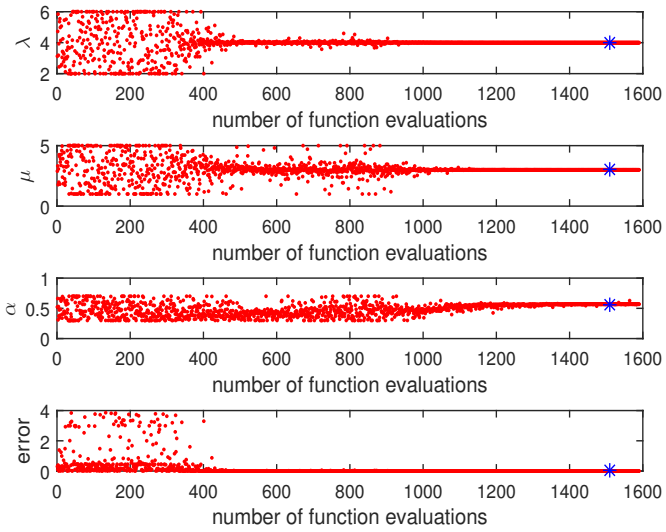

Figure 6.7: Evolution of the estimated values of λ, μ, and α, together with the error $e_{\lambda,\mu,\alpha}$, during the objective function evaluations in the fourth run of the PSO algorithm for Example 2

The corresponding final error is

$$e_{\lambda,\mu,\alpha} = 2.2178 \times 10^{-8}.$$

Figure 6.8 compares the numerical solution obtained by the proposed direct method using the estimated parameter values with the exact solution. It is evident that the numerical and exact solutions are in excellent agreement at $t = 0.25$, $t = 0.5$, $t = 0.75$, and $t = 1$. In Figure 6.8, the numerical solutions are marked by small red circles.

Error of this setting is $e_{\lambda,\mu,\alpha} = 2.21783262358613e-08$. If we run the *PSO* algorithm more times, we might get more accurate solutions for the inverse problem.

Example 3. Consider the inverse fractional advection-dispersion as

$$\begin{cases} \partial_t^{\alpha} u = \lambda u_{xx} - \mu u_x + \frac{-60}{\Gamma(5.7)} x(x-1)t^{4.7} - 5(2-t^5) + (2x-1)(1-0.5t^5), \\ B.C.: \quad u(0,t) = u(1,t) = 0, \\ I.C: \quad u(x,0) = x(x-1), \\ A.C: \begin{cases} u(x,0.25) = \frac{2047}{2048}x(1-x), \quad u(x,0.5) = \frac{63}{64}x(1-x), \\ u(x,0.75) = \frac{1805}{2048}x(1-x), \quad u(x,1) = 0.5x(1-x). \end{cases} \end{cases} \tag{6.3}$$

Exact solution for direct problem of this example is $u(x,t) = x(x-1)(1-0.5t^5)$ and exact solution for inverse problem of this example is $\lambda = 5$, $\mu = 1$ and $\alpha = 0.3$ for more confidence, we execute *PSO* algorithm for 10 times with the same conditions and select the best parameters among them. Table 6.4 and Figure 6.9 show these results.

As it can be seen that the lowest error is achieved in the 10^{th} execution. These parameters can be selected as the best parameters. Figure 6.10 shows the search steps of the *PSO* algorithm in the

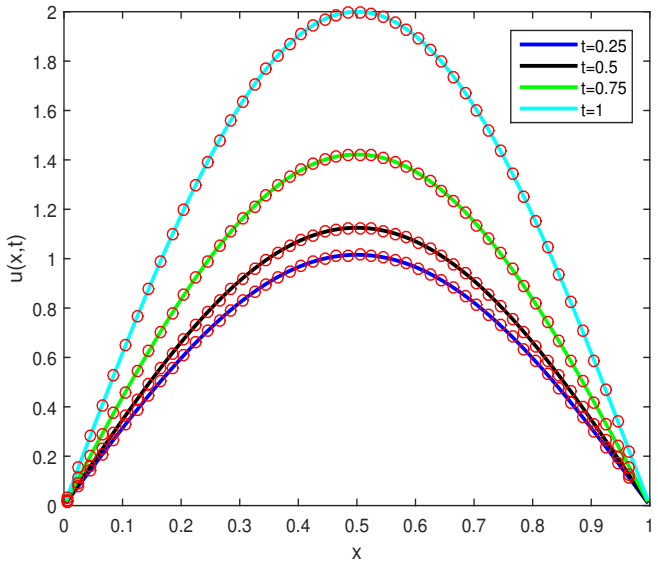

Figure 6.8: Comparison of the numerical solutions $\tilde{u}(x,t)$ obtained by the proposed direct method with the exact solution, using the optimal parameter values identified by the PSO algorithm for the inverse problem, at $t = 0.25$, $t = 0.5$, $t = 0.75$, and $t = 1$ in Example 2

Table 6.4: The optimal values of λ, μ and α obtained in repeated execution of algorithm *PSO* for 10 times and these errors $e_{\lambda,\mu,\alpha}$, in the each time of execution of the algorithm, the number of object function evaluations ($feval$) is inserted. for Example 3

	λ	μ	α	$Iter$	$feval$	$e_{\lambda,\mu,\alpha}$
1	5	0.99888	0.32303	22	660	2.6364e-11
2	5	1.0008	0.32236	31	930	1.6953e-11
3	5.0003	0.99728	0.32259	34	1080	4.8942e-10
4	5.0002	1.0013	0.32878	24	720	6.0917e-10
5	5	0.99896	0.32308	31	930	2.1167e-11
6	5.0002	0.996	0.3223	34	1020	4.1499e-10
7	5.0001	0.99955	0.32351	31	930	1.5217e-11
8	5.0003	1	0.3218	32	960	4.2482e-10
9	5.0001	0.9998	0.32363	34	1020	1.2076e-11
10	5	0.99998	0.32324	35	1050	3.0474e-12

10^{th} run. As it can be seen, the algorithm stopped after 1050 evaluation of the objective function and found the minimum error in 958^{th} evaluation. The optimal values obtained for the inverse problem are $\lambda^* = 5.00001661209564$, $\mu^* = 0.999984384218672$ and $\alpha^* = 0.323236031265377$.
Figure 6.11 compares the numerical and exact solution of the direct problem with these values obtained for the parameters in the inverse problem. As it can be seen, the exact solution and the approximation solution coincide with each other for $t = 0.25$, $t = 0.5$, $t = 0.75$ and $t = 1$. In the Figure 6.9, approximation solutions are drawn with small red circles. Error of this setting is $e_{\lambda,\mu,\alpha} = 3.04742170859597e-12$. If

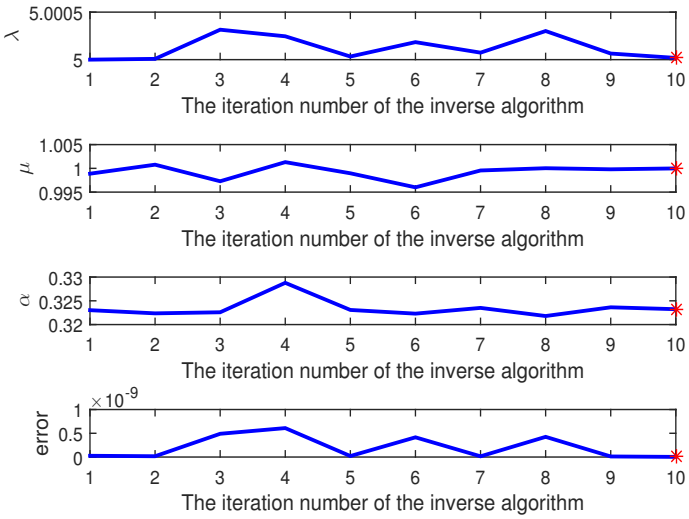

Figure 6.9: The optimal values λ, μ and α obtained in repeated execution of algorithm *PSO* for 10 times and these errors $e_{\lambda,\mu,\alpha}$ in Example 6

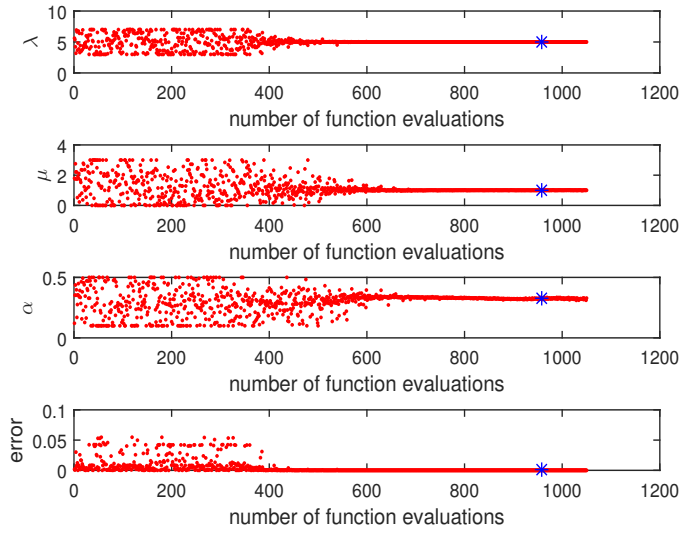

Figure 6.10: The values of λ, μ and α proposed by *PSO* and these errors $e_{\lambda,\mu,\alpha}$ for each time the object function is evaluated. For example 3

we run the *PSO* algorithm more times, we might get more accurate solutions for the inverse problem.

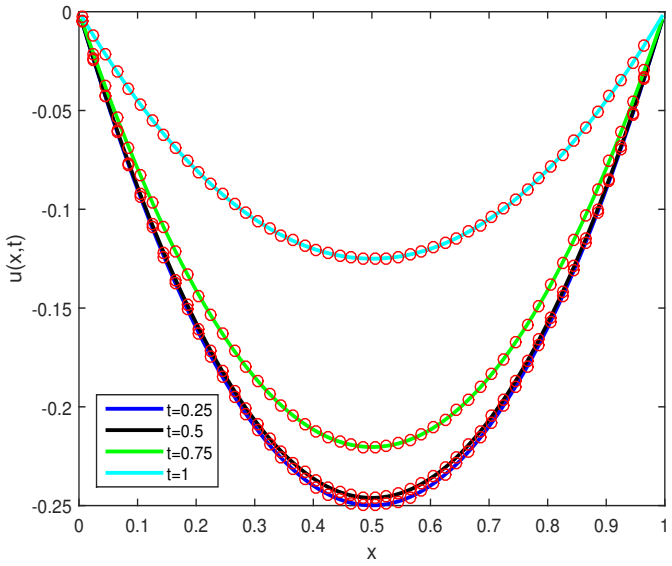

Figure 6.11: Comparison of the numerical solutions $\tilde{u}(x,t)$ obtained from the proposed direct method (3.12) for the direct problem with the optimal parameters obtained from the *PSO* method for the inverse problem (1.6) with monitored data as additional conditions of the problem (for t=0.25, t=0.5, t=0.75 and t=1) in Example 3

7 Conclusion

In modeling natural and physical phenomena using by fractional order derivatives, it is very important to find model parameters, especially the appropriate fractional order. For this purpose, in this article, first a conditionally stable method was presented and then with the help of the *PSO* algorithm, the optimal parameters were obtained with acceptable accuracy. To evaluate the proposed method, three examples are presented. Each of the examples has four additional conditions. These additional conditions , be exact values of the solution at moments $t = 0.25$, $t = 0.5$, $t = 0.75$ and $t = 1$. The inverse problem in Example 1 is solved by two optimization algorithms *GA* and *PSO*. Figure 6.4 and Figure 6.5 shows the superiority of algorithm *PSO*. It is clear that using more accurate algorithms for direct problem makes the parameters obtained from inverse method are more acceptable and the obtained model can better describe the phenomenon. As future work, one can mention the use of other new heuristic optimization algorithms such as the Gray Wolves Optimization algorithm (*GWO*), Starfish Optimization algorithm (*SFO*), Sea Lion Optimization algorithm (*SLO*) and others. Also, for obtain the better numerical methods , one can use the spectral method with suitable orthogonal base.

Competing interests

The authors declare that they have no competing interests.

References

[1] M. Abbas, H. Alotaibi, T. Muhammad, J.F. Gomez-Aguilar, *Significance of Marangoni convection and heat generation on Darcy–Forchheimer 3D flow of Maxwell (AA 7072 + AA 7075-CH_3OH) hybrid nanofluid over a rotating disk*, J. Therm. Anal. Calorim. **150** (2025) 1967–1981.

[2] R. Ashurov, S. Umarov, *Determination of the order of fractional derivative for subdiffusion equation*, Fract. Calc. Appl. Anal. **23** (2020) 1647–1662.

[3] S. Abdi-Mazraeh, H. Kheiri, S. Irandoust-Pakchin, *Construction of operational matrices based on linear cardinal B-spline functions for solving fractional stochastic integro-differential equation*, J. Appl. Math. Comput. **68** (2022) 151–175.

[4] S. Abdi-Mazraeh, A. Khani, S. Irandoust-Pakchin, *Multiple shooting method for solving Black–Scholes equation*, Comput. Econ. **56** (2020) 723–746.

[5] E. Babolian, M. Adabitabar Firozja, B. Agheli, *An approximate method for solving space-time fractional advection-dispersion equation*, J. Ind. Math. **14(1)** (2022) 119–128.

[6] M.R. Dastranj, M. Rouhani, *Design of optimal fractional order PID controller using PSO algorithm*, Int. J. Comput. Theory Eng. **4(3)** (2012) 429–432.

[7] M.H. Derakhshan, Y. Ordokhani, P. Kumar, *A hybrid numerical method with high accuracy to solve a time-space diffusion model in terms of the Caputo and Riesz fractional derivatives*, J. Supercomput. **81(7)** (2025) 863.

[8] S. Irandoust-Pakchin, Sh. Babapour, M. Lakestani, *Image deblurring using adaptive fractional-order shock filter*, Math. Methods Appl. Sci. **44** (2021) 4907–4922.

[9] J. Joshi, A. Rajeev, J.F. Gomez-Aguilar, J.E. Laven-Delgado, *A novel approximation method for two dimensional phase transfer problem in a moving domain with heat generation parameter*, Phys. Scr. **100** (2025), 025259.

[10] F.A. Kamran, F.A. Shah, I. Mahariq, S. Bouzgarrou, S. Aljawi, *Numerical solution of a multi-term time-fractional viscoelastic non-Newtonian fluid model via a hybrid spectral method*, Fractals (2025), 2540180.

[11] I. Karatay, N. Kale, S.R. Bayramoglu, *A new difference scheme for time fractional heat equations based on the Crank–Nicolson method*, Fract. Calc. Appl. Anal. **16** (2013), 892–910.

[12] M.M. Khader, N.H. Sweilam, *Approximate solutions for the fractional advection–dispersion equation using Legendre pseudo-spectral method*, Comput. Appl. Math. **33** (2014) 739–750.

[13] R.A. Khan, S. Yang, S. Fahad, S.U. Khan, Kalimullah, *A modified particle swarm optimization with a smart particle for inverse problems in electromagnetic devices*, IEEE Access **9** (2021) 99932–99943.

[14] M. Krasnoschok, S. Pereverzyev, S.V. Siryk, N. Vasylyeva, *Determination of the fractional order in quasilinear subdiffusion equations*, Fract. Calc. Appl. Anal. **23(3)** (2020) 694–722.

[15] Z. Liu, X. Li, *A Crank–Nicolson difference scheme for the time variable fractional mobile–immobile advection–dispersion equation*, J. Appl. Math. Comput. **56** (2017) 391–410.

[16] F. Liu, V.V. Anh, I. Turner, P. Zhuang, *The fractional advection–dispersion equation*, J. Appl. Math. Comput., **13(1)** (2003) 233–245.

[17] F. Mallawi, J.F. Alzaidy, R.M. Hafez, *Application of a Legendre collocation method to the space–time variable fractional-order advection–dispersion equation*, J. Taibah Univ. Sci. **13(1)** (2019) 324-330.

[18] A.E. Olsson, *Particle Swarm Optimization: Theory, Techniques and Applications*, Nova Science Publishers, 2011.

[19] M. Pashapour, M. Abbaszadeh, M. Dehghan, *Combining finite volume method and physics-informed neural networks for parameter identification and model selection in cell invasion models*, Eur. Phys. J. Plus **140** (2025), 272.

[20] V. Saw, S. Kumar, *Fourth kind shifted Chebyshev polynomials for solving space fractional order advection–dispersion equation based on collocation method and finite difference approximation*, Appl. Comput. Math. **4** (2018) 82.

[21] D. Wang, D. Tan, L. Liu, *Particle swarm optimization algorithm: an overview*, Soft Comput.**22** (2018) 387–408.

[22] G.R. Zaki, A. Jodayree Akbarfam, S. Irandoust-Pakchin, *On the inverse problem of time fractional heat equations by using Crank–Nicolson type methods and genetic algorithm*, Filomat **38(19)** (2024) 505–521.

[23] X. Zhang, R. Liu, *Adaptive fractional image enhancement algorithm based on rough set and particle swarm optimization*, Fractal Fract. **6(2)** (2022) 100.