Filomat 38:19 (2024), 6829–6849
https://doi.org/10.2298/FIL2419829Z

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

On the inverse problem of time fractional heat equation using Crank-Nicholson type method and genetic algorithm

Golamreza Zaki[a], Aliasghar Jodayree Akbarfam[a], Safar Irandoust-Pakchin[a,*]

[a]*Department of Applied Mathematics, Faculty of Mathematics, Statistics and Computer Sciences, University of Tabriz, Tabriz, Iran*

Abstract. In this paper, the time-fractional heat equation with the Caputo derivative of order α where $0 < \alpha \leq 1$ is considered. The parametric Crank-Nicholson type method for direct problems is used. But for the inverse problem, for finding the best conduction parameter c and the best order of fractional derivative α, we use genetic algorithm (GA) for minimizing fitting error such that the numerical solution obtained for direct problem at the final time be fitted by the final conditions. Several examples are carried out to describe the method and to support the theoretical claims. Finally, we conclude that fractional partial differential equations ($FPDE$) are more flexible than partial differential equations (PDE) and have a higher ability to model physical phenomena.

1. Introduction

In the last decades, the fractional differential equations have been studied by many researchers in some physical phenomena and numerous areas such as finance [5-6], engineering [13], viscoelasticity [12], control [7], [9], medical [8], image processing [3, 10, 11] and etc [1, 2, 4]. Due to the ability and flexibility of fractional partial differential equations (FPDEs) for describing scientific phenomena, they have made a very good tools. The accuracy of this FPDEs varies with the order of the fractional derivative. In fact, determining the appropriate order of derivative is very important for modeling a scientific phenomenon and if the derivative order is not selected correctly, the model based on FPDEs may be worse than the model based on PDE.

In recent years, many attempts have been made to determine the appropriate order of the fractional derivative in modeling various phenomena. For example in [14] the determination of the order of fractional derivative and source term in a fractional sub-diffusion equation are discussed. And also only determining of the order of the fractional derivative has been investigated in [15] for the wave equation. Note that determination of the order of the fractional derivative for sub-diffusion equation has been studied in [16]. And as well as in [17] determination of the order of fractional derivative and a kernel in an inverse problem for a generalized time fractional diffusion equation have been discussed. Inverse problems are hard for non-linear problems. Determination of fractional order and conduction parameter is a two-dimensional optimization problem. In these problems, we minimize the error function. This minimization may not

2020 *Mathematics Subject Classification*. 65M32; 65M22.
Keywords. Crank-Nicholson difference scheme; time fractional heat equation; genetic algorithm; inverse initial value problem.
Received: 21 October 2023; Accepted: 09 November 2023
Communicated by Maria Alessandra Ragusa
* Corresponding author: Safar Irandoust-Pakchin
Email addresses: zaki@tabrizu.ac.ir (Golamreza Zaki), akbarfam@tabrizu.ac.ir (Aliasghar Jodayree Akbarfam), s.irandoust@tabrizu.ac.ir (Safar Irandoust-Pakchin)

has a unique solution. Due to the lack of an explicit form of the error function, in these problems, gradient-based methods such as the steepest descent direction method and quasi-Newton methods do not work. Evolutionary optimization methods work much better for these types of inverse problems. Because of these algorithms use random methods, so repeated implementations of these algorithms may not provide the same answers. But this challenge should not be considered an important challenge because by implementing it repeatedly, the best relative solution can be chosen from among them. Probably, the genetic algorithm (*GA*) is the most well-known evolutionary algorithm. Genetic algorithm was developed by John Holland [18], inspired by the Darwinian evolution of biological systems in nature. Solution vectors to an optimization problem are encoded as binary strings of 0s and 1s, called chromosomes. Three genetic operators (crossover, mutation, and selection) are used to modify the strings [19–21]. Two new (child) solutions are formed via crossover from two parent solutions by swapping one part or multiple parts of their chromosomes. Mutation generates a new solution by mutating one bit or multiple bits of an existing solution or binary string. To solve the inverse problem we need a direct numerical algorithm. If the direct numerical algorithm is given then we research the minimum error by setting different values for parameters. We use the parametric Crank-Nicholson type method as a direct method and *GA* for minimization.

2. Problem Statement

Consider the time fractional heat equation in one dimension for homogeneous rods :

$$\begin{cases} \frac{\partial^{\alpha} u(x,t)}{\partial t^{\alpha}} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad 0 \le t, \quad 0 \le x \le 1, \quad 0 < \alpha \le 1, \\ u(x,0) = g(x), \quad 0 \le x \le 1, \\ u(0,t) = 0, \quad u(1,t) = 0, \quad 0 \le t \le 1, \end{cases} \tag{2.1}$$

where f and g are specific functions and the fractional partial derivative is in the Caputo sense:

$$\frac{\partial^{\alpha} u(x,t)}{\partial t^{\alpha}} = \begin{cases} \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{u_t(x,\tau)}{(t-\tau)^{\alpha}} d\tau, \quad 0 < \alpha < 1, \\ u_t(x,t), \quad \alpha = 1. \end{cases} \tag{2.2}$$

If the order of fractional derivative α and conduction parameter c are given, we refer to (2.1) as a *direct problem*. In the direct problem, we try to find the $u(x,t)$ for $0 \le x \le 1, 0 \le t \le 1$.
If the order of fractional derivative α and conduction parameter c or one of them are unknown and, instead of them, the final condition of problem is given, in this case, how to find the best values of α and c such that the numerical solution obtained from numerical algorithm in $t = 1$, fitted by the final condition? In the other words, we seek the conduction parameter c and the order of fractional derivative α in the following problem :

$$\begin{cases} \frac{\partial^{\alpha} u(x,t)}{\partial t^{\alpha}} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad 0 \le t \le 1, \quad 0 \le x \le 1 \quad 0 < \alpha \le 1, \\ u(x,0) = g(x), \quad u(x,1) = h(x), \quad 0 \le x \le 1, \\ u(0,t) = 0, \quad u(1,t) = 0, \quad 0 \le t \le 1. \end{cases} \tag{2.3}$$

where f, g and h are specific functions and the fractional partial derivative is in the Caputo sense (2.2). We refer to (2.3) as a *Inverse problem*. In the inverse problem, we search for the best values of α and c so that the solution of the direct problem for these parameters fits the final condition $h(x)$.

Let us denote the algorithm that solves problem (2.1) for given α, c by $\Lambda_{\alpha,c}$ and its solution by $\tilde{u}_{\alpha,c}$. Note that, $\tilde{u}_{\alpha,c}(x,1)$ is a discrete solution and in the vector form with M components but $u(x,1) = h(x)$ is a continuous function. Therefore for comparing them, we should construct an interpolation function for $\tilde{u}_{\alpha,c}(x_i,1)$, $i = 1, ..., M$ and use the L_2 norm for fitting them :

$$E(\alpha, c) = \left\|\tilde{u}_{\alpha,c} - h\right\|_{L_2} = \int_0^1 \left|\tilde{u}_{\alpha,c}(x,1) - h(x)\right|^2 dx\,, \tag{2.4}$$

or discretization of h and use the l_2 norm:

$$e(\alpha, c) = \left\|\tilde{u}_{\alpha,c} - h\right\|_{l_2} = \sum_{i=1}^{n} \left[\tilde{u}_{\alpha,c}(x_i, 1) - h(x_i)\right]^2. \tag{2.5}$$

For finding the best values of α and c, we minimize the following constrained optimization problem:

$$\min_{\alpha,c} e(\alpha, c) \quad s.t.: \quad 0 < \alpha \leq 1, \quad c_0 \leq c \leq c_1. \tag{2.6}$$

Note that, $0 \leq e(c, \alpha)$ has lower bound. Therefore, it has a minimum on the compact domain $[c_l, c_u]$, $[\alpha_l, \alpha_u]$ and for each evaluation of $e(c, \alpha)$, the direct problem must be solved. In this paper, the genetic algorithm, GA, for solving (2.6) is used.

3. Direct problem and discretization

The Crank-Nicholson type method for solving (2.1) in special case $c = 1$ was introduced in [22] and its stability has been proved for $1 - Ln(2)/Ln(3) \leq \alpha$. In addition, this method has a convergence order of $O\left(\Delta_t^{2-\alpha} + \Delta_x^2\right)$.

In this section, we generalize this method to more general case $c > 0$ and the parametric Crank-Nicholson type method for FPDE is introduced. For discretization, one can use the following nodes:

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

where the Δ_x is step size of x and Δ_t is step size of t. Time fractional derivative is discretized in the following form

$$\begin{aligned} \frac{\partial^\alpha u\left(x_i, t_{j+1/2}\right)}{\partial t^\alpha} &= \frac{1}{\Gamma(1-\alpha)} \int_0^{t_{j+1/2}} u_t(x_i, \tau)\left(t_{j+1/2} - \tau\right)^{-\alpha} d\tau \\ &= \frac{1}{\Gamma(1-\alpha)} \left[\begin{array}{l} \int_0^{t_j} u_t(x_i, \tau)\left(t_{j+1/2} - \tau\right)^{-\alpha} d\tau \\ + \int_{t_j}^{t_{j+1/2}} u_t(x_i, \tau)\left(t_{j+1/2} - \tau\right)^{-\alpha} d\tau \end{array} \right] \\ &= \frac{1}{\Gamma(1-\alpha)} \left[\begin{array}{l} \sum_{k=1}^{j} \int_{t_{k-1}}^{t_k} u_t(x_i, \tau)\left((j+1/2)\Delta_t - \tau\right)^{-\alpha} d\tau \\ + \int_{t_j}^{t_{j+1/2}} u_t(x_i, \tau)\left((j+1/2)\Delta_t - \tau\right)^{-\alpha} d\tau \end{array} \right]. \end{aligned} \tag{3.2}$$

If one use the approximation formula:

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

then (3.2) is written as follows:

$$\begin{aligned}\frac{\partial^{\alpha}u\left(x_i,t_{j+1/2}\right)}{\partial t^{\alpha}} &= \frac{1}{\Gamma(1-\alpha)}\left[\begin{array}{l}\sum_{k=1}^{j}(\frac{u_{i,k}-u_{i,k-1}}{\Delta_t})\int_{t_{k-1}}^{t_k}((j+1/2)\Delta_t-\tau)^{-\alpha}d\tau\\ +(\frac{u_{i,j+1}-u_{i,j}}{\Delta_t})\int_{t_j}^{t_{j+1/2}}((j+1/2)\Delta_t-\tau)^{-\alpha}d\tau\end{array}\right]\\ &= \frac{1}{\Gamma(1-\alpha)}\left[\begin{array}{l}\sum_{k=1}^{j}(\frac{u_{i,k}-u_{i,k-1}}{\Delta_t})\frac{(j-k+3/2)^{1-\alpha}-(j-k+1/2)^{1-\alpha}}{1-\alpha}(\Delta_t)^{1-\alpha}\\ +(\frac{u_{i,j+1}-u_{i,j}}{\Delta_t})\frac{(1/2)^{1-\alpha}-(0)^{1-\alpha}}{1-\alpha}(\Delta_t)^{1-\alpha}\end{array}\right]\\ &= \frac{\Delta_t{}^{-\alpha}}{\Gamma(2-\alpha)}\left\{\frac{\mathrm{u_{i,j+1}-u_{i,j}}}{2^{1-\alpha}}+\sum_{k=1}^{\mathrm{j}}\left(\mathrm{u_{i,k}-u_{i,k-1}}\right)\gamma_{j,k}\right\}+\mathrm{R_1}+\mathrm{R_2}\\ &= -\sigma\gamma_{j,1}u_{i,0}+\sigma\sum_{k=1}^{\mathrm{j}-1}(\gamma_{j,k}-\gamma_{j,k+1})u_{i,k}+\sigma(\gamma_{j,j}-\frac{1}{2^{1-\alpha}})u_{i,j}\\ &+\frac{\sigma}{2^{1-\alpha}}u_{i,j+1}+\mathrm{O}\left(\Delta_{\mathrm{t}}^{2-\alpha}\right)\\ &= \sum_{k=0}^{j+1}d_k u_{i,k}+O(\Delta_t^{2-\alpha}),\end{aligned}\tag{3.3}$$

where
$\sigma=\frac{\Delta_t^{-\alpha}}{\Gamma(2-\alpha)},\quad \gamma_{j,k}=\left(j-k+\frac{3}{2}\right)^{1-\alpha}-\left(j-k+\frac{1}{2}\right)^{1-\alpha},k=1,2,...,j$
$d_{j,0}=-\sigma\gamma_{j,1},\quad d_{j,k}=\sigma(\gamma_{j,k}-\gamma_{j,k+1}),\quad k=1,2,...,j-1$
$d_{j,j}=\sigma(\gamma_{j,j}-\frac{1}{2^{1-\alpha}})=\sigma((\frac{3}{2})^{1-\alpha}-2(\frac{1}{2})^{1-\alpha}),\quad d_{j,j+1}=\sigma\frac{1}{2^{1-\alpha}}$

$$\begin{aligned}R_1 &= \tfrac{1}{\Gamma(1-\alpha)}\textstyle\sum_{k=1}^{j}\int_{(k-1)\Delta_t}^{k\Delta_t}(\tau-t_{k-1/2})\,u_{tt}(x_i,\xi_k)\left((j+1/2)\Delta_t-\tau\right)^{-\alpha}d\tau\\ &\le \tfrac{\Delta_t{}^{2-\alpha}}{\Gamma(1-\alpha)}\max_{1\le i\le n}\left|u_{tt}(x_i,\xi_k)\right|.\\ R_2 &= \tfrac{1}{\Gamma(2-\alpha)}\tfrac{1}{2^{1-\alpha}}O\left(\Delta_t{}^{2-\alpha}\right).\end{aligned}$$

On the other hand,

$$\begin{aligned}u_{xx}\left(x_i,t_{j+1/2}\right) &= \frac{\theta}{(\Delta_x)^2}\left\{u_{i-1,j+1}-2u_{i,j+1}+u_{i+1,j+1}\right\}\\ &+\frac{(1-\theta)}{(\Delta_x)^2}\left\{u_{i-1,j}-2u_{i,j}+u_{i+1,j}\right\}+\mathrm{O}\left(\Delta_{\mathrm{x}}^2\right).\end{aligned}\tag{3.4}$$

By replacing (3.3) and (3.4) in the (2.3) one can obtain

$$\begin{aligned}\sum_{k=0}^{j+1}d_{j,k}u_{i,k} =& \frac{\theta c^2}{(\Delta_x)^2}\left\{u_{i-1,j+1}-2u_{i,j+1}+u_{i+1,j+1}\right\}\\ &+\frac{(1-\theta)c^2}{(\Delta_x)^2}\left\{u_{i-1,j}-2u_{i,j}+u_{i+1,j}\right\}\\ &+f\left(x_i,t_{j+1/2}\right).\end{aligned}\tag{3.5}$$

By rearranging the above system, we have

$$\begin{aligned}
&-ru_{i-1,j+1} + (a+2r)u_{i,j+1} - ru_{i+1,j+1} = \\
&su_{i-1,j} + (b-2s)u_{i,j} + 2su_{i+1,j} \\
&-\sum_{k=0}^{j-1} d_k^j u_{i,k} + f\left(x_i, t_{j+1/2}\right), \quad i = 1,2,...,n-1,
\end{aligned} \tag{3.6}$$

where

$$a = d_{j,j+1} = \frac{\sigma}{2^{1-\alpha}}, \quad b = -d_{j,j} = \sigma(\frac{2}{2^{1-\alpha}} - \frac{3^{1-\alpha}}{2^{1-\alpha}}), \quad r = \frac{c^2\theta}{(\Delta_x)^2}, \quad s = \frac{c^2(1-\theta)}{(\Delta_x)^2}, \tag{3.7}$$

Equation (3.6) for $1 \le i \le n-1$, yields a linear system of (n-1) equations in (n+1) unknowns $\tilde{u}_{0,j+1}, u_{1,j+1}, ..., u_{n,j+1}$. In order to close this system we can use the boundary conditions $\tilde{u}_{0,j} = \tilde{u}_{n,j} = 0$. In the matrix form we have

$$AU_{j+1} = BU_j - \sum_{l=0}^{j-1} d_l^j U_l + F_{j+1/2}, \tag{3.8}$$

where

$$\begin{aligned}
&U_j = \begin{bmatrix} 0 & \tilde{u}_{1,j} & \tilde{u}_{2,j} & \cdots & \tilde{u}_{n-1,j} & 0 \end{bmatrix}^T, \\
&F_{j+1/2} = \begin{bmatrix} 0 & f_{1,j+1/2} & f_{2,j+1/2} & \cdots & f_{n-1,j+1/2} & 0 \end{bmatrix}^T, \\
&A = aI - rT, \\
&B = bI + sT,
\end{aligned} \tag{3.9}$$

where I is an $N \times N$ unit matrix and T is an $N \times N$ three diagonal matrix with diagonal value -2 and upper and lower diagonal value 1 which the first and last rows of this matrix are zero as follows

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

4. Stability and Convergency

The stability of the difference scheme is analyzed by the Fourier method. Let $\tilde{u}_{i,j}$ be the approximate solution and define $\rho_{i,j} = \tilde{u}_{i,j} - u_{i,j}$, $i = 1,2,...,M$, $j = 1,2,...,N$. Then, the following roundoff error equation is obtained as

$$\begin{aligned}
&-\frac{\theta c^2}{(\Delta_x)^2}\rho_{i-1,j+1} + \left(\frac{\sigma}{2^{1-\alpha}} + \frac{2\theta c^2}{(\Delta_x)^2}\right)\rho_{i,j+1} - \frac{\theta c^2}{(\Delta_x)^2}\rho_{i+1,j+1} \\
&= \frac{(1-\theta)c^2}{(\Delta_x)^2}\rho_{i-1,j} + \left(\frac{\sigma}{2^{1-\alpha}} - \frac{2(1-\theta)c^2}{(\Delta_x)^2}\right)\rho_{i,j} + \frac{(1-\theta)c^2}{(\Delta_x)^2}\rho_{i+1,j} \\
&+ \omega_1\rho_{i,j} + \sum_{k=1}^{\mathrm{j}-1}\left(\omega_{\mathrm{i,j-k+1}} - \omega_{\mathrm{i,j-k}}\right)\rho_{i,k} - \omega_{\mathrm{j}}\rho_{i,0}.
\end{aligned} \tag{4.1}$$

The grid functions is defined

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

then $\rho^{j}(x)$ can be expanded in the Fourier series

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

where $d_j(l)=\int_0^1 \rho^{j}(x)\, e^{-\mathbf{i}2\pi l x}dx,\ j=1,2,...,N.$ and $\mathbf{i}=\sqrt{-1}$ is the imaginary unit.

Based on the above analysis, one can suppose that the solution of (4.4) has the following form $\rho_{i,j}=d_j e^{\mathbf{i}i\Delta_x\beta},\ \beta=2\pi l.$
Substituting the above expression into (4.4), it can be obtained

$$\begin{aligned}&-\frac{\theta c^2}{(\Delta_x)^2}d_{j+1}e^{\mathbf{i}(i-1)\Delta_x\beta}+\left(\frac{\sigma}{2^{1-\alpha}}+\frac{2\theta c^2}{(\Delta_x)^2}\right)d_{j+1}e^{\mathbf{i}(i)\Delta_x\beta}\\&-\frac{\theta c^2}{(\Delta_x)^2}d_{j+1}e^{\mathbf{i}(i+1)\Delta_x\beta}=\frac{(1-\theta)c^2}{(\Delta_x)^2}d_je^{\mathbf{i}(i-1)\Delta_x\beta}\\&+\left(\frac{\sigma}{2^{1-\alpha}}-\frac{2(1-\theta)c^2}{(\Delta_x)^2}\right)d_je^{\mathbf{i}(i)\Delta_x\beta}+\frac{(1-\theta)c^2}{(\Delta_x)^2}d_je^{\mathbf{i}(i+1)\Delta_x\beta}\\&+\ \omega_1 d_je^{\mathbf{i}(i)\Delta_x\beta}+\sum_{m=1}^{\mathrm{j}-1}\left(\omega_{\mathrm{j-m+1}}-\omega_{\mathrm{j-m}}\right)d_me^{\mathbf{i}(i)\Delta_x\beta}-\omega_{\mathrm{j}}d_0e^{\mathbf{i}(i)\Delta_x\beta}.\end{aligned}\tag{4.4}$$

After simplifications, we have

$$\begin{aligned}&d_{j+1}\left(-\frac{\theta c^2}{(\Delta_x)^2}\left[e^{\mathbf{i}\Delta_x\beta}+e^{-\mathbf{i}\Delta_x\beta}\right]+\left(\frac{\sigma}{2^{1-\alpha}}+\frac{2\theta c^2}{(\Delta_x)^2}\right)\right)\\&=d_j\left(\frac{(1-\theta)c^2}{(\Delta_x)^2}\left[e^{\mathbf{i}\Delta_x\beta}+e^{-\mathbf{i}\Delta_x\beta}\right]+\left(\frac{\sigma}{2^{1-\alpha}}-\frac{2(1-\theta)c^2}{(\Delta_x)^2}\right)\right)\\&\qquad+\omega_1 d_j+\sum_{m=1}^{\mathrm{j}-1}\left(\omega_{\mathrm{j-m+1}}-\omega_{\mathrm{j-m}}\right)d_m-\omega_{\mathrm{j}}d_0,\end{aligned}\tag{4.5}$$

then, it can be resulted that

$$\begin{aligned}&d_{j+1}\left(-\frac{2\theta c^2}{(\Delta_x)^2}\cos(\Delta_x\beta)+\left(\frac{\sigma}{2^{1-\alpha}}+\frac{2\theta c^2}{(\Delta_x)^2}\right)\right)=\\&d_j\left(\frac{2(1-\theta)c^2}{(\Delta_x)^2}\cos(\Delta_x\beta)+\left(\frac{\sigma}{2^{1-\alpha}}-\frac{2(1-\theta)c^2}{(\Delta_x)^2}\right)\right)\\&\qquad+\omega_1 d_j+\sum_{m=1}^{\mathrm{j}-1}\left(\omega_{\mathrm{j-m+1}}-\omega_{\mathrm{j-m}}\right)d_m-\omega_{\mathrm{j}}d_0.\end{aligned}\tag{4.6}$$

Proposition 4.1. *If* $(\frac{3}{2})^{1-\alpha}-2(\frac{1}{2})^{1-\alpha}<\frac{\sigma(1-2\theta)c^2}{(\Delta_x)^2}\left[\cos(\Delta_x\beta)-1\right]$ *and* $\theta\ge 1/2$ *then* $|d_j|\le|d_0|$, *j= 1,2, ..., N. where* d_j *is the solution of* (4.6).

Proof. The mathematical induction is used for the proof. It is started with k = 0, so

$$
\begin{aligned}
|d_1| &= \left| \frac{\left(\frac{2(1-\theta)c^2}{(\Delta_x)^2} \cos\left(\Delta_x \beta\right) + \left(\frac{\sigma}{2^{1-\alpha}} - \frac{2(1-\theta)c^2}{(\Delta_x)^2} \right) \right)}{\left(-\frac{2\theta c^2}{(\Delta_x)^2} \cos\left(\Delta_x \beta\right) + \left(\frac{\sigma}{2^{1-\alpha}} + \frac{2\theta c^2}{(\Delta_x)^2} \right) \right)} \right| |d_0| \\
&= \left| \frac{\left(\frac{-2(1-\theta)c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} \right| |d_0|.
\end{aligned}
$$

If $\theta \geq 1/2$ then $d_1 \leq d_0$.

Now, assume that $|d_n| \leq |d_0|$, n= 1,2, ..., j. We need to prove this for n=j+1. Indeed,

$$
\begin{aligned}
\left|d_{j+1}\right| &\leq \left\{ \begin{array}{l} \left| \frac{\left(\frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] + \frac{\sigma}{2^{1-\alpha}} - \omega_1 \right)}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} \right| \\ + \frac{\omega_1 + \sum_{m=1}^{\mathrm{j}-1} \left(\omega_{\mathrm{j-m+1}} - \omega_{\mathrm{j-m}} \right) - \omega_{\mathrm{j}}}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} \end{array} \right\} |d_0| \\
&= \left\{ \begin{array}{l} \left| \frac{\left(\frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] + \frac{\sigma}{2^{1-\alpha}} - \omega_1 \right)}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} \right| \\ + \frac{\omega_1}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} \end{array} \right\} |d_0|.
\end{aligned}
$$

If $\left(\frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] + \frac{\sigma}{2^{1-\alpha}} - \omega_1 \right) \geq 0$ *then*

$$
\left|d_{j+1}\right| \leq \frac{\left(\frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] + \frac{\sigma}{2^{1-\alpha}} \right)}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} |d_0| \leq |d_0|.
$$

If $\left(\frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] + \frac{\sigma}{2^{1-\alpha}} - \omega_1 \right) < 0$ *then*

$$
\left|d_{j+1}\right| \leq \frac{\left(2\omega_1 - \frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] - \frac{\sigma}{2^{1-\alpha}} \right)}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} |d_0|.
$$

$$
\begin{aligned}
&\frac{\left(2\omega_1 - \frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] - \frac{\sigma}{2^{1-\alpha}} \right)}{\left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right)} < 1 \\
&\Rightarrow \\
&\left(2\omega_1 - \frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] - \frac{\sigma}{2^{1-\alpha}} \right) < \left(\frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{\sigma}{2^{1-\alpha}} \right) \\
&\Rightarrow \\
&2\omega_1 < \frac{2\theta c^2}{(\Delta_x)^2} \left[1 - \cos\left(\Delta_x \beta\right)\right] + \frac{2\sigma}{2^{1-\alpha}} + \frac{2(1-\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right] \\
&\Rightarrow \\
&\left(\tfrac{3}{2}\right)^{1-\alpha} - 2\left(\tfrac{1}{2}\right)^{1-\alpha} < \frac{\sigma(1-2\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right].
\end{aligned}
$$

Theorem 4.2. *The finite difference scheme* (3.8) *is stable, if $\theta \geq 1/2$ and $\left(\frac{3}{2}\right)^{1-\alpha} - 2\left(\frac{1}{2}\right)^{1-\alpha} < \frac{\sigma(1-2\theta)c^2}{(\Delta_x)^2} \left[\cos\left(\Delta_x \beta\right) - 1\right]$.*

Proof. *Assume* $\theta \geq 1/2$ *and* $(\frac{3}{2})^{1-\alpha} - 2(\frac{1}{2})^{1-\alpha} < \frac{\sigma(1-2\theta)c^2}{(\Delta_x)^2}\left[\cos(\Delta_x \beta) - 1\right]$, *then using Proposition 4.1 and the Parseval equality, we obtain*
$\left\|\rho^j\right\|_2 \leq \left\|\rho^0\right\|_2$, *j=1,2,...,N,*
which means the proposed difference scheme is conditionally stable.

Corollary 4.3. *When* $\theta = 1/2$ *then the finite difference scheme* (3.8) *is stable, if* $1 - log(2)/log(3) \leq \alpha$.

5. Inverse Problem

In the inverse problem, we try to find the most appropriate parameters in such a way that if we solve the direct problem with these parameters, the solution obtained in $t = 1$ fits the final condition. In other words, for each time of checking the fit error (2.5), the direct problem should be solved with the proposed algorithm (3.8) (Crank-Nicholson algorithm). For minimizing (2.6) after solving (2.1) by numerical method $\Lambda_{c,\alpha}$ (i.e. using (3.8) algorithm) and obtaining numerical solution $\tilde{u}_{c,\alpha}(x, 1)$ at the time level $t = 1$ (as final condition), we evaluate the error (2.6). If the value of error is not enough small, we change the values of c and α. How should we change the values of c and α so that the error decreases? Note that, the objective function *direct search* does not have an explicit form, so we cannot use derivative-based methods such as descent direction method, Newton's method, and etc.
One can use the *direct search* and sketch the plot of $e(c, \alpha)$. In other words, for sketch $e(c, \alpha)$ one can evaluate the error $e(c_i, \alpha_j)$ for $c_i = c_l + i\Delta_c$, $\alpha_j = \alpha_l + j\Delta_\alpha$, $i = 0, 1, 2, ..., \frac{c_u - c_l}{\Delta_c}$, $j = 0, 1, 2, ..., \frac{\alpha_u - \alpha_l}{\Delta_\alpha}$ and draw its mesh plot. Note that this method cannot find the parameters with sufficient accuracy and requires a very large number of objective function evaluations to achieve acceptable accuracy.
As an other method, one can use the *random search*(i.e. randomly select α , c and evaluate the error for them (e(α,c)) , then choose the case that corresponds to the minimum error). Note that this method does not provide any guarantee of convergence and no criterion for stopping.

In this paper, we use the Genetic algorithm for minimizing the (2.6). For this purpose, we consider the c parameter as first gene and the α parameter as second gene. In other words, each chromosome in the proposed method, is written as

$$Chromosome = [c, \alpha].$$

5.1. Initial population

After determining the population size (N_{pop}) and intervals related to each of the genes, we generate the N_{pop} chromosomes with random genes. Thus, the initial population will be as follows

$$Chromosome_i = [c_i, \alpha_i], \quad i = 1, 2, ..., N_{pop},$$

where, the $c_i \in [c_l, c_u]$ and $\alpha_i \in [\alpha_l, \alpha_u]$ are randomly generated. For each Chromosome in the population, solve the direct problem and compute the error $e_i = e(c_i, \alpha_i)$ according to (2.5).

5.2. Selection

The fittest chromosomes have higher probability to be selected for the next generation. To compute fitness probability we must compute the fitness of each chromosome. To avoid divide by zero problem, the value of e_i is added by 1.

$$fit_i = \frac{1}{1+e_i}.$$

The probability for each chromosomes is formulated by

$$prob_i = \frac{fit_i}{\sum_{i=1}^{N_{pop}} fit_i}.$$

From the probabilities above we can see that Chromosome with highest fitness, has highest probability to be selected for next generation chromosomes. For the selection process we use roulette wheel process. By using the roulette wheel, choose a mate for each chromosome. note that chromosomes with high fitness have a higher chance of being selected as a mate. If chromosome[j] is selected for chromosome[i], then these two chromosomes will be the parents of chromosome[i] in the next generation.

5.3. *Crossover*

In this paper, we use one-cut point, i.e. randomly select a position in the parent chromosome then exchange these gens with high position. In other word, consider the $chromosome[i]$ and $chromosome[j]$ as his mates then $Newchromosome[i]$ in the next generation, calculated as follows
if random number=0 then:

$$Newchromosome[i] = chromosome[i] >< chromosome[j] = [c_j, \alpha_j],$$

if random number=1 then:

$$Newchromosome[i] = chromosome[i] >< chromosome[j] = [c_i, \alpha_j],$$

if random number=2 then:

$$Newchromosome[i] = chromosome[i] >< chromosome[j] = [c_i, \alpha_i],$$

where >< is the symbol of the crossover action and it means to change the higher gene of the parents.

5.4. *Mutation*

Number of chromosomes that have mutations in a population is determined by the mutation rate parameter(ρ_m). Mutation process is done by replacing the gene at random position with a new value. The process is as follows:
First, we must calculate the total length of gene in the population. In this case the total length of gene is

$$\text{TLG}=N_{gen} * N_{pop}=2 * N_{pop}.$$

Mutation process is done by generating a random integer between 1 and TLG. This random number determines the position of the gene that should mutate and to determine whether this gene will mutate or not, we use another random number. If the generated random number is smaller than mutation rate (ρ_m) then marked the position of gene in chromosomes for mutation. Suppose we define $\rho_m = 0.1$, it is expected that 0.1 of total gene in the population that will be mutated.

$$N_{mut} = \rho_m.TLG = 2\rho_m.N_{ps}.$$

Suppose generation of random number yield r then the chromosome which have mutation are Chromosome number q and gene number p where $r = 2(p-1) + q,\ q = 1, 2$. The value of mutated genes at mutation point is replaced by random number between $[c_l, c_u]$ for first gene and between $[\alpha_l, \alpha_u]$ for the second gene. After finishing mutation process then we have one iteration or one generation of the genetic algorithm.

5.5. *End of first generation*

We can now evaluate the objective function after one generation. Chromosomes with high fitness may not be observed in the new generation, but the mean value of fitness in the new generation will improve compared to the previous generation. In other words, the mean value of error in future generations is a decreasing sequence and if the genetic algorithm converges to the global optimal value, then it will tend to zero.

ALGORITHM

```
Step 0 : Input
    i) f(x,t), g(x), h(x)
    ii) select interval for $c$.
    iii) select interval for $\alpha$
Step 1 : Produce a new generation.
Step 2 : For each of the population chromosomes run the algorithm (3.8) and evaluate of errors (2.5).
Step 3 : Find the chromosome with minimum error.
Step 4 : If the stop condition is not met, go to step 1.
Step 5 : Stop and print the chromosome with minimum error.
```

5.6. Convergence Criteria

Genetic algorithms are a robust search method for finding global optima of functions. This algorithm is very sensitive to the initial population(N_{pop}), mutation(ρ_m), and crossover(ρ_c) parameters, so that, the required time to reach an acceptable solution depends on these parameters [23]. Although they appear to perform well in practice, but there is not a great deal of mathematical work underpinning their performance. The genetic algorithm process is a Markov chain [23] and ergodic process. In other words, the mean error of the generated solution by GA is an almost decreasing sequence (that is, it may not be decreasing for a few elements of the sequence, but it is decreasing in the long time). For more certainty, the reader can refer to the works that have discussed in the convergence of the genetic algorithm, such as [23–25].

6. Numerical results

In this section, in order to evaluate the accuracy of the proposed algorithms, we present three examples. Firstly, we assign values to the unknown parameters and use the direct method to solve the $FPDE$ equation and obtain an approximate solution to $u(x,T)$, then the errors are estimated. Secondly, if the resulting error is not negligible, we change the values of the parameters by GA so that the corresponding error is reduced. In the other word, the GA solve the $IFPDE$.
We use the $\Delta_x = \Delta_t = 1/32$ for direct problem and $c_l = 0.1$, $c_u = 1.5$, $\alpha_l = 0.1$, $\alpha_u = 1$ for inverse problem. When using the direct search for solving inverse problem, we set $\Delta_\alpha = \Delta_c = 0.01$ and When using the $GA - search$ we set $N_{pop} = 50$.

Example 6.1. Consider the following $IFPDE$:

$$\begin{cases} \frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2}, \quad 0 \le t \le 1, \quad 0 \le x \le 1, \quad 0 < \alpha \le 1, \\ u(x,0) = \sin(\pi x), \quad u(x,1) = e^{-1}\sin(\pi x), \quad 0 \le x \le 1, \\ u(0,t) = 0, \quad u(1,t) = 0, \quad 0 \le t \le 1. \end{cases} \tag{6.1}$$

For this example we have :

$$f(x,t) = 0, \quad g(x) = \sin(\pi x), \quad h(x) = e^{-1}\sin(\pi x).$$

If $\alpha = 1$, then exact solution of (6.1) is $c = 1/\pi$. For solving (6.1) we use the classic Crank-Nicholson method ($\theta = 0.5$) for direct problem and GA algorithm for solving inverse problem. We have obtained $c^* = 0.31838099819151$ and $e = 0.000969900429175373$ in 3550 $feval$(Number of function evaluation) as optimal value in the interval $[0.1, 0.5]$. The following figure shows the convergence of the proposed algorithm.

Figure 1 shows the convergence of the GA. After 1000 $feval$, the error significantly decreases to zero. GA has stopped after 3550$feval$ and found appropriate c with error less than 10^{-4} and Figure 2 shows the diagram of $u_c(x,1)$ for different values of c. As it can be seen in Figure 2, the numerical solution $\tilde{u}_{c^*,\alpha=1}(x,1)$ is matched with the exact solution $u_{1/\pi,\alpha=1}(x,1)$ with error less than 10^{-3}.
To get better results, we search for two parameters together. First, we use direct search. Figure 3 shows its 3d plots.

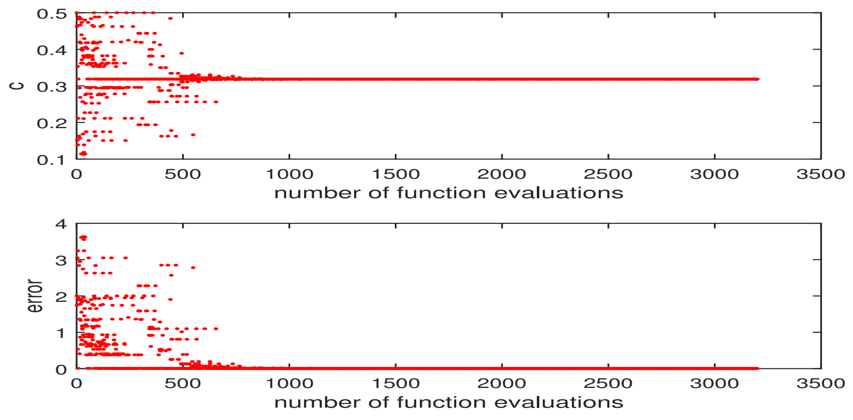

Figure 1: The values of c and these errors($e(c, \alpha = 1)$) in the $GA - search$ for Example 6.1.

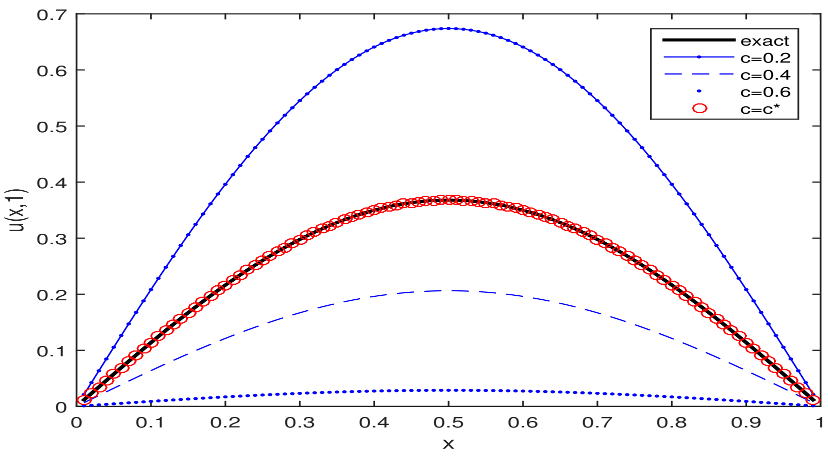

Figure 2: The diagram of $\tilde{u}_{c,\alpha=1}(x, 1)$ for several value of c and compare with exact solution for Example 6.1.

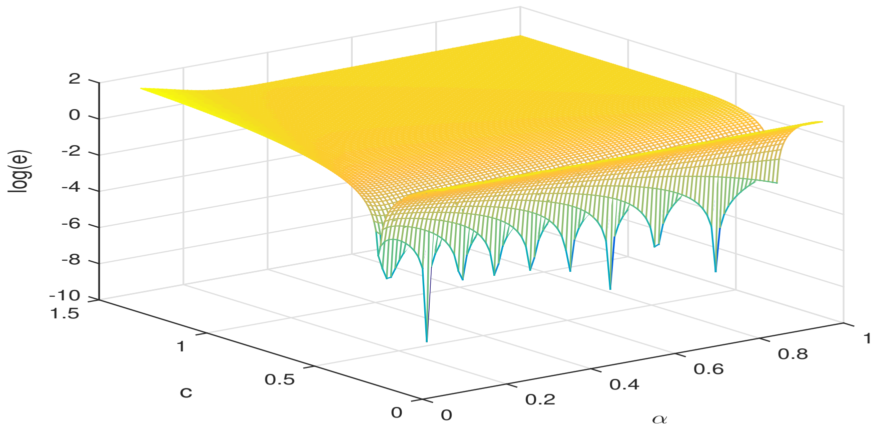

Figure 3: 3d plot of $e(c, \alpha)$ in *direct − search* for Example 6.1.

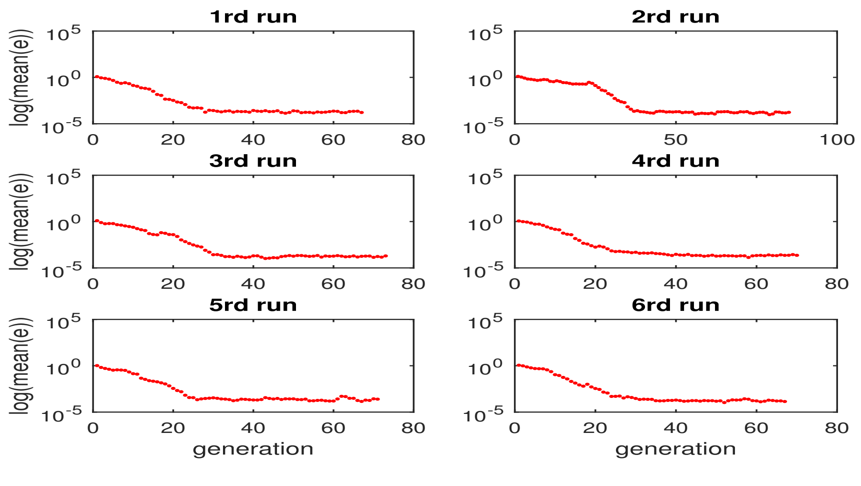

Figure 4: The graph of $log(mean(e))$ (logarithm of the average error in the generations of a GA execute) in 6 times independent executions for Example 6.1.

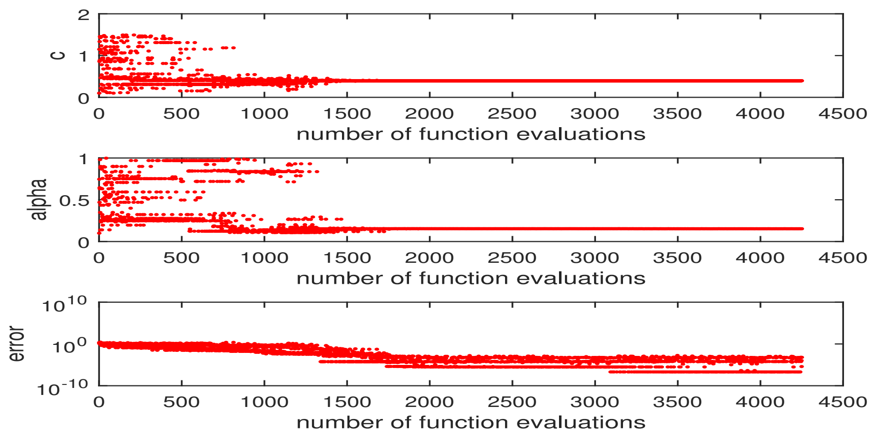

Figure 5: The values of c,α and their error $e(c, \alpha)$ at each $feval$ in the $GA - search$ for Example 6.1.

As it can be seen from Figure 3, the problem has several local minimum and the global minimum($e = 0.00010933$) is achieved in $c = 0.39, \alpha = 0.21$. Table 1 shows the three of the best local minimum.

Table 1: Three cases of best local minimum in $diect - search$.

c	α	e
0.39	0.21	0.00010933
0.34	0.62	0.00041957
0.32	0.86	0.000432

Now, we use the $GA - search$ and repeat the GA execution independently for 6 times. Table 2 shows the results of these independent executions.

Table 2: Optimum values and their errors in 6 times independent executions of $GA - search$ for Example 6.1.

Run	c^*	α^*	e^*	$feval^*$	$feval$	*Generation*
1	0.33966	0.62274	4.6666e-06	3346	3350	67
2	0.39717	0.15292	2.0099e-07	3094	4250	85
3	0.37677	0.31106	1.4865e-06	3587	3650	73
4	0.33792	0.63991	1.217e-06	1424	3500	70
5	0.34278	0.59289	1.0952e-06	3019	3550	71
6	0.38075	0.28058	1.0982e-05	2788	3350	67

Table 2 shows, the best error was obtained in the second run and the worst in the sixth run. Comparing Tables 1 and 2 shows that $GA - search$ is more than 10 times better than the best solution of $direct - search$ even in the worst case. This is despite the fact that the best solution of $direct - search$ was obtained after 15,000 $feval$ and $GA - search$ after 67 generations (3,350 $feval$). Figure 5, shows the details of the algorithm in the second run.

It should be noted that the two-parameter search gave much better results than the one-parameter search. In other words, *FPDE* models the problem much better than *PDE*.

Example 6.2. Consider the following *IFPDE*:

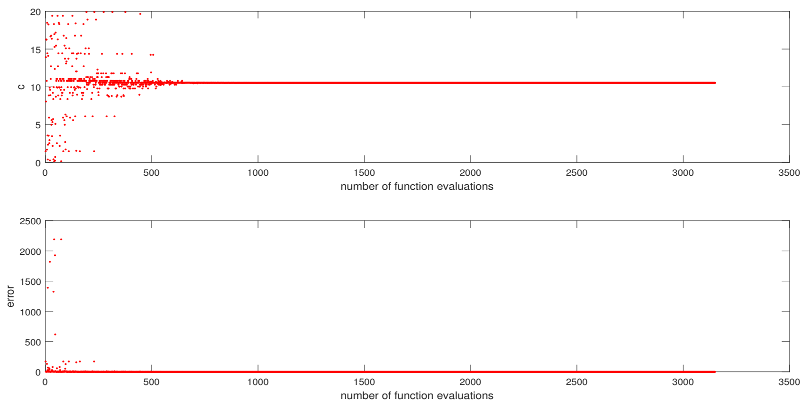

Figure 6: The values of c and their errors in the GA iterations for Example 6.2 when $\alpha = 1$ and $0.1 \le c \le 20$.

$$\begin{cases} \frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} + x(1-x)\frac{6t^{3-1/2}}{\Gamma(4-1/2)} + 2\left(t^3+1\right), \\ 0 \le t \le 1, \quad 0 \le x \le 1, \quad 0 < \alpha \le 1, \\ u(x,0) = x(1-x), \quad u(x,1) = 2x(1-x), \quad 0 \le x \le 1, \\ u(0,t) = 0, \quad u(1,t) = 0, \quad 0 \le t \le 1. \end{cases} \tag{6.2}$$

For this example we have :

$$f(x,t) = x(1-x)\frac{6t^{3-1/2}}{\Gamma(4-1/2)} + 2\left(t^3+1\right), \quad g(x) = x(1-x),$$

$$h(x) = 2x(1-x).$$

For $c = 1$, $\alpha = 0.5$ the exact solution of direct problem of (6.2) is $u(x,t) = (1-x)x(t^3+1)$. Therefore, for given $f(x,t)$, $g(x)$ and $h(x)$, the exact solution of inverse problem of (6.2) is $c = 1$ and $\alpha = 0.5$.
Let us set $\alpha = 1$. In this case, the fractional partial differential equation of the problem will become the partial differential equation. For solving the inverse problem, we select the appropriate interval $[c_l, c_u]$ for searching the optimal value for c. We use the classical Crank-Nicholson method for the direct problem. When $\alpha = 1$ and $0.1 \le c \le 1$ then GA terminated after 4000 $feval$ and obtained $c^* = 0.999999992201451$ with $error = 347.513626451307$ in 3242 $feval$. Since the obtained optimal value is near to upper band of the searching interval, so we expand the searching interval to get a better value for c. Now, instead of the interval $0.1 \le c \le 1$, we set $0.1 \le c \le 3$. In this case, GA terminated after 4000 $feval$, and the value $c^* = 2.99999999051722$ is obtained with $error = 40.5466118651176$.The optimal value of c is still near the upper bound. Again, we expand the search interval. Now, we consider the interval $0.1 \le c \le 5$. The GA is terminated after 3850 $feval$ and the value $c^* = 4.99999999294413$ with $error = 12.4397071031382$ is obtained. Still, the optimal value of c is near to upper bound. We expand the range of the interval for the third time. By considering the interval $0.1 \le c \le 10$ we see that GA terminated after 4300 $feval$ and $c^* = 9.99999999241637$ with $error = 0.390808865742124$ in the 2792 $feval$. If the searching interval is $0.1 \le c \le 20$ then GA terminated after 3150 $feval$ and the value of $c^* = 10.5221462486276$ with $error = 0.0101026234498751$ is obtained. Since this time, the optimal value of c is not near to the upper or the lower bounds, so there is no hope of improving the value of c by extending the interval. Figure 6 shows the details of this search for $\alpha = 1$ and $0.1 \le c \le 20$.

Now, we are searching the values of c and α together and we hope to get better values for them. By setting $0.1 \le \alpha \le 1$ and $0.1 \le c \le 1.5$ and using the suggested fractional Crank-Nicholson method for direct

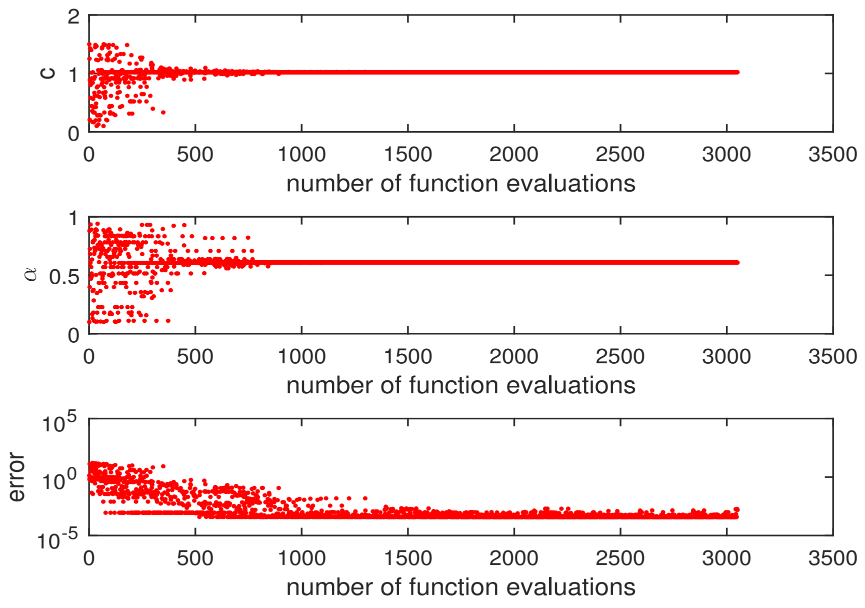

Figure 7: The values of c, α and their errors in GA iterations for example 6.2 when $0.1 \leq \alpha \leq 1$ and $0.1 \leq c \leq 1.5$.

problem, we see that the GA stopped after the 61 *Generation* (3050 *feval*) and found the $c^* = 1.03006913150768$, $\alpha^* = 0.486675726623848$ with $error = 0.000802284984493785$. Figure 7 shows the details of this search for $0.1 \leq \alpha \leq 1$ and $0.1 \leq c \leq 1.5$.

For more certainty, we run the algorithm GA for 6 times independently, and the results are shown in Table 3. These results show that the algorithm has fallen into a local minimum on some runs. For example, in the second run it reaches a local minimum $c^* = 1.0378, \alpha^* = 0.24112$ and in the fifth run it reaches an other local minimum $c^* = 1.0374, \alpha^* = 0.25730$, while in the third run $c^* = 1.0257, \alpha^* = 0.51566$ and sixth run $c^* = 1.0216, \alpha^* = 0.59429$ it approaches the global minimum and is much closer to the exact solution $c^* = 1, \alpha^* = 0.5$.

Table 3: Optimum values and their errors in 6 times independent executions of $GA - search$ for Example 6.2.

Run	c^*	α^*	e^*	$feval^*$	$feval$	$Generation$
1	1.0209	0.60767	0.0003749	3045	3050	61
2	1.0378	0.24112	0.0034914	6495	6550	123
3	1.0257	0.51566	0.0003635	2919	3450	69
4	1.0167	0.68134	0.0009458	3385	3450	69
5	1.0374	0.25730	0.0033207	9999	10050	201
6	1.0216	0.59429	0.0002769	3549	3650	73

To show the ergodic behavior of the GA, we obtain the average error in each generation. Figure 8, shows that the average error in the generations of one execution of GA is almost decreasing.

For the direct method, after 1500 *feval* the following results are obtained. Figure 9, shows the 3d plot of log(e) and Table 4 shows the three of the best local minimum.

Table 4: Three cases of best local minimum in $diect - search$ for Example 6.2.

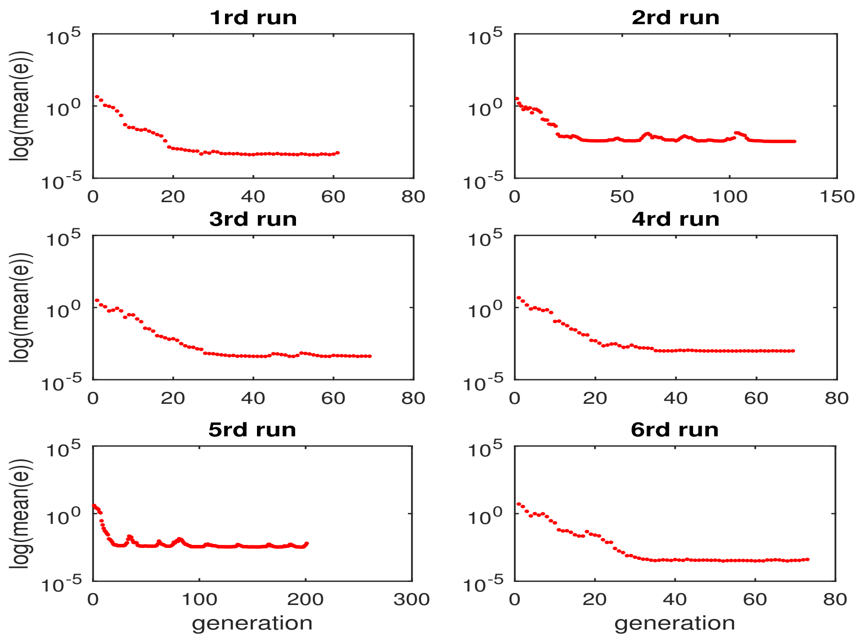

Figure 8: The graph of $log(mean(e))$ (logarithm of the average error in the generations of a GA execute) in 6 times independent executions for Example 6.2.

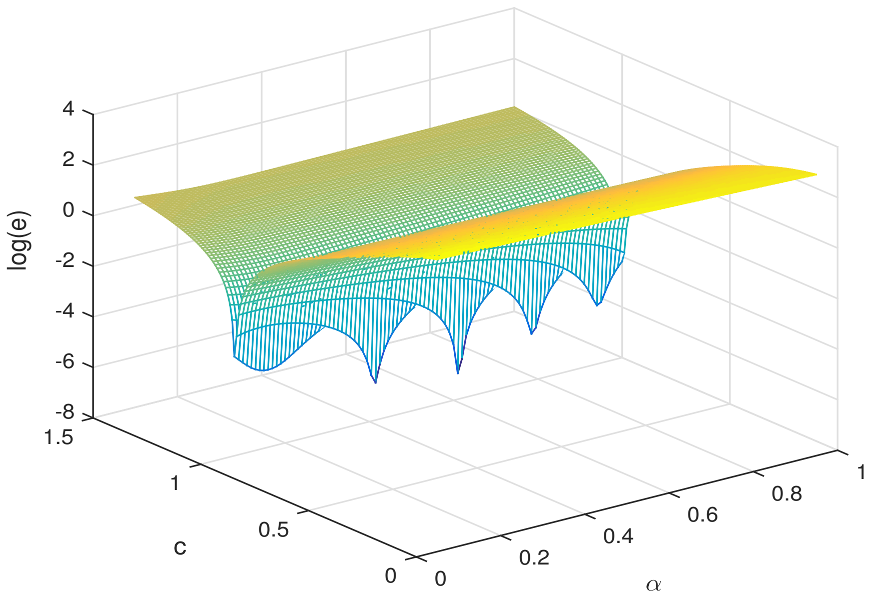

Figure 9: 3d plot of $e(c, \alpha)$ in $direct - search$ for Example 6.2 when $0.1 \leq \alpha \leq 1$ and $0.1 \leq c \leq 1.5$.

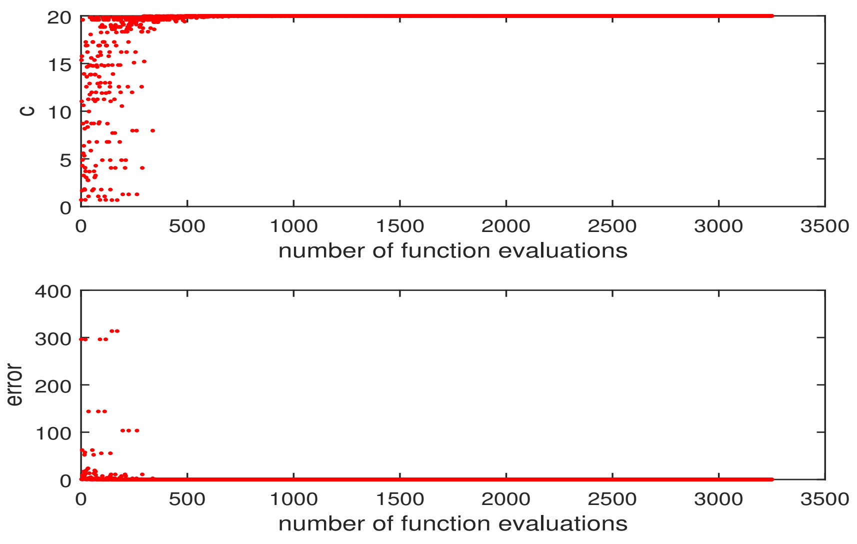

Figure 10: The values of c and their errors in GA iterations for Example 6.3 when $\alpha = 1$ and $0 < c < 20$.

c	α	e
1.02	0.62	0.00083425
1.03	0.43	0.0012313
1.02	0.63	0.0014089

As it can be seen, $GA - search$ has performed much better than $direct - search$. Comparison of Table 3 and Table 4 shows that $GA - search$ has obtained much fewer errors with a smaller number of $feval$.

Example 6.3. Consider the following FPDE:

$$\begin{cases} \frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2} + \left(t^3 + 1\right), \quad 0 \le t, \quad 0 \le x \le 1, \quad 0 < \alpha \le 1, \\ u\left(x, 0\right) = x\left(1 - x\right), \quad u\left(x, 1\right) = x\left(1 - x\right)/5, \quad 0 \le x \le 1, \\ u\left(0, t\right) = 0, \quad u\left(1, t\right) = 0, \quad 0 \le t \le 1, \end{cases} \tag{6.3}$$

For this example we have:
$f(x,t) = (t^3 + 1), \quad g(x) = x(1 - x), \quad h(x) = 0.2x(1 - x).$

Set c_l=0, c_u=20 and α=1. we obtain the $c^* = 19.9999999971851$ as optimal value of c in the interval [0,20] by GA. Solving the direct problem (6.3) by this value as FPDE by using the classical Crank-Nicholson, we see that the error is 0.101159857056873. Figure 10 shows its details. It is observed that the optimal value of c is obtained near the upper bound of the interval. It seems that choosing a larger range can have better results. We set $c_l = 0$, $c_u = 30$, $\alpha = 1$ and search to find the optimal value of c by GA algorithm. The optimal value of c in this interval is obtained as $c^* = 22.5171260637894$. For this selection of c, the value of error is 0.0272536097018295. Figure 11 shows its details.

Extending the interval caused that the error improve slightly. Because the optimal value of c is not obtained at the near of boundaries of the interval, there is no hope of finding a better value for c by extending the interval.
To further reduce the error, we search c, α together and hope to find better value for them. We set $c_l = 0.1$, $c_u = 1.5$, $\alpha_l = 0.1$, $\alpha_u = 1$ and use the GA algorithm.

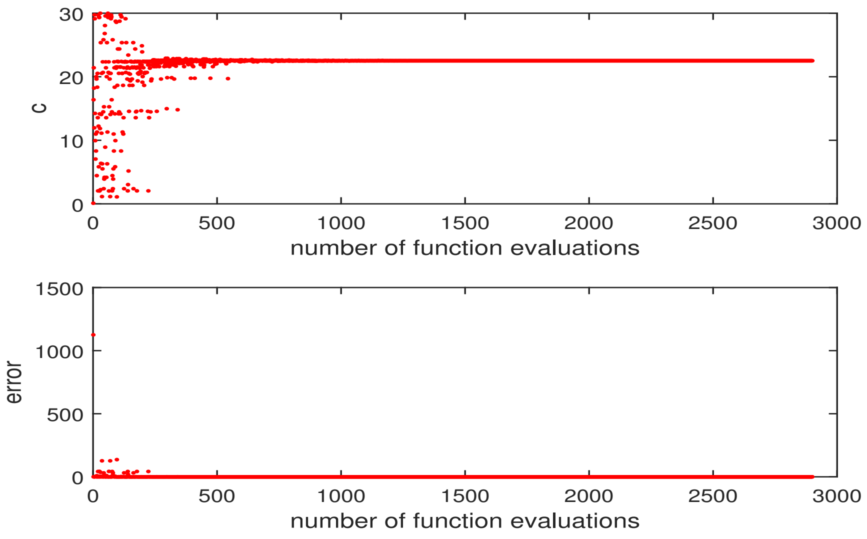

Figure 11: The values of c and their errors in GA iterations for Example 6.3 when $\alpha = 1$ and $0 < c < 30$.

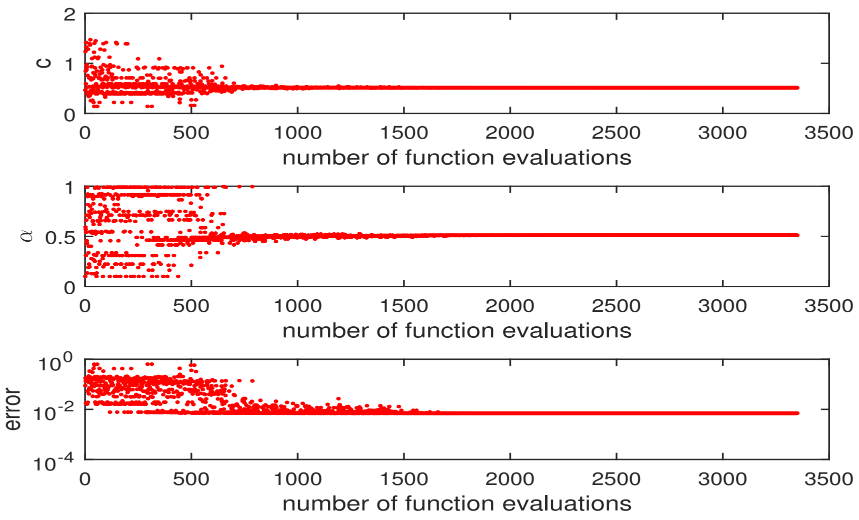

Figure 12: The values of c, α and their errors $e(c, \alpha)$ in GA iterations for Example 6.3 when $0.1 \leq \alpha \leq 1$ and $0.1 \leq c \leq 1.5$.

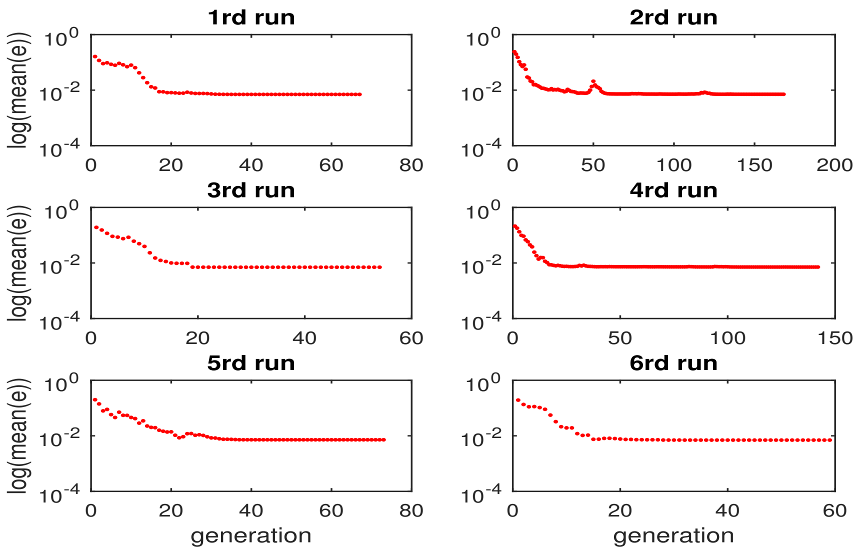

Figure 13: The graph of $log(mean(e))$ (logarithm of the average error in the generations of a GA execute) in 6 times independent executions for Example 6.3.

After 67 generation(3350 $feval$), the algorithm GA stops and values of $c^* = 0.51157$ and $\alpha = 0.51072$ are obtained. Solving the direct problem (6.3) by these values as FPDE by using the proposed Crank-Nicholson type method, we see that the error is $e = 0.0070215$. Figure 12 shows its details.
For more certainty, we run the algorithm GA for 6 times independently, and the results are shown in the Table 5. In all executions, the error is less than 7.2×10^{-3}, and in the sixth execution, the error is less than other executions. So the best values of c and α are $c = 0.5055$ and $\alpha = 0.53257$.

Table 5: Optimum values and their errors in 6 times independent executions of $GA - search$ for Example 6.3.

Run	c^*	α^*	e^*	$feval^*$	$feval$	$Generation$
1	0.51157	0.51072	0.0070215	3342	3350	67
2	0.526	0.45839	0.0070931	8355	8400	168
3	0.49122	0.58395	0.0070357	2693	2700	54
4	0.52962	0.44609	0.0071311	7091	7100	142
5	0.45453	0.72006	0.0071908	3595	3650	73
6	0.5055	0.53257	0.0070167	2069	2950	59

To show the ergodic behavior of the GA, we obtain the average error in each generation. Figure 13 shows that the average error in the generations of one execution of GA is almost decreasing.
The results obtained for this example shows that FPDE model of the problem is much better than PDE. As can be seen, the error of PDE model is at least 10 times larger than the error of FPDE model.
To show the superiority of the proposed algorithm($GA - search$) over the $direct - search$ algorithm, we have displayed the $direct - search$ results after 15,000 $feval$ in Figure. 14 and listed the three of best local minimum in Table 6.

Table 6: Three cases of best local minimum
in $diect - search$ for Example 6.3.

c	α	e
0.50	0.55	0.0070368
0.49	0.59	0.0070489
0.52	0.48	0.0070502

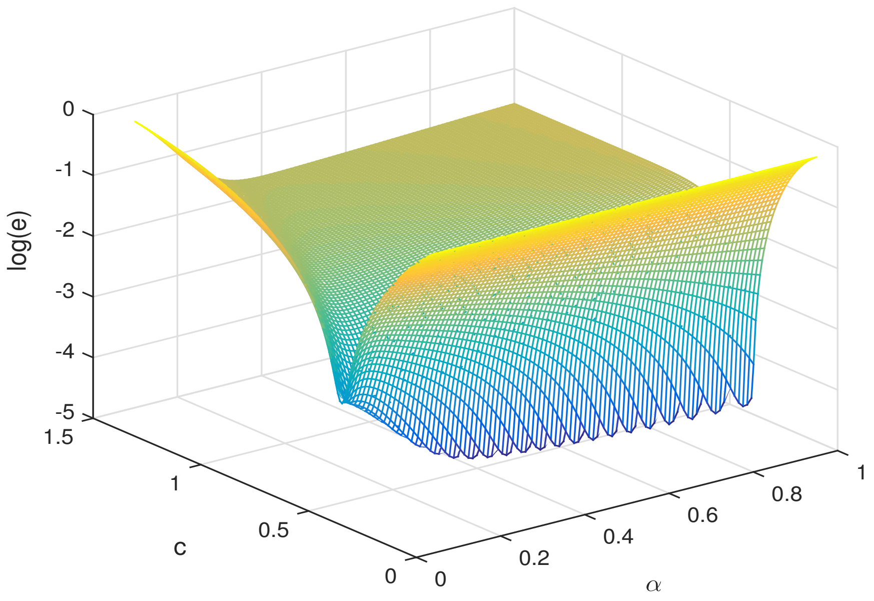

Figure 14: 3d plot of $e(c, \alpha)$ in *direct − search* for Example 6.3 when $0.1 \leq \alpha \leq 1$ and $0.1 \leq c \leq 1.5$.

For this example, the superiority of the proposed algorithm(*GA − search*) over the *direct − search* is only in the number of evaluations of the object function(*feval*), and both methods have obtained almost the same values for error.

7. Conclusion

In this work, the parametric Crank-Nicholson difference scheme was successfully extended to solve a time fractional heat equation with Caputo derivative. We selected $\theta = 0.5$ and used *GA* algorithm for finding the best value of c and α. We illustrated that modeling with FPDE is much better than modeling with PDE. In this work, the parametric Crank-Nicholson difference method was successfully extended to solve a time-fractional heat equation with Caputo derivative. We chose $\theta = 0.5$ and used the algorithm *GA* to find the best value for c and α. We have shown that modeling with FPDE is much better than modeling with PDE.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Competing interests

The authors declare that they have no competing interests.

Funding

Not applicable.

Authors' contributions

The authors declare that the study was realized in collaboration with equal responsibility. All authors read and approved the final manuscript.

References

[1] M. Alotaibi, M. Jleli, M. A. Ragusa and B. Samet, *On the absence of global weak solutions for a nonlinear time-fractional Schrödinger equation*, Applicable Analysis. (2023), 1–15 https://doi.org/10.1080/00036811.2022.2036335

[2] H. Boujemaa, B. Oulgiht, M. A. Ragusa, *A new class of fractional Orlicz-Sobolev space and singular elliptic problems*, Journal of Mathematical Analysis and Applications. (2023), **526**(1),127342. https://doi.org/10.1016/j.jmaa.2023.127342

[3] S. Irandoust-pakchin, Sh. Babapour, M. Lakestani, *Image deblurring using adaptive fractional-order shock filter*, Mathematical Methods in the Applied Sciences. (2021)**44**(6), 4907–4922. https://doi.org/10.1002/mma.7076

[4] I. Fahimi-khalilabad, S. Irandoust-pakchin, S. Abdi-mazraeh,,*High-order finite difference method based on linear barycentric rational interpolation for Caputo type sub-diffusion equation*, Mathematics and Computers in Simulation. (2022), **199**, 60–80, https://doi.org/10.1016/j.matcom.2022.03.008.

[5] P. Roul, *A high accuracy numerical method and its convergence for time-fractional Black-Scholes equation governing European options*, Applied Numerical Mathematics. (2020), **151**, 472–493; https://doi.org/10.1016/j.apnum.2019.11.004

[6] W. Chen, X. Xu, S. P. Zhu, *Analytically pricing double barrier options based on a time-fractional Black–Scholes equation*, Computers and Mathematics with Applications. (2015) **69**, 1407–1419, https://doi.org/10.1016/j.camwa.2015.03.025

[7] P. Warrier, P. Shah, *Fractional order control of power electronic converters in industrial drives and renewable energy systems: A Review*, IEEE Access. (2021), **9**, 58982–59009, https://doi.org/10.1109/ACCESS.2021.3073033

[8] A. Jajarmi, D. Baleanu, K. Z. Vahid, S. Mobayen, *A general fractional formulation and tracking control for immunogenic tumor dynamics*, Mathematical Methods in the Applied Scinces. (2022), **45** (2) 667–680, https://doi.org/10.1002/mma.7804

[9] I. S. Jesus, J.A. T. Machado, *Fractional control of heat diffusion systems*, Nonlinear Dynamics. (2008), **54**: 263–282; https://doi.org/10.1007/s11071-007-9322-2

[10] R. W. Ibrahim, H. A. Jalab, F. K. Karim, E. Alabdulkreem, *A medical image enhancement based on generalized class of fractional partial differential equations*, Quantitative Imaging Medicine Surgery. (2022), **12**(1): 172–183. https://doi.org/10.21037/qims-21-15

[11] A. Gupta, S. Kumar, *Generalized framework for the design of adaptive fractional-order masks for image denoising*, Digital Signal Processing. (2022), **121**, https://doi.org/10.1016/j.dsp.2021.103305

[12] A. Muliana, *A fractional model of nonlinear multiaxial viscoelastic behaviors*, Mechanics of Time-Dependent Materials. (2022); https://doi.org/10.1007/s11043-022-09542-3

[13] A. Elsadany, A. Al-khedhairi, H. N. Agiza, B. Xin, A. Elsonbaty, *Discrete Fractional-Order Systems with Applications in Engineering and Natural Sciences*, Mathematical Problems in Engineering. (2022), ID 9767920 https://doi.org/10.1155/2022/9767920

[14] R. Ashurov, R.T. Zunnunov , *Inverse problem for determining the order of fractional derivative in mixed-type equations*, Lobachevskii Journal of Mathematics. (2021), **42**, 2714–2729. https://doi.org/10.1134/S1995080221120052

[15] R. Ashurov, Y. Fayziev, *Inverse Problem for finding the order of the fractional derivative in the wave equation*, Mathematical Notes. (2021) **110**, 842–852. https://doi.org/10.1134/S0001434621110213

[16] R. Ashurov, S. Umarov, *Determination of the order of fractional derivative for subdifusion equation*, Fractional Calculus Applied Analysis in Engineering, (2020), **23** 1647–1662. https://doi.org/10.1515/fca-2020-0081

[17] J. Janno, *Determination of the order of fractional derivative and a kernel in an inverse problem for a generalized time fractional diffusion equation*, Electronic Journal of Differential Equations. (2016), **199**, 1–28.

[18] J. Holland, *Adaptation in Nature and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA, 1975.

[19] K. De Jong, *Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, Ph.D. Thesis. University of Michigan, Ann Arbor, MI, USA, 1975.

[20] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison- Wesley, Reading, MA, USA, 1989.

[21] X.S. Yang, *Nature-Inspired Optimization Algorithms*, Elsevier Insight, London, 2014.

[22] I. Karatay, N. Kale, S. R. Bayramoglu *A new difference scheme for time fractional heat equation base on the Crank-Nicholson method*, Fractional Calculus Applied Analysis. (2013), **16**, 892–910. https://doi.org/10.2478/s13540-013-0055-2

[23] D. Greenhalgh, S. Marshall, *Convergence criteria for Genetic Algorithms*, Siam Journal of Computing. (2000) **30**(1), 269–282. https://doi.org/10.1137/S009753979732565X

[24] G. Rudolph, *On convergence analysis of particle swarm optimization algorithm*,IEEE Transactions on Neural Networks. (1994) **5** (1), 96–101 https://doi.org/10.1109/72.265964

[25] D. Barrios, L. Malumbres, J. Rios, *Convergence conditions of genetic algorithms*, International Journal of Computer Mathematics. (1998) **68**:3-4, 231–241, https://doi.org/10.1080/00207169808804691