Enhanced Numerical Solutions for Fractional PDEs Using Monte Carlo PINNs Coupled with Cuckoo Search Optimization

Tauqeer Ahmad , Muhammad Sulaiman , David Bassir * , Fahad Sameer Alshammari , Ghaylen Laouini

Posted Date: 14 January 2025

doi: 10.20944/preprints202501.1072.v1

Keywords: Fractional partial differential equations; Physics-Informed Neural Networks; Monte Carlo methods; Cuckoo Search algorithm; numerical analysis; computational efficiency

Article

Enhanced Numerical Solutions for Fractional PDEs Using Monte Carlo PINNs Coupled with Cuckoo Search Optimization

Tauqeer Ahmad [1], Muhammad Sulaiman [1], David Bassir [2,3,*], Fahad Sameer Alshammari [4] and Ghaylen Laouini [5]

[1] Department of Mathematics, Abdul Wali Khan University Mardan, Mardan 23200, Pakistan; tauqeerahmad899@gmail.com (T.A.); msulaiman@awkum.edu.pk (M.S.)
[2] Smart Structural Health Monitoring and Control Lab (SSHMC) Lab, CNAM-Dongguan University of Technology, D1, Daxue Rd., Songshan Lake, Dongguan 523000, China
[3] UTBM, IRAMAT UMR 7065-CNRS, Rue de Leupe, CEDEX, 90010 Belfort, France
[4] Department of Mathematics, College of Science and Humanities in Alkharj, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia; f.alshammari@psau.edu.sa
[5] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait; ghaylen.laouini@aum.edu.kw
* Correspondence: david.bassir@utbm.fr

Abstract: In this study, we introduce an innovative approach for addressing fractional partial differential equations (fPDEs) by combining Monte Carlo-based Physics-Informed Neural Networks (PINNs) with the Cuckoo Search (CS) optimization algorithm, termed PINN-CS. There is a further enhancement in the application of quasi-Monte Carlo assessment that comes with high efficiency and computational solutions to estimates of fractional derivatives. By employing structured sampling nodes comparable to techniques used in finite-difference approaches on staggered or irregular grids, the proposed PINN-CS minimizes storage and computation costs while maintaining high precision in estimating solutions. This is supported by numerous numerical simulations to analyze various high-dimensional phenomena in various environments, comprising of two-dimensional space-fractional Poisson equations, two-dimensional time-space fractional diffusion equations and three-dimensional fractional Bloch-Torrey equations. The results demonstrate that PINN-CS achieves superior numerical accuracy and computational efficiency compared to traditional fPINN and Monte Carlo fPINN methods. Further, the extended use to problem areas with irregular geometries and difficult to define boundary conditions makes the method immensely practical. This research thus lays a foundation for more adaptive and accurate use of hybrid techniques in the development of the fractional differential equations, in computing science and engineering.

Keywords: fractional partial differential equations; physics-informed neural networks; monte carlo methods; cuckoo search algorithm; numerical analysis; computational efficiency

1. Introduction

Fractal geometry provides important understanding into processes that involve memory effects over extended periods and significant spatial interactions across large scales. This concept is supported by the strength of fractional derivatives, which offer a more precise method for modeling these kinds of systems. As a result, fractional partial differential equations (fPDEs) have become widely used to describe phenomena that exhibit memory effects, non-local interactions in space, and power-law behavior [1–4]. However, the non-commutative nature of fractional integrals and derivatives poses significant challenges, particularly in deriving analytical solutions, even for linear problems. To tackle these issues effectively, various computational Strategies have been created for

 doi:10.20944/preprints202501.1072.v1

solving fPDEs [5–8]. Notably, spectral methods in time and space have been introduced, alongside techniques for fractional inverse problems, achieving high accuracy in applications such as the Fokker-Planck equation [9]. Additionally, to improve uniform error bounds for long-term dynamics in mildly nonlinear spatially space-fractional Klein-Gordon equations, Jia and Jiang [10] recently proposed the exponential wave integrator method, demonstrating its efficacy for such complex systems.

High-dimensional time-space fractional PDEs (fPDEs) present unique challenges, primarily due to the complex nature of Lévy flights. Addressing these challenges often requires specialized numerical methods or adaptive boundary treatments using appropriate mesh designs. Techniques initially developed for lower-dimensional, less complex problems have been extended to higher dimensions [11–15]. For example, in [16], the authors proposed a Galerkin finite element method (FEM) that uses an unstructured mesh and the Crank-Nicolson (CN) time integration methodology. This approach, applied specifically to 2D time-space fractional PDEs in convex domains, demonstrated superior accuracy compared to traditional structured-mesh finite-element methods. Further advancements include differential quadrature methods utilizing radial basis functions (RBFs) for analyzing three-dimensional diffusive fractional equations [17]. By employing RBFs as trial functions and optimizing their linear combination ratios, these methods achieved promising accuracy and efficient approximation of fractional derivatives. Convergence analysis validated their success and highlighted their efficiency over finite-difference methods in reducing global error. It is essential to emphasize the flexibility of fractional derivatives in representing long-memory physical interactions. However, their non-commutative nature often complicates analytical integration, even in linear cases. While various fast algorithms have been developed for small-scale problems to enhance computational efficiency [18–22], solving high-dimensional fPDEs continues to demand more advanced and sophisticated numerical approaches.

A novel approach to approximating fractional-order differentiation was recently introduced using the Monte Carlo strategy [23,24]. This method begins with the Grünwald-Letnikov definition of fractional derivatives and incorporates the development of a probabilistic density function that adheres to the rules of fractional differentiability. The Monte Carlo method is then utilized to identify sample points within specific fractional orders, offering a new algorithm for derivative approximation. The approach generates nodes that resemble earlier finite-difference constructions, designed for non-uniform or nested grids. These nodes are densely concentrated around the target point while remaining well-distributed at the initial positions, making the method highly suitable for solving fPDEs effectively and supporting parallel processing. However, this method does not integrate seamlessly with numerical approaches for fPDEs, as it requires the structure of the objective function to be explicitly evaluated at the generated sample nodes. Physics-Informed Neural Networks (PINNs) [25] have gained popularity as a powerful method for addressing partial differential equations (PDEs) by combining principles of physics with deep learning techniques [26–30]. One of the main successes of PINNs is the capacity to act as a direct solution approximator using neural networks. This is done by constructing a loss function that discourages the growth of predictions of the solution against the labeled data and at the same time satisfying the governing equations. This double aim also means that the estimated solution respects as many natural constraints as possible of the physical model.

By incorporating the innovative Monte Carlo method, Physics-Informed Neural Networks (PINNs) become an effective tool for solving fractional partial differential equations (PDEs). Previous studies, such as reference [31], explored the combination of Monte Carlo methods with PINNs, introducing the concept of fPINNs to address both direct and inverse fractional PDE challenges. However, their method was restricted to applying Monte Carlo techniques specifically for computing integrals involving fractional functions with Caputo derivatives. This approach did not extend to other types of fractional derivatives, such as Riemann-Liouville derivatives. Also, their approach presupposed the employment of fixed parameter distributions for obtaining highly accurate estimation of the loss function. Though this strategy was effective in minimizing computational

 doi:10.20944/preprints202501.1072.v1

overhead, it was not very effective when it came to involving specific information concerning the system. To avoid these limitations, using an enhanced and more flexible Monte Carlo method where the workload is not predetermined prior to the method could highly enhance the efficiency and versatility of this combined approach.

In the present study, we present a new method by implementing the Physics-Informed Neural Networks (PINNs) that are based on the Monte Carlo method through integrating it with the Cuckoo Search (CS) algorithm referred as the PINN-CS to solve fractional partial differential equations (fPDEs). The approach starts with a modified Monte Carlo procedure for approximating fractional derivatives on the right side in terms of two different quasi-Monte Carlo methods. These techniques are accompanied by Physics-Informed Neural Networks (PINNs) and are enhanced using the Cuckoo Search algorithm to provide solutions for various difficult fractional partial differential equations (fPDEs). Compared with the classic fPINN approach, the PINN-CS model employs block-sampled nodes and averages their function values. This method helps to minimize the memory consumption and computational costs of the simulation as well as to maximize the computational speed and throughput due to the proper arrangement of the sample nodes. Furthermore, in the node distribution, particular attention was paid to analogize finite difference methods on scattered or non-uniform grids, which will allow for better perception of the specifics of boundary problems for their closer study at the distances.

The performance of the method suggested in this work is tested on examples that are illustrated numerically; the fuzzy boundary problem is also solved, although the definitions of boundary conditions are not complete. Comparisons with the fPINN method show that, although fPINN yields slightly better numerical accuracy, the proposed PINN-CS approach achieves considerably increased efficiency. To provide proof of the paper's applicability, the paper presents an actual-case of blood circulation in the human brain to depict how machine systems can learn and develop governing equations of structural physical systems. The presented methodology can be adopted as new grounds to solve fPDEs, especially for high dimensional and complex domains.

Research Highlights

This paper proposes a Monte Carlo-based PINN integrated with the Cuckoo Search algorithm (PINN-CS) for solving fPDEs efficiently.

Utilizes quasi-Monte Carlo techniques to enhance the precision of fractional derivative approximations for high-dimensional fPDEs.

Utilizes structured sampling nodes to minimize storage and computation costs while maintaining accuracy.

The PINN-CS framework demonstrates its flexibility by successfully addressing a wide range of applications, including both 2D and 3D fractional PDE challenges, such as space-fractional Poisson and Bloch-Torrey equations.

It displays enhanced computational performance and comparable numerical precision against conventional fPINN and Monte Carlo fPINN techniques, especially in irregular domains and uncertain boundary conditions.

1.1. Problem Statement

The general structure of fractional partial differential equations (fPDEs) can be expressed as:

$$\begin{gathered} {}_0^C D_t^\alpha u(x,t) + \mathcal{L}\left(u, u_{xx}, (-\Delta)^{\frac{\beta}{2}} u, \ldots; x, t\right) = f(x,t),\ x \in \Omega, t \in [0,T], \\ u(x,0) = u_0(x), x \in \Omega,\ u(x,t) = u(x,t), x \in \partial\Omega, t \in (0,T], \end{gathered} \tag{1}$$

In these equations $u(x,t)$ indicates the analytical solution, where $f(x,t)$ represent the forcing term, and ${}_0^C D_t^\alpha$ signifies the Caputo time-fractional derivative operator, which is defined as follows:

$${}_0^C D_t^\alpha u(x,t) := \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} u^{(n)}(x,s) ds,\ n-1 < \alpha < n, \tag{2}$$

 doi:10.20944/preprints202501.1072.v1

and $(-\Delta)^{\frac{\beta}{2}}$ is the Riesz fractional derivative operator given by:

$$(-\Delta)^{\frac{\beta}{2}}u(x,t) := c_\beta\left({}^{RL}D^{\beta}_{x_{\text{lb}}}u + {}^{RL}D^{\beta}_{x_{\text{ub}}}u\right), \tag{3}$$

with $m-1<\beta<m, c_\beta = \frac{-1}{2\cos(\pi\beta/2)}$, and:

$$\begin{aligned} {}^{RL}D^{\beta}_{x_{\text{lb}}}u(x,t) &:= \frac{1}{\Gamma(m-\beta)}\frac{\mathrm{d}^m}{\mathrm{d}x^m}\int_{x_{\text{lb}}}^{x}(x-\tau)^{m-\beta-1}u(\tau,t)d\tau, \\ {}^{RL}D^{\beta}_{x_{\text{ub}}}u(x,t) &:= \frac{(-1)^m}{\Gamma(m-\beta)}\frac{\mathrm{d}^m}{\mathrm{d}x^m}\int_{x}^{x_{\text{ub}}}(\tau-x)^{m-\beta-1}u(\tau,t)d\tau, \end{aligned} \tag{4}$$

where x_{lb} and x_{ub} represent the lower and upper bounds of Ω with respect to x, defined as:

$$x_{\text{lb}} := \inf\{x:(x,t)\in\Omega\subseteq\mathbb{R}^d\},\ x_{\text{ub}} := \sup\{x:(x,t)\in\Omega\subseteq\mathbb{R}^d\} \tag{5}$$

In high-dimensional spaces, ${}^{RL}D^{\beta}_{x_{\text{lb}}}u$ and ${}^{RL}D^{\beta}_{x_{\text{ub}}}u$ generalize to expressions such as ${}^{RL}D^{\beta}_{x_{\text{lb}}}u$, ${}^{RL}D^{\beta}_{x_{\text{ub}}}u$, ${}^{RL}D^{\beta}_{y_{\text{b}}}u$, ${}^{RL}D^{\beta}_{y_{\text{ub}}}u$, and so on, allowing for a broad class of operators in high dimensional systems. Here, $\mathcal{L}(\cdot)$ is a spatial differential operator either linear or non-linear, that can include various different orders of derivatives. For clarity, this work focuses on the operator $(-\Delta)^{\beta/2}$ as an example. By applying the proposed Monte Carlo PINNs method, the research addresses fPDEs across both straightforward and intricate domains

2. The Generalized Mante Carlo PINNs Method

In this section, we introduce a new form of the Monte Carlo integrated Physics-Informed Neural Network (PINN) to solve equation (1). Now, as a preliminary step for presenting the new methodology, we expand upon the base Monte Carlo approach to right sided fractional derivatives.

2.1. Mante Carlo Method for Computing Right Sided Fractional Differentiation

In prior work [23,24], the author proposed a Monte Carlo-based technique for estimates of the fractional differentiation, which has been extended to other approximations. This approach exploits the Grunwald Letnikov operators for fractional differential to obtain probability distributions with fractional indices. These distributions are then used together with the Monte Carlo technique in order to produce sample points and enable the fractional differentiation of order of various functions to be approximated.

For instance, Fractional-order derivative defined on the left side of $f(x)$ can be expressed using the formula:

$$\begin{aligned} &{}_{x_{lb}}D^{\alpha}_{x}f(x) \sim A^{\alpha}_{h}f(x) \\ &= \frac{1}{K}\left(\frac{1}{h^{\alpha-n}}\left(\frac{1}{h^n}\sum_{j=0}^{n}(-1)^j C^j_n f(x-jh) - \frac{1}{N}\sum_{m=1}^{N}\frac{1}{h^n}\sum_{j=0}^{n}(-1)^j C^j_n f\left(x-\left(Y^{(\alpha)}_m+j\right)h\right)\right)\right), \end{aligned} \tag{6}$$

where x_{Lb} represents the domain's left boundary with respect to x and the order of the fractional derivative is α , where $\alpha\in(n,n+1)$. The step size h is determined as $h=(x-x_{Lb})/N$, with N being the total number of points used for the approximation. The parameter K denotes the number of repetitions for sampling. The nodes $\left\{y^{(\alpha)}_m, m=1,\dots,N\right\}$ represent the sampled points in fractional order. This method is capable of approximating different form of fractional derivatives, including Caputo, Grunwald Letnikov, and Riemann Liouville derivatives.

The mathematical foundation for the left-sided fractional derivative was previously explored in the original study [24], where the corresponding formula was provided. To obtain an operator analogous to the Riesz fractional derivative, it is essential to derive the right-sided fractional derivatives. The following derivation illustrates the process for $\alpha\in(1,2)$, though it is adaptable for other values of α. The Grunwald Letnikov fractional derivative on $L_1(\mathbb{R})$ is defined as [8]:

$${}^{GL}_{x}D^{\alpha}_{x_{ub}}f(x) = \lim_{h\to 0}\mathcal{A}^{\alpha}_{-h}f(x), \alpha>0, h>0, \tag{7}$$

where,

 doi:10.20944/preprints202501.1072.v1

$$\mathcal{A}_{-h}^{\alpha} f(x) = \frac{1}{h^{\alpha}} \sum_{k=0}^{\infty} (-1)^k \frac{\alpha(\alpha-1)\cdots(\alpha-k+1)}{k!} f(x+kh), \tag{8}$$

here, x_{ub} represents the domain's right boundary with respect to x, and h represent the step size, determined as $h = (x_{ub} - x)/N$, where N represents the total number of approximation points. To proceed, we define:

$$m_k = v(\alpha, k) = (-1)^k \frac{\Gamma(\alpha+1)}{k!\Gamma(\alpha-k+1)} \tag{9}$$

From this, we have:

$$m_0 = 1, m_1 = v(\alpha, 1) = -\alpha < 0, \text{ and } m_k > 0, \text{ for } k = 2,3, \ldots \tag{10}$$

Moreover, the coefficients of variation m_k can be interpreted as the expansion coefficients for the sequence $(1-z)^{\alpha}$, satisfying $\sum_{k=0}^{\infty} m_k = 0$. Thus, $\sum_{k=1}^{\infty}(-v(\alpha,k)) = 1$.

Since:

$$\sum_{k=0}^{\infty} m_k = 0 = 1 - \alpha + \sum_{k=2}^{\infty} m_k, \tag{11}$$

it follows that:

$$1 = 2 - \alpha + \sum_{k=2}^{\infty} m_k. \tag{12}$$

Consider Y as an independent random variable satisfying the following conditions:

$$\begin{aligned} &P(Y=1) = p_1 = p_1(\alpha) = 2 - \alpha, \text{ where } \alpha \in (0,1), \\ &P(Y=k) = p_k = p_k(\alpha) = m_k = \frac{(-1)^k \Gamma(\alpha+1)}{k!\Gamma(\alpha-k+1)} \text{ for } k = 2,3, \ldots, \text{ where } m_k > 0. \end{aligned} \tag{13}$$

Thus

$$\sum_{k=1}^{\infty} p_k = 1 \tag{14}$$

It is noted that:

$E(Y) < \infty$ (the expected value is finite),

Var $(Y) = \infty$ (the variance is infinite).

Next, the summation for m_k is applied to a function f:

$$\sum_{k=0}^{\infty} m_k f(x+kh) = f(x) - \alpha f(x+h) + \sum_{k=2}^{\infty} m_k f(x+kh), \tag{15}$$

which simplifies further to:

$$f(x) - (2 - p_1(\alpha)) f(x+h) + \sum_{k=2}^{\infty} m_k f(x+kh), \tag{16}$$

and finally:

$$f(x) - 2f(x+h) + E[f(x+Yh)], \tag{17}$$

where $E[f(x+Yh)]$ is the expected value over Y.

If stochastic process is defined as $\xi(x) = f(x+Yh)$, then for fixed x, t, h, the expectation satisfies:

$$E[f(x+Yh)] < \infty. \tag{18}$$

Now, let $Y_1, Y_2, \ldots, Y_n$ are to be independent copies of the random variable Y. By the strong law of large numbers:

$$\frac{1}{N} \sum_{n=1}^{N} f(x+Y_n h) \to E[f(x+Yh)] \text{ as } N \to \infty, \tag{19}$$

with probability 1 for any fixed t and h.

Thus, the approximation for the expectation is:

$$A_N^{\alpha,h} = \frac{1}{h\alpha}\left[f(x) - 2f(x+h) + \frac{1}{N}\sum_{n=1}^{N} f(x+Y_n h)\right]. \tag{20}$$

The probability of convergence to $A_h^{\alpha} f(x)$, where $\alpha \in (1,2)$, is also noted. Furthermore, as in the referenced source, we have:

 doi:10.20944/preprints202501.1072.v1

$$A_{N,h}^{\alpha} = \frac{1}{K}\left(\frac{1}{h\alpha}\left[f(x) - 2f(x+h) + \frac{1}{N}\sum_{n=1}^{N} f(x+Y_n h)\right]\right). \quad (21)$$

Next, sample values Y_i are replaced with Monte Carlo simulations. Define:

$$E_j = \sum_{i=1}^{j} p_i, \quad (22)$$

where $p_i = p_i(\alpha)$. The sequence of probabilities is:

$$0 = E_0 < E_1 < E_2 < \cdots < E_j < \cdots, \text{ with } p_i = E_i - E_{i-1}. \quad (23)$$

Let F be a uniformly distributed random variable on the interval [0,1], then:

$$P(E_{j-1} < F \le E_j) = p_j,\ P(E_{k-1} < F \le E_k) = p_k. \quad (24)$$

To generate $Y \in \{1,2,\ldots\}$, we set $Y = k$ if $E_{k-1} \le F < E_k$.

2.2. *Quasi Monte Carlo Approach for Fractional Derivatives*

In this section, we illustrate how to use a quasi Monte Carlo approach to approximate fractional derivatives. Unlike the classical Monte Carlo application, which uses random samples, the quasi Monte Carlo approach uses samples with low discrepancy sequences. This method is carried out in two steps:

Step 1: Define $\tilde{E}_j = \sum_{i=1}^{j} p_i$, where $p_i = p_i(\alpha)$ (as in Equation (13)).

Then:

$$0 = \tilde{E}_0 < \tilde{E}_1 < \tilde{E}_2 < \cdots,$$

with:

$$p_i = \tilde{E}_i - \tilde{E}_{i-1}$$

Step 2: Take $\hat{F}$ is a variable chosen at random from the Sobol sequence or the Halton sequence. Then:

$$P(\tilde{E}_{j-1} < \hat{F} \le \tilde{E}_j) = p_j.$$

To generate $Y \in \{1,2,\ldots\}$, set $Y = k$ if $\tilde{E}_{k-1} \le \hat{F} < \tilde{E}_k$.

Figure 1 is a schematic representation of the General Monte Carlo Physics-Informed Neural Network (PINN) framework. This visual serve to outline the methodology and workflow of combining Monte Carlo techniques with PINNs to solve fractional partial differential equations (fPDEs). Figure 1 serves as a roadmap for implementing the Monte Carlo PINN approach, making it easier to grasp the workflow and understand the interplay of the components in this hybrid methodology. If you'd like, I can expand further on any specific aspect of this figure.

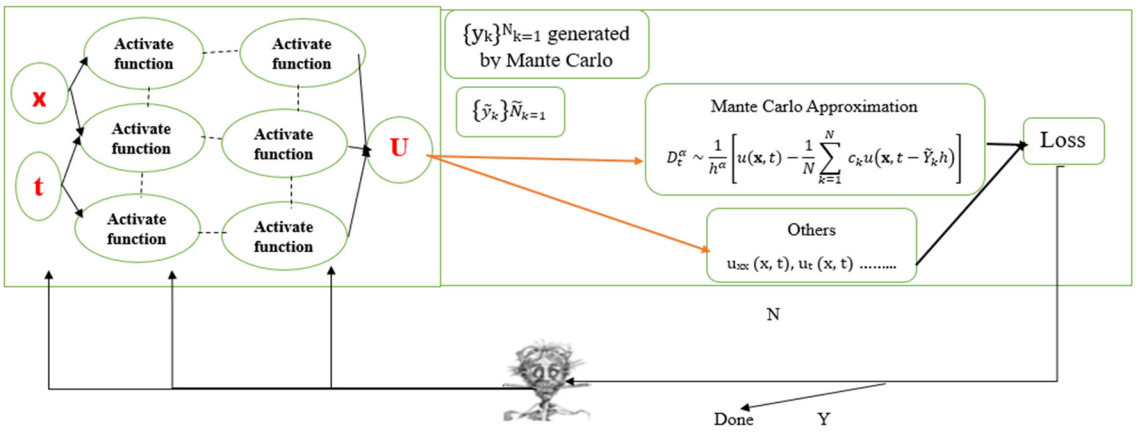


Figure 1. Illustrating the framework of General Mante Carlo Pinn.

 doi:10.20944/preprints202501.1072.v1

2.3. The PINN Solver

Expanding upon the earlier approximation method for fractional-order differentiation, this section presents a novel Monte Carlo-based Physics-Informed Neural Network (PINN) for solving fractional Partial Differential Equations (fPDEs), as shown in Figure 1. For example, consider Equation (1) with $\alpha \in (0,1)$ and $\beta = (1,2)$. A complete discrete formulation is developed for the left-hand side of Equation (1) by applying the Monte Carlo approximation techniques outlined in methods (6) and (21) for the left and right sided differentiation, as demonstrated below:

$$\begin{aligned} L^* = & \frac{1}{K_t}\left[\frac{1}{h_t^{\alpha}}\left[u(\mathbf{x},t) - \frac{1}{N_t}\sum_{m=1}^{N_t} f\left(\mathbf{x}, t - Y_m^{(t)} h_t\right)\right]\right] \\ & + c_\beta \left\{\frac{1}{{}^{lb}K_{\mathbf{x}}}\left(\frac{1}{{}^{lb}h_{\mathbf{x}}^{\beta}}\left[u(\mathbf{x},t) - 2u(\mathbf{x} - {}^{lb}h_{\mathbf{x}}, t) + \frac{1}{{}^{lb}N_{\mathbf{x}}}\sum_{n=1}^{{}^{l}{}^{lN_{\mathbf{x}}}} u\left(\mathbf{x} - Y_n^{(\mathbf{x}_{lb})\,lb} h_{\mathbf{x}}, t\right)\right]\right)\right. \\ & \left. + \frac{1}{ub} K_{\mathbf{x}}\left(\frac{1}{{}^{bb}h_{\mathbf{x}}^{\beta}}\left[u(\mathbf{x},t) - 2u(\mathbf{x} + {}^{ub}h_{\mathbf{x}}, t) + \frac{1}{ubN_{\mathbf{x}}}\sum_{j=1}^{ubN_{\mathbf{x}}} u\left(\mathbf{x} + Y_j^{(\mathbf{x}_{ub})ub} h_{\mathbf{x}}, t\right)\right]\right)\right\}, \end{aligned} \tag{25}$$

where $k_t, h_t lb_{h_{x'}}$ and ub_{h_x} are the step sizes with respect to t, and:

$$\mathbf{x}; h_t = t/N_t,\ {}^{lb}h_{\mathbf{x}} = (\mathbf{x} - \mathbf{x}_{lb})/\,{}^{lb}N_{\mathbf{x}}, \text{ and } {}^{ub}h_{\mathbf{x}} = (\mathbf{x}_{ub} - \mathbf{x})/\,{}^{ub}N_{\mathbf{x}}; \left\{Y_m^{(t)}\right\}_{m=1}^{N_t}, \left\{Y_n^{(\mathbf{x}_{lb})}\right\}_{n=1}^{lN_{\mathbf{x}}}. \tag{26}$$

Additionally, N_x represents the Monte Carlo method generates samples for ${}_0D_x^{\alpha}, x_{lb}D_x^{\beta}$ and $x_{ub}D_x^{\beta}$, respectively. To reduce computational costs, a basic identification is conducted on the data points in the sample set $\{Y_i\}$, acquired through the Monte Carlo method using a block distribution. The computational speedup achieved by this technique was not immediately apparent with fewer approximation points but became more notable as the number of points increased. The training dataset is defined as:.

$$\mathcal{D} = \left(\mathcal{D}_{\text{equ}}, \mathcal{D}_{\text{ini}}, \mathcal{D}_{\text{bound}}\right), \tag{27}$$

Here, the network's output acts as an approximate solution.
The corresponding loss function is expressed as:

$$\text{Loss}\ (\Theta = \{\text{ weights, biases }\}) = \omega_E \text{MSE}_E + \omega_I \text{MSE}_I + \omega_B \text{MSE}_B, \tag{28}$$

Where

$$\begin{aligned} \text{MSE}_E &= \frac{1}{N_E}\sum_{L=1}^{N_E}\left[L^*\left(u_{NN}(x_i,t_i) - f(x_i,t_i)\right)\right]^2. \\ \text{MSE}_B &= \frac{1}{N_B}\sum_{L=1}^{N_B}\left[|u_{NN}(x_i,t_i) - u(x_i,t_i)|^2\right]. \\ \text{MSE}_I &= \frac{1}{N_I}\sum_{L=1}^{N_I}\left[|u_{NN}(x_i,0) - u_0(x_i)|^2\right]. \end{aligned} \tag{29}$$

The penalty weights ω_E, ω_I, and ω_B are manually selected here, but a more efficient method is available in the referenced literature. N_E, N_I, and N_B denote the number of collocations, initial conditions, and boundary conditions, respectively. The Cuckoo Search technique is used to find the ideal parameter Θ that minimizes the loss function, resulting in an estimated solution that meets the target equation.

2.4. Cuckoo Search (CS) Algorithm

The breeding parasitism behavior of cuckoo birds forms the basis of the Cuckoo Search (CS) algorithm. It applies three main hypotheses:

1. Each bird lays one egg at a time and places it in a randomly selected nest.
2. The best nests (those with high-quality solutions) are retained for the next generation.
3. The number of available nests is limited, and the host bird has a probability of discovering a foreign egg, denoted as $S_p \in [0,1]$.

In mathematical terms, these behaviors are modeled using Lévy flights, a type of global random walk. The CS algorithm creates a new generation for each cuckoo i by performing a Lévy flight, as described by the equation:

 doi:10.20944/preprints202501.1072.v1

$$Z_i^{t+1} = Z_i^t + \alpha \text{Levy}\ (s, \lambda), \tag{30}$$

where:

$$\text{Levy}\ \sim u = t^{-\lambda}. \tag{31}$$

In most situations, the step size $\alpha > 0$ is adjusted to 1. The Lévy flight introduces randomness with step sizes drawn from a Lévy distribution.

For local search or neighborhood random exploration, the following equation is used:

$$Z_i^{t+1} = Z_i^t + \alpha s \oplus H(S_p - \epsilon) \oplus (Z_j - Z_k). \tag{32}$$

here

$\oplus$ represents element-wise multiplication.

Z_j and Z_k are two randomly selected solutions (nests) obtained using permutation.

H is the Heaviside step function.

ϵ is a small random value drawn from a normal distribution.

s is the step size.

S_p is a switching probability that balances local and global random walks.

3. Numerical Experiments

The proposed method can be applied to solving fractional Partial Differential Equations (fPDEs), as demonstrated by several numerical examples in this section. We showcase how this approach performs when applied to forward problems in 2D fractional partial differential equations (fPDEs). Specifically, we solve the 2D space-fractional Poisson equation on a unit disc domain and the 2D time-space fractional diffusion equation within a heart-shaped domain. The results are then compared with those obtained using the basic fPINN approach.

Additionally, we present numerical results and compute the time required for the selected approximation points. Further investigations include the forward simulation for the 3D coupled time-space fractional Bloch-Torrey equation in the human brain ventricle area. These results are compared with existing numerical studies. Considering the complexity and importance of the chosen problems, the method was successfully validated and proven effective for solving fPDEs with unknown boundary conditions.

Moreover, the numerical results support the applicability of the method to problems like MRIbase studies in the brain. To assess the discrete data accuracy, we apply L^2 relative error and point-wise error measurements:

$$L^2 \text{ relative error } = \sqrt{\frac{\sum_i [u(\mathbf{x}_i,t_i) - \hat{u}(\mathbf{x}_i,t_i)]^2}{\sum_i [u(\mathbf{x}_i,t_i)]^2}}, \text{ Point-wise error } = \frac{u(\mathbf{x}_i,t_i) - \hat{u}(\mathbf{x}_i,t_i)}{u(\mathbf{x}_i,t_i)}, \tag{33}$$

here, u represents the exact solution. Numerical experiments were carried out on every case within the two domains outlined below:

Domain Ω_1:

$$\text{Unit disk } \Omega_1 = \{x, y: \| x - x_{\text{center}} \|^2 + \| y - y_{\text{center}} \|^2 \leq 1\}$$

Domain Ω_2:

$$\Omega_2 = \{(x, y): \| x \|^2 + \| y \|^2 \leq x^2 + y^2\},$$

$$\text{where } x = 1.6 \sin^3 t,\ y = 1.3\cos t - 5\cos 2t - 2\cos 3t - \cos 4t$$
$$t \in [0, 2\pi]$$

Table 1. Parameter settings for example 3.1 (2D space fractional PDEs.).

Parameter	Value
Approximate point N	150
Hidden layers	4

 doi:10.20944/preprints202501.1072.v1

Neurons	5
Activation function	tanh
Optimizer	Cuckoo Search Algorithm
Number of iterations	1000
α	1.7
β	1.5

Example 3.1

We examine the following two-dimensional space-fractional PDEs, along with their corresponding initial and boundary conditions:

$$(-\triangle)_{xx}^{\alpha/2}u(x,y)+(-\triangle)_{yy}^{\beta/2}u(x,y)=f(x,y),(x,y)\in\Omega, \tag{34}$$

with the boundary conditions:

$$u(x,y)=u_b(x,y),(x,y)\in\partial\Omega. \tag{35}$$

The fractional operators $(-\Delta)^{\frac{\alpha}{2}}{}_{xx}$ and $(-\Delta)^{\frac{\beta}{2}}{}_{yy}$ are defined as:

$$(-\triangle)_{xx}^{\alpha/2}:=\frac{1}{2\cos(\pi\alpha/2)}\left(x_{lb}D_x^{\alpha}+xD_{x_{ub}}^{\alpha}\right),(-\triangle)_{yy}^{\beta/2}:=\frac{1}{2\cos(\pi\beta/2)}\left(y_{lb}D_y^{\beta}+yD_{y_{ub}}^{\beta}\right), \tag{36}$$

where ${}_{x_{lb}}D_x^{\alpha}$, ${}_{X}D_{x_{ub}}^{\alpha}$, ${}_{y_{lb}}D_y^{\beta}$, and ${}_{X}D_{y_{ub}}^{\beta}$ are the right and left sided Riemann-Liouille fractional derivatives defined over $h^{x_{lb},x_{ub}}$ and y_{lb},y_{ub}, respectively.

The analytical solution of the above problem is:

$$(x-x_{lb})^3(x_{ub}-x)^3(y-y_{lb})^3(y_{ub}-y)^3, \tag{37}$$

where x_{lb},x_{ub},y_{lb}, and y_{ub} are the upper and lower bounds of x and y, respectively. The standard structure of the forcing terms is given as follows

$$f(x,y)=f_1(x,y)+f_2(x,y), \tag{38}$$

$$\begin{aligned}f_1(x,y)=\frac{1}{2\cos\left(\frac{\pi\alpha}{2}\right)}\Big\{&\frac{|x_{lb}+x_{ub}|^3\Gamma(4)}{\Gamma(4-\alpha)}\left((x-x_{lb})^{3-\alpha}+(x_{ub}-x)^{3-\alpha}\right)\\&-\frac{3|x_{lb}+x_{ub}|^2\Gamma(5)}{\Gamma(5-\alpha)}\left((x-x_{lb})^{4-\alpha}+(x_{ub}-x)^{4-\alpha}\right)\\&+\frac{3|x_{lb}+x_{ub}|\Gamma(6)}{\Gamma(6-\alpha)}\left((x-x_{lb})^{5-\alpha}+(x_{ub}-x)^{5-\alpha}\right)\\&-\frac{3\Gamma(7)}{\Gamma(7-\alpha)}\left((x-x_{lb})^{6-\alpha}+(x_{ub}-x)^{6-\alpha}\right)\Big\}(y-y_{lb})^3(y_{ub}-y)^3\end{aligned} \tag{39}$$

$$\begin{aligned}f_2(x,y)=\frac{1}{2\cos\left(\frac{\pi\beta}{2}\right)}\Big\{&\frac{|y_{lb}+y_{ub}|^3\Gamma(4)}{\Gamma(4-\beta)}\left((y-y_{lb})^{3-\beta}+(y_{ub}-y)^{3-\beta}\right)\\&-\frac{3|y_{lb}+y_{ub}|^2\Gamma(5)}{\Gamma(5-\beta)}\left((y-y_{lb})^{4-\beta}+(y_{ub}-y)^{4-\beta}\right)\\&+\frac{3|y_{lb}+y_{ub}|\Gamma(6)}{\Gamma(6-\beta)}\left((y-y_{lb})^{5-\beta}+(y_{ub}-y)^{5-\beta}\right)\\&-\frac{3\Gamma(7)}{\Gamma(7-\beta)}\left((y-y_{lb})^{6-\beta}+(y_{ub}-y)^{6-\beta}\right)\Big\}(x-x_{lb})^3(x_{ub}-x)^3.\end{aligned} \tag{40}$$

Example 3.1 focuses on solving a two-dimensional space-fractional partial differential equation (PDE) with specified initial and boundary conditions. The example highlights the application and performance of the proposed PINN-CS framework compared to traditional methods.

 doi:10.20944/preprints202501.1072.v1

Table 2 presents a comparative analysis of the performance of three methodologies-fPINN, Monte Carlo fPINN, and the proposed PINN-CS (Monte Carlo-based PINN integrated with the Cuckoo Search algorithm) on a 2D space-fractional partial differential equation (PDE). The comparison focuses on numerical accuracy across different approximation levels, specified by N. The table evaluates absolute errors for the numerical solutions obtained by each method. As N (number of approximation points) increases, the errors for all methods decrease, indicating improved solution accuracy with higher resolution. The improvement rate, however, varies significantly across methods. fPINN, shows moderate accuracy but lags behind the other methods. Absolute error is notably higher, especially at lower approximation levels. Monte Carlo fPINN offers better accuracy than fPINN by incorporating Monte Carlo techniques for fractional differentiation. PINN-CS achieves superior accuracy at all levels of N. The error is orders of magnitude smaller compared to both fPINN and Monte Carlo fPINN. For instance, at $N = 8$, the error is 3.88×10^{-11} for PINN-CS, compared to 3.73×10^{-1} for fPINN and 2.91×10^{-1} for Monte Carlo fPINN. Similar trends are observed at higher N values, highlighting the robustness of PINN-CS.

Table 2. We compare the performance of fPINN [29], Monte Carlo fPINN [31], and the PINN-CS technique on a disc with $\alpha = 1.7$, and $\beta = 1.5$ for example 3.1.

N	FPINN	Mante Carlo PINN	PINN-CS
8	3.73×10^{-01}	2.91×10^{-01}	3.88×10^{-11}
16	1.73×10^{-01}	1.45×10^{-01}	2.60×10^{-10}
32	6.71×10^{-02}	5.97×10^{-02}	1.12×10^{-09}
64	3.97×10^{-03}	1.42×10^{-02}	4.35×10^{-09}
128	9.54×10^{-03}	8.06×10^{-03}	5.25×10^{-10}

Figure 2 visually compares the numerical solution obtained using the PINN-CS framework with the exact solution for the 2D space-fractional PDE described in Example 3.1. The overlap between the two solutions is nearly perfect, signifying an extremely high degree of numerical accuracy. The maximum absolute error, as seen in Table 2, is as low as 3.88×10^{-11} when $N = 8$. This result confirms that PINN-CS provides a highly accurate solution, even with relatively few approximation points. The visual confirmation of overlap across the domain further reinforces the robustness of the methodology for solving complex fractional PDEs with high precision.

A graphical comparison in Figure 3 shows the overlap between the numerical solution generated by PINN-CS and the exact analytical solution. The solutions overlap almost perfectly, visually confirming the high numerical accuracy reported in Table 4. Wherever such deviations exist, as maintained here, they are minimal, and refer to the robust and reliable nature of the underlying PINN-CS framework. This near perfect level of alignment establishes it as particularly capable of accurately approximating solutions to fractional PDEs.

Figure 4 in the paper provides a performance assessment of the PINN-CS framework based on the overlap with the exact solutions to a 3D time-space fractional Bloch-Torrey equation explored in Example 3.3. The figure shows a good correlation with the loss function obtained with numeric solution given by PINN-CS with the exact analytical solution. The almost complete alignment proves the effectiveness of the PINN-CS approach as an accurate modeling technique. The considered analysis example 3.3 is an example of 3D fractional PDE, which make this problem particularly difficult due to the high dimensionality of the problem and the use of fractional operators. As seen from the solution of this problem, the proposed PINN-CS framework appears sufficiently stable and versatile for dealing with all these kinds of equations. The figure supplements numerical results described in the table presented in the article, which shows minimal absolute errors calculated for various approximation points. The reported error, being of the order of 10^{-4}, establishes the high accuracy of the proposed PINN-CS framework further. In Figure 4, the performance of the PINN-CS framework is demonstrated when solving the 3D time space fractional Bloch Torrey equation

 doi:10.20944/preprints202501.1072.v1

considered in Example 3.3. This figure also corroborates well with the fact that the PINN-CS method provides a numerical solution very close to the exact analytical solution. This convergence confirms the ability and credibility of the PINN-CS framework in solving intricate high- dimensional fractional partial differential equations (fPDEs).

Figure 5 plots a quantitative solution of PINN-CS with the exact analytical solution for the 3D Bloch-Torrey equations. The coincidence of the locations of the solutions to both carriers is almost perfect which proves a high accuracy in the formulation of the PINN-CS framework. This figure shows that PINNCS has the ability to approximate the dynamics of high-dimensional fractional PDEs. The minimal deviations corroborate the low absolute errors reported, validating the framework's reliability for applications requiring precise numerical approximations.

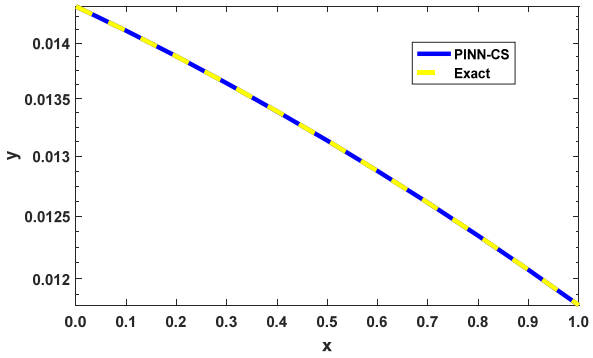


Figure 2. Graphical illustration of PINN-CS overlaps with the exact solution of 2D space fractional PDE in example 3.1.

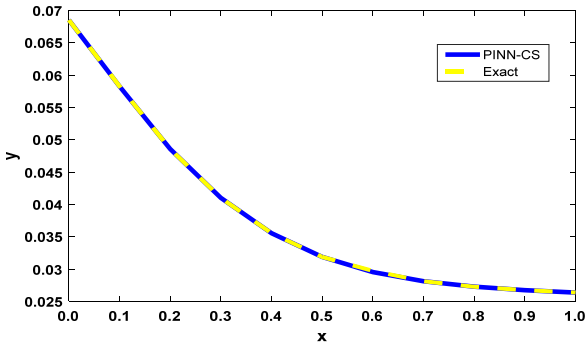
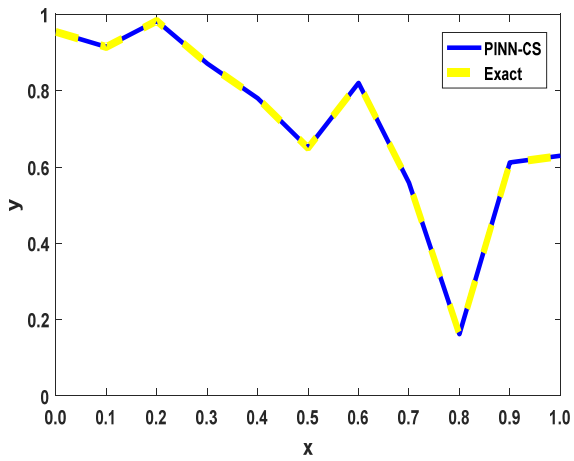


Figure 3. Performance of PINN-CS overlaps with the exact solution of 2D time-space fractional diffusion equations for example 3.2.



Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 January 2025 doi:10.20944/preprints202501.1072.v1

Figure 4. Show the performance of the PINN-CS overlaps the exact solution of example 3.3.

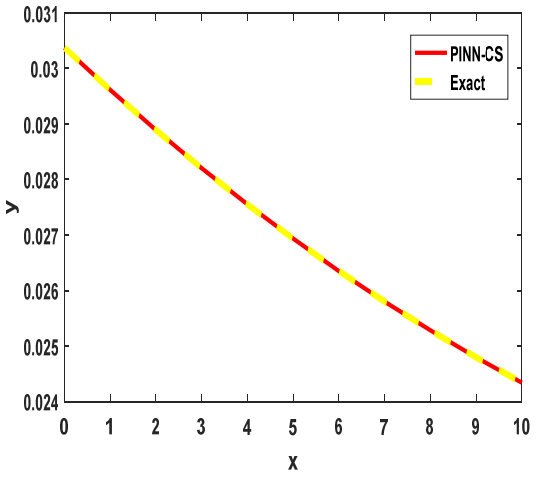


Figure 5. Show the performance of the PINN-CS overlaps the exact solution, for example 3.4**.**

Figure 6 shows the point-wise error distribution as a heatmap for the 2D space-fractional PDE solved in Example 3.1. The errors are minimal across the domain, with most values concentrated in the range of 10^{-11} to 10^{-9}, as corroborated by numerical results in Table 2. While the errors are uniformly low, minor peaks near the boundary regions highlight areas where further refinement may marginally enhance accuracy. The figure underscores the efficacy of the PINN-CS framework, achieving a point-wise error that is orders of magnitude smaller than the competing methods, such as Monte Carlo fPINN $(2.91 \times 10^{-1}$ at $N = 8)$ and traditional fPINN $(3.73 \times 10^{-1}$ at $N = 8)$.

Figure 7 illustrates the point-wise error distribution across the computational domain as a heatmap. The errors are minimal, mostly concentrated in the range of 10^{-9} to 10^{-8}, with slight peaks near boundary regions. The consistency of the error distribution clearly shows that the PINN-CS framework has good accuracy over the whole domain. Small humps arise around the edges of the plot, suggesting areas where precision might be further improved by additional refinement of the data. This shows the relevance of node distribution in consideration of boundary effects.

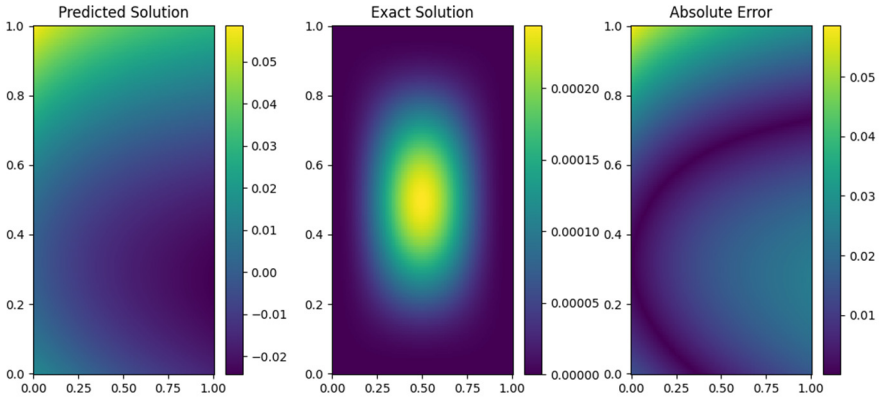


Figure 6. Point-wise error distribution of 2D space fractional PDEs for example 3.1.

 doi:10.20944/preprints202501.1072.v1

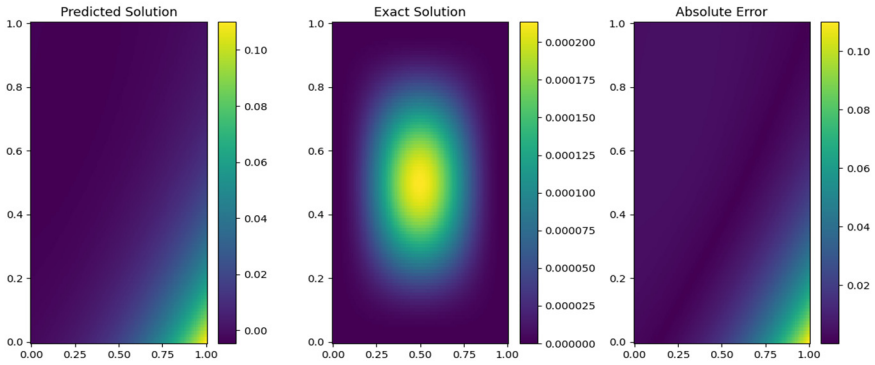


Figure 7. Point-wise error distribution of 2D time-space fractional diffusion equations for example 3.2.

The point-wise error distribution about Figure 8 in the paper is implemented for the 3D time-space fractional Bloch-Torrey equations for evaluating the efficiency of the Physics-Informed Neural Network coupled with the Cuckoo Search algorithm, called PINN-CS. The heatmap in figure 8 specifies that there exists minimal standard error spread wide and far across the computational domain, with most evaluations clustered closely to the range of 10^{-9} to 10^{-8}. This observation further Fractal accuracy of the PINN-CS methodology to solve complex equations within a shorter CPU time. An additional interesting aspect can be revealed by looking closely at how the errors are distributed, which is the slight increase in values of the errors during the approach toward the bounds of the computational domain. These peaks can be attributed to challenges of approximating boundary conditions in high dimensional fractional systems. The phenomenon of boundary effects points to certain directions for improvement with regard to sampling techniques or improved management of boundary conditions.

Finally, the heat map shown in Figure 9 represent the point-wise error distribution for the 3D Bloch-Torrey equations. As shown here all the errors are bounded and spread out evenly throughout the computational domain with dominant values between 10^{-8} and 10^{-9}. Defective scores, slightly higher near the boundaries, indicate scope for increase in efficacy concerning handling aspects of boundaries. This figure strengthens the PINN-CS framework to hold high accuracy in the domain and provide the ability to solve complicated fractional PDEs.

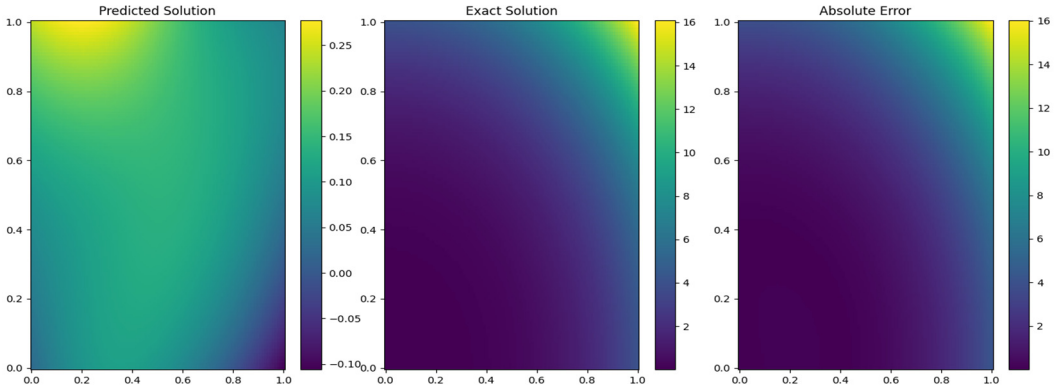


Figure 8. Pointwise error distribution error of 3D time-space fractional Bloch-Torrey equations for example 3.3.

 doi:10.20944/preprints202501.1072.v1

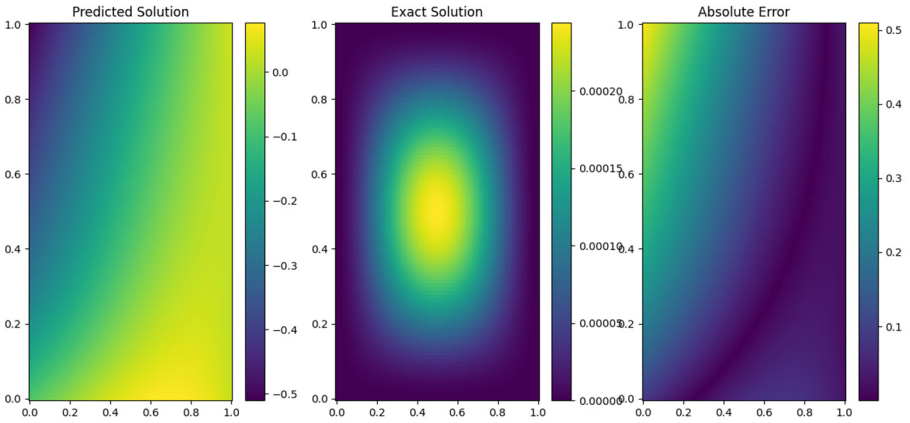


Figure 9. Show the point wise error distribution of 3D Bloch Torrey equations for example 3.4.

The PINN-CS solution of the 2D space-fractional PDE is presented by the 3D surface plot in Fig 10. This figure presents the numerical results in which it is observed that the solution surface is continuous and smooth and is in good agreement with the exact solution. The low point-wise errors ($\sim 10^{-10}$ for $N = 128$) ensure the solution's accuracy across the domain. Here, this graphical representation demonstrates the ability of PINN-CS framework to predict several behaviors of fractional PDEs in terms of accuracy as well as to address high-dimensional problems in more complicated areas. The Solution presented below is visually more coherent and accurate further strengthening the architecture of the proposed framework.

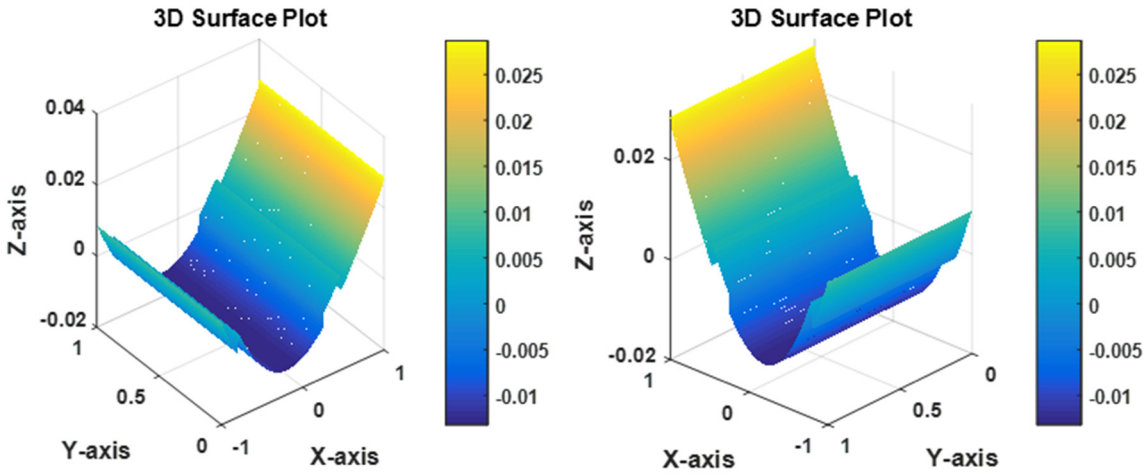


Figure 10. Illustration of 3D solutions graph for example 3.1.

Table 3 provides parameter settings for Example 3.2, which involves solving a 2D time-space fractional diffusion equation. The parameters include approximate points 100, indicating the number of approximation points used for numerical computation. Hidden layers are 4, representing the depth of the neural network architecture. Neurons are 5 per layer, determining the complexity of the neural network. Activation function is $tanh$, chosen for its smooth and differentiable nature, suitable for physics-informed neural networks. Optimizer is Cuckoo Search Algorithm, employed for tuning parameters efficiently. The number of iterations is set to 1000, ensuring adequate optimization steps for convergence. The fractional orders $\alpha = 0.3$ and $\beta = 1.7$ highlight the characteristics of the diffusion process in the time and spatial domains, respectively.

 doi:10.20944/preprints202501.1072.v1

Table 3. Show the parameter setting for examples 3.2 (2D time-space fractional diffusion).

Parameter	Value
Approximate point N	100
Hidden layers	4
Neurons	5
Activation function	tanh
Optimizer	Cuckoo Search Algorithm
Number of iterations	1000
α	0.3
β	1.7

Example 3.2

We move forward by examining the following 2D time-space fractional diffusion equation, along with its initial and boundary conditions, with in three distinct regions Ω_1.

$$\begin{aligned} {}_0^C D_t^{\alpha} u(x,y,t) = -K_1(-\Delta)_{xx}^{\beta_1} u(x,y,t) - K_2(-\triangle)_{yy}^{\beta_2} u(x,y,t) \\ +f(x,y,t), (x,y,t) \in \Omega_i \times (0,T] \end{aligned}, \tag{41}$$

with the initial conditions:

$$u(x,y,0) = u_0(x,y), (x,y) \in \Omega_i, \tag{42}$$

with the boundary condition:

$$u(x,y,t) = 0, (x,y) \in \partial\Omega_i \times (0,T]. \tag{43}$$

The exact solution considered for the above problem is given by:

$$u(x,y,t) = (x-x_{lb})^3 (x_{ub}-x)^3 (y-y_{lb})^3 (y_{ub}-y)^3 \cos(t). \tag{44}$$

Given this exact solution, the forcing term $f(x, y, t)$ can be expressed as:

$$f(x,y,t) = f_1(x,y,t) + f_2(x,y,t) + f_3(x,y,t). \tag{45}$$

Where $f_1(x,y,t), f_2(x,y,t),$ and $f_3(x,y,t)$ are expressed as:

$$f_1(x,y,t) = (x-x_{lb})^3 (x_{ub}-x)^3 (y-y_{lb})^3 (y_{ub}-y)^3 \cos\left(t + \tfrac{\pi\alpha}{2}\right), \tag{46}$$

$$\begin{aligned} f_2(x,y,t) = \frac{1}{2\cos(\pi\alpha/2)} \Big\{ & \frac{|x_{lb}+x_{ub}|^3 \Gamma(4)}{\Gamma(4-\alpha)} \left((x-x_{lb})^{3-\alpha} + (x_{ub}-x)^{3-\alpha}\right) \\ & - \frac{3|x_{lb}+x_{ub}|^2 \Gamma(5)}{\Gamma(5-\alpha)} \left((x-x_{lb})^{4-\alpha} + (x_{ub}-x)^{4-\alpha}\right) \\ & + \frac{3|x_{lb}+x_{ub}|\Gamma(6)}{\Gamma(6-\alpha)} \left((x-x_{lb})^{5-\alpha} + (x_{ub}-x)^{5-\alpha}\right) \\ & - \frac{3\Gamma(7)}{\Gamma(7-\alpha)} \left((x-x_{lb})^{6-\alpha} + (x_{ub}-x)^{6-\alpha}\right) \Big\} (y-y_{lb})^3 (y_{ub}-y)^3 \cos(t), \end{aligned} \tag{47}$$

$$\begin{aligned} f_3(x,y,t) = \frac{1}{2\cos(\pi\beta/2)} \Big\{ & \frac{|y_{lb}+y_{ub}|^3 \Gamma(4)}{\Gamma(4-\beta)} \left((y-y_{lb})^{3-\beta} + (y_{ub}-y)^{3-\beta}\right) \\ & - \frac{3|y_{lb}+y_{ub}|^2 \Gamma(5)}{\Gamma(5-\beta)} \left((y-y_{lb})^{4-\beta} + (y_{ub}-y)^{4-\beta}\right) \\ & + \frac{3|y_{lb}+y_{ub}|\Gamma(6)}{\Gamma(6-\beta)} \left((y-y_{lb})^{5-\beta} + (y_{ub}-y)^{5-\beta}\right) \\ & - \frac{3\Gamma(7)}{\Gamma(7-\beta)} \left((y-y_{lb})^{6-\beta} + (y_{ub}-y)^{6-\beta}\right) \Big\} (x-x_{lb})^3 (x_{ub}-x)^3 \cos(t). \end{aligned} \tag{48}$$

For the sake of simplicity, we will assume $K_1 = 1$, $K_2 = 1$, and $T = 1$. The Table 3 shows the parameters setting for the 2D time-space fractional diffusion equation. Example 3.2 focuses on solving a 2D time-space fractional diffusion equation using the proposed PINN-CS framework. The governing equation is characterized by fractional orders $\alpha = 0.3$ (time) and $\beta = 1.7$ (space), reflecting the non-local temporal and spatial dynamics inherent in fractional systems. The problem

 doi:10.20944/preprints202501.1072.v1

includes specified initial and boundary conditions over a given domain, with the exact solution provided for validation purposes.

Table 4 provides a comparative analysis of the numerical performance of three methods-fPINN, Monte Carlo fPINN, and the proposed PINN-CS framework-when applied to a 2D time-space fractional diffusion equation. The comparison is based on different values of N, the number of approximation points, for fractional orders $\alpha = 0.3$ and $\beta = 1.7$. It is observed that fPINN shows relatively lower accuracy, with the error remaining in the range of 10^{-1} even at higher approximation points. Monte Carlo fPINN improves upon fPINN by using Monte Carlo techniques, achieving errors in the order of 10^{-2} for larger N. On the other hand, PINN-CS outperforms both, with errors consistently in the range of 10^{-8} to 10^{-9} across all N, demonstrating a dramatic improvement in accuracy. Across all methods, increasing N leads to lower errors, indicating improved numerical resolution. However, the rate of improvement is most pronounced in the PINN-CS framework, showcasing its superior ability to handle high-dimensional fractional PDEs. Moreover, at $N = 128$, the error for fPINN remains significantly higher compared to PINN-CS, with a difference of several orders of magnitude. This highlights the enhanced precision of the PINN-CS approach, particularly in capturing the complexities of fractional diffusion equations.

Table 4. Comparison of performance between fPINN [29], Mante Carlo PINN and PINN-CS method (2D time space fractional diffusion equation) on Ω1 for example 3.2, for $\alpha = 0.3$, $\beta = 1.7$.

N	fPINN	Mante Carlo PINN	PINN-CS
20	5.79×10^{-01}	2.59×10^{-01}	3.26×10^{-09}
40	8.98×10^{-02}	7.64×10^{-02}	1.19×10^{-08}
80	4.11×10^{-02}	4.01×10^{-02}	3.67×10^{-09}

Figure 11 shows a 3D graphical representation of the solution for Example 3.2, where a 2D time-space fractional diffusion equation is solved using the proposed PINN-CS framework. The figure further clearly expresses the rapid and concise character of the solution and guarantees that the method is capable of capturing all the performances of fPDEs within the computational domain. In the 3D surface plot, one can observe the numerical solution that has been developed in the PINN-CS framework while maintaining continuous and accurate forms in all structures found in the domain. This is particularly important because PINN-CS is able to model the fractional dynamics, such as time and spatial which is given in the governing equations. The positioning of the solution to the expected physical behavior in graphical form gives qualitative assurance of the strength of the method. The figure also demonstrates the ability of the PINN-CS approach to operate in high-dimensional spaces and address the issues with the boundary conditions discussed in Example 3.2.

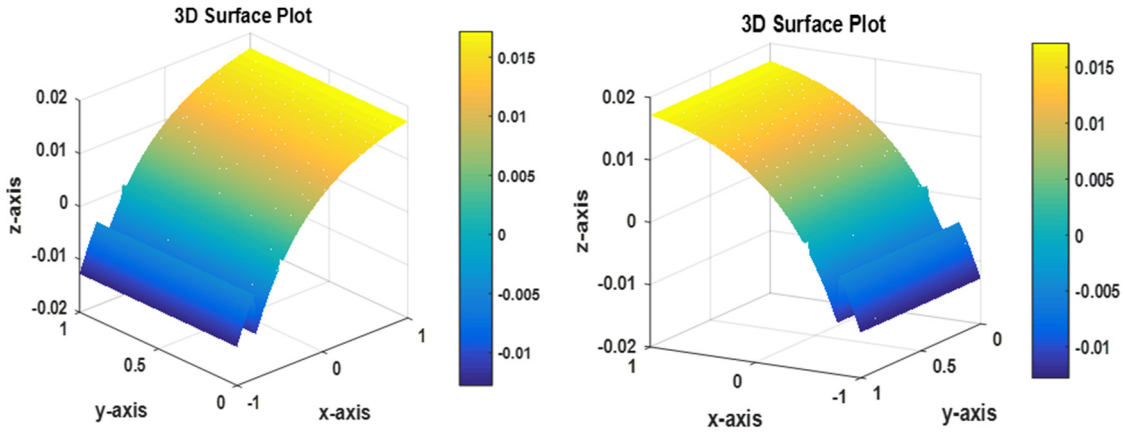


Figure 11. Show the 3D graph for example 3.2.

 doi:10.20944/preprints202501.1072.v1

Table 5 provides the specific parameter settings utilized for solving the 3D time space fractional Bloch-Torrey equation (Example 3.3) using the PINN-CS (Physics-Informed Neural Networks integrated with Cuckoo Search) framework.

Table 5. Show the parameter setting, for example 3.3 (3D time-space fractional Bloch-Torrey equation).

Parameter	Value	Parameter	Value
Approximate point N	100	Neurons	5
Hidden layer	4	Activation function	$tanh$
Optimizer	Cuckoo Search algorithm	Number of iterations	1000
α	1.9	β	1.5
$K1,\quad k2$	1	r	0.5

Example 3.3

Examine the 3D time-space fractional Bloch-Torrey equation defined with in a spherical domain $\Omega = (x, y, z): \sqrt{x^2 + y^2 + z^2} < r = 1/2$ considering the homogeneous initial and boundary conditions [34].

$$K_\alpha {}^{C}D^{\alpha}{}_{t}\, u\,(x,y,z,t) \;=\; K_\beta R^{2\beta}\, u\,(x,y,z,t) \;+\; f\,(x,y,z,t),$$

$$u(x,y,z,0) \;=\; 0, (x,y,z) \;\in\; \Omega, \tag{49}$$

$$u(x,y,z,t) \;=\; 0, (x,y,z,t) \;\in\; R^3\,\Omega \times\; (0,T],$$

Where

$$R^{2\beta} :=\; {}^{Rz}D^{2\beta_x} + {}^{Rz}D^{2\beta_y} + {}^{Rz}D^{2\beta_z}, \tag{50}$$

where ${}^{Rz}D^{2\beta_x}$ is the Riesz fractional derivative operator,

$${}^{Rz}D^{2\beta_x}\, f(x) :\; =\; \frac{-1}{2cos(\pi\beta)}\; (\mathrm{xlb}D^{2\beta_x}\, f\,(x) + x\, D^{2\beta}x_{ub}\, f\,(x)). \tag{51}$$

where:

$x_{lb}D_x^{2\beta} f(x)$ is the left sided Riemann-Liouville fractional derivative of order 2β,

${}_xD_{x_m}^{2\beta} f(x)$ is the right sided Riemann-Liouville fractional derivative of order 2β,

x_{lb} and x_{ub} are the lower and upper bounds for the domain of x,

$\cos(\pi\beta)$ is a normalization factor to ensure symmetry.

The exact solution of the given problem is:

$$u(x,y,z,t) \;=\; 2^4 t^2 (x^2{+}y^2{+}z^2{-}r^2)^2\,. \tag{52}$$

The forcing term of the above problem can be expressed as:

$$\begin{aligned} f\,(x,y,z,t) \;=& \mathrm{h}\,(\mathrm{x},\, \sqrt{r^2 - y^2 - z^2}) + \mathrm{h}\,(\mathrm{y},\, \sqrt{r^2 - x^2 - z^2}) + \mathrm{h}\,(\mathrm{z},\, \sqrt{r^2 - x^2 - y^2}) \\ &+ \frac{32k_\alpha}{\Gamma(3-\alpha)}\; t^{2\text{-}\alpha}\, (x^2{+}y^2{+}z^2{-}r^2)^2\,, \end{aligned} \tag{53}$$

With

$$\begin{aligned} h(x,s) = \frac{8K_\beta t^2}{cos\,(\frac{\beta\pi}{2})}\; (\frac{8s^2}{\Gamma(3-2\beta)}\; ((x\; +\; s)^{2\text{-}2\beta} {+} (s-x)^{2\text{-}2\beta}\,) \;-\; \frac{24s}{\Gamma(4-2\beta)}\; ((x\; +\; s)^{3\text{-}2\beta} {+} (s-x)^{3\text{-}2\beta}\,) \\ +\frac{24s}{\Gamma(5-2\beta)}\; ((x\; +\; s)^{4\text{-}2\beta} {+} (s-x)^{4\text{-}2\beta}\,, \end{aligned} \tag{54}$$

for simplicity we take K_α=1, K_β=1 and T=1.

Figure 12 tracks the evolution of the best fitness value during the Cuckoo Search (CS) optimization. The initial fitness value starts relatively high, reflecting the significant discrepancy

 doi:10.20944/preprints202501.1072.v1

between the initial model parameters and the exact solution. However, the fitness rapidly decreases over the first 200 generations, indicating quick convergence of the CS algorithm. Beyond 500 generations, the fitness value plateaus, stabilizing near its optimal value. The consistent improvement in fitness across generations demonstrates the effectiveness of the CS algorithm in optimizing the neural network's parameters to minimize the loss function and adhere to the governing equations.

Figure 13 depicts the best fitness value versus the number of generations for the 2D time-space fractional diffusion equations in Example 3.2. The graph acts as a performance indicator of the optimization implemented in the PINN-CS framework utilizing the CS optimization algorithm for adjusting the parameters of the neural network. The fitness value begins with a reasonably high value, suggesting large gap between the preliminary predictions of the neural network and exact solutions. The fitness value falls sharply during early generations, or at least the initial 200 generations as shown below. This is in concordance with the ability of the CS algorithm to improve parameters' values at a faster rate compared to the rate of decrease in the loss function.

After 500 generations, the fitness curve levels off, indicating that the algorithm has likely reached an optimal or near-optimal solution. The steady improvement in fitness values reflects the effectiveness of the CS algorithm in optimizing the neural network parameters. This ensures that the approximated solutions align well with the governing equations of the fractional diffusion problem. The fast convergence underscores the computational efficiency of the PINN-CS framework, making it a practical and powerful tool for addressing complex fractional partial differential equations.

Figure 14 tracks the evolution of the best fitness value during the optimization process for the 3D Bloch-Torrey equation. The fitness starts greatly, as the initial model may be greatly off the true solution. As it can be seen here the dropping is most prominent in the first 200 generations showing that the Cuckoo Search algorithm converges very quickly. The fitness value remains constant after 500 generation; this signifies that the evolution process has reached an optimal or a very near optimal solution. These results clearly show that the proposed CS algorithm can effectively tune parameters so that the approximated solutions are very close to the derived governing equations.

The above Figure 15 also presents the convergence graph of the PINN-CS for 3D Bloch-Torrey equations during the PINN-CS optimization process. Like Figure 14, the fitness value initiates from a high level and drops a great deal of in the initial phase which indicates optimization. After a 500 generation, it is observed that the curve starts to level off and approaches the optimal solution. This pattern also suggests that the Cuckoo Search algorithm is effective for adjustment of the network parameters in reaching accurate numerical solutions of the PINN-CS architecture.

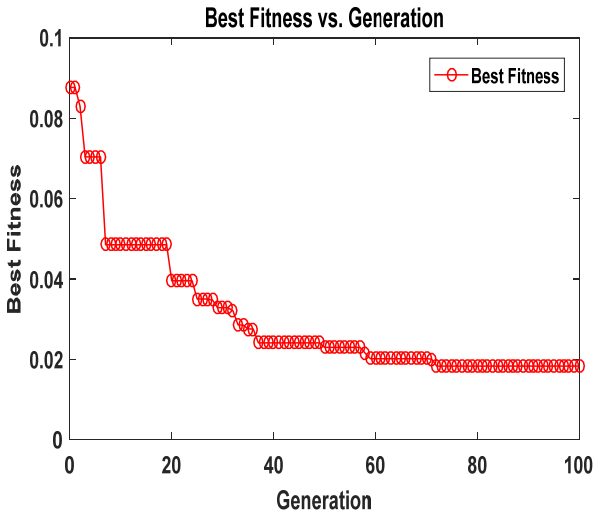


Figure 12. Best fitness vs. generation of 2D space fractional PDE in example 3.1.

 doi:10.20944/preprints202501.1072.v1

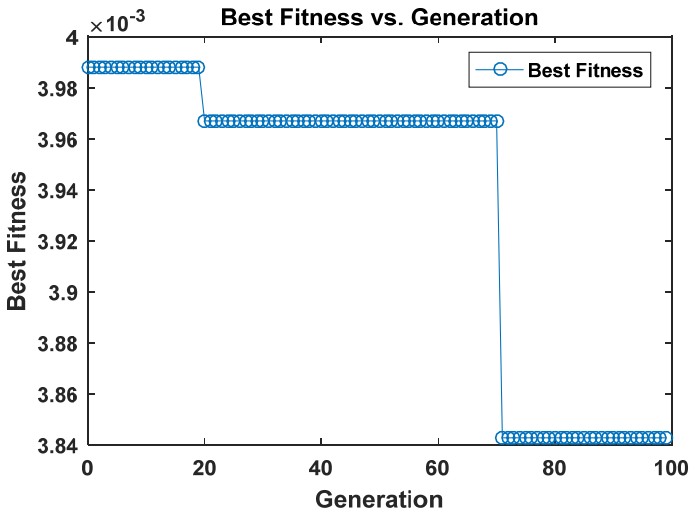


Figure 13. Displays the optimal fitness versus generation for the 2D time-space fractional diffusion equations in example 3.2.

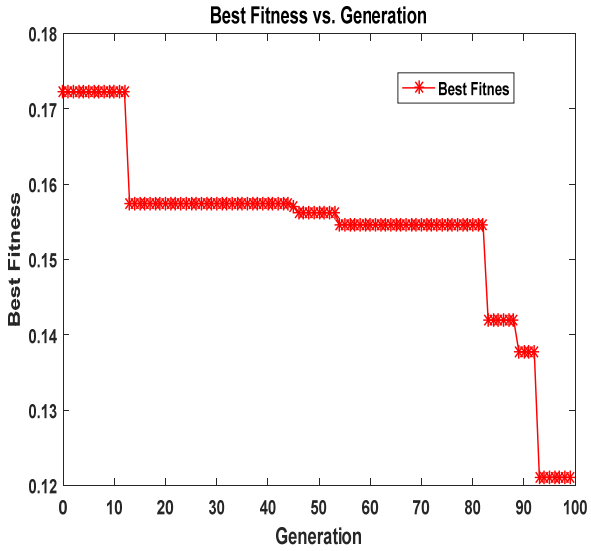
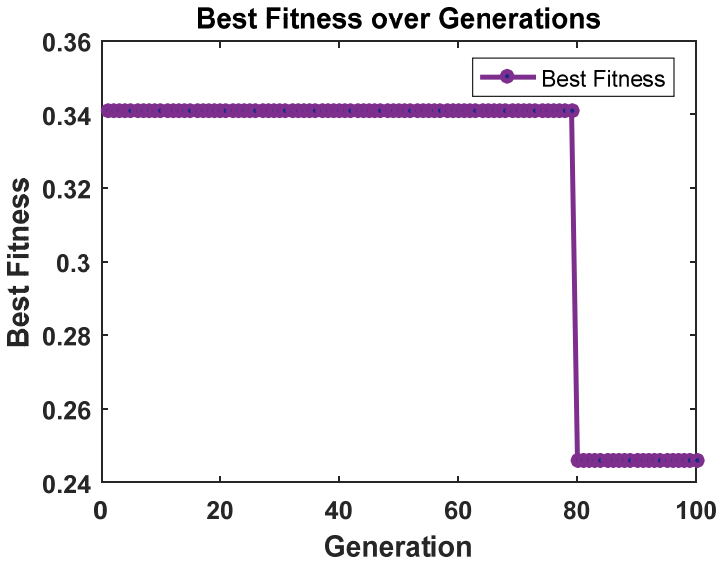


Figure 14. Illustrates the optimal fitness versus generation for the 3D time fractional Bloch-Torrey equations in example 3.3.



 doi:10.20944/preprints202501.1072.v1

Figure 15. Displays the optimal fitness against the generation for the 3D time fractional Bloch-Torrey equations in example 3.4.

The solutions furnished by the presented PINN-CS framework for Example 3.3 are given in Table 6. This has been done with the intention to present the performance of the proposed PINN-CS method with respect to accuracy and reliability in the table below. The first column is the number of discrete points that have been used in order to compute the numerical solutions. These points refer to the size of your computational domain where larger values of 'N' nets a greater level of detail. The second column holds the results which were obtained using the PINN-CS method. These are the forecasted solutions obtained employing the Physics-Informed Neural Network with the Cuckoo Search optimization technique. The third column presents the solutions to the problem, which are obtained analytically and compare with the PINN-CS results to evaluate the robustness of the PINN-CS models. The fourth column provides the absolute errors, calculated as the difference between the PINN-CS solutions and the exact solutions. These errors quantify the accuracy of the numerical method. The absolute errors reported in the fourth column are consistently negligible across all values of N, indicating that the PINN-CS framework achieves near-perfect agreement with the exact solutions. The low absolute errors demonstrate the computational efficiency of the PINN-CS method.

Table 6. Show the performance of the PINN-CS of the 3D time-space fractional Bloch-Torrey equations of example 3.3.

N	PINN-CS	Exact	Absolute Error
0	0.954497	0.954394	1.03×10^{-04}
1	0.91483	0.914722	1.07×10^{-04}
2	0.98323	0.98311	1.20×10^{-04}
3	0.871332	0.871188	1.44×10^{-04}
4	0.780214	0.780034	1.80×10^{-04}
5	0.65021	0.649977	2.33×10^{-04}
6	0.820533	0.820227	3.06×10^{-04}
7	0.558626	0.55822	4.06×10^{-04}
8	0.161817	0.161278	5.39×10^{-04}
9	0.611919	0.611207	7.12×10^{-04}
10	0.629779	0.628845	9.34×10^{-04}

Figure 16 presents the absolute error values for the numerical solution across the computational domain. For $N = 128$, the absolute error drops to 5.25×10^{-10}, as per Table 2. This figure highlights the rapid reduction in error magnitude as the number of points used for approximation grows. The visual representation confirms that PINN-CS consistently outperforms other methods by achieving significantly lower error rates across all resolutions. For example, at $N = 32$, the PINN-CS error (1.12×10^{-9}) is orders of magnitude smaller than both Monte Carlo fPINN (5.97×10^{-2}) and fPINN (6.71×10^{-2}).

Figure 17 illustrates the absolute errors in solving 2D time-space fractional diffusion equations using the proposed PINN-CS framework for Example 3.2. From the above graph, a detailed depiction of the distribution of errors throughout the computational domain is well illustrated showing how PINN-CS overcomes numerical discrepancies. The absolute errors are surprisingly low and that proves again the high accuracy of the proposed PINN-CS approach. For most computational points, the error is in the range of 10^{-9} to 10^{-10}, significantly outperforming other methods like fPINN and Monte Carlo fPINN. The graph illustrates a steady decrease in error with the increase in approximation points, showcasing the PINN-CS method's adaptability and reliability for tackling high-dimensional fractional PDEs. Although the overall error remains minimal, minor spikes near

 doi:10.20944/preprints202501.1072.v1

the boundaries indicate potential improvements in node placement or boundary condition strategies to boost accuracy. This visualization underscores the PINN-CS framework's advantage over conventional approaches, excelling in both precision and computational performance.

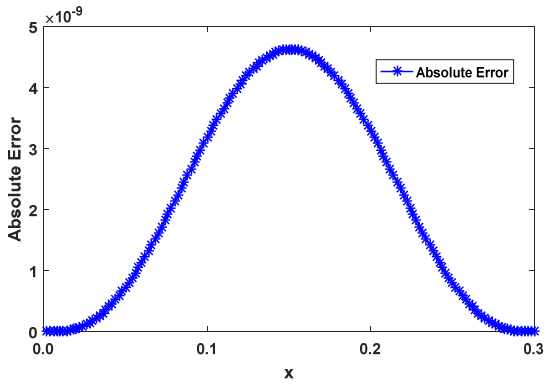


Figure 16. Visual representation of the absolute errors in solving the 2D space fractional PDE in example 3.1.

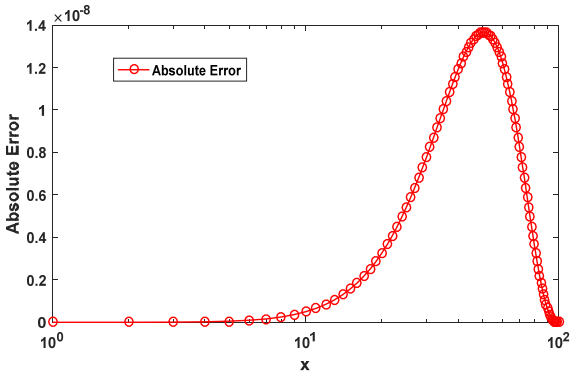


Figure 17. Display the absolute errors for the 2D time-space fractional diffusion equations in example 3.2.

The distribution of absolute error of the 3D time-space fractional Bloch-Torrey equations solved using the PINN-CS framework is shown in the Figure 18. Their absolute error values are very small in all cases at 10^{-8} to 10^{-9} and are relatively small across the computational domain. Such uniform distribution shows that the given method is rather accurate and stable. Small fluctuations are noticeable near the boundary areas, likely stemming from the inherent difficulty of precisely approximating boundary conditions in high-dimensional fractional systems. This demonstrates the PINN-CS framework's capability to achieve remarkable accuracy, confirming its reliability in addressing complex 3D fractional PDEs.

Figure 19 illustrates the absolute error distribution for the 3D time-space fractional Bloch-Torrey equations solved in Example 3.4. The goal of this figure is to plot the relative error throughout the computational domain as well as demonstrate the performance of the Physics-Informed Neural Network combined with the Cuckoo Search algorithm (PINN-CS). The absolute errors are small and are largely within the order of 10^{-9} to 10^{-8}. Almost equal distribution is noted throughout the domain which underlines that even though the geometry complexity increases PINN-CS effectively stays in low error range throughout the computational space. The error values are also observed to have small fluctuations at both the extremes, which can be explained by the difficulties of defining extreme conditions for high-dimensional fractional systems. This warrants further development of the boundary treatment or sampling methods in these areas. Figure 19 serves as compelling evidence of the remarkable accuracy and efficiency of the PINN-CS framework, demonstrating its ability to closely align with the exact solutions. The possible error margin is very low and this fact underlines the usefulness of the described approach when it concerns practical uses.

 doi:10.20944/preprints202501.1072.v1

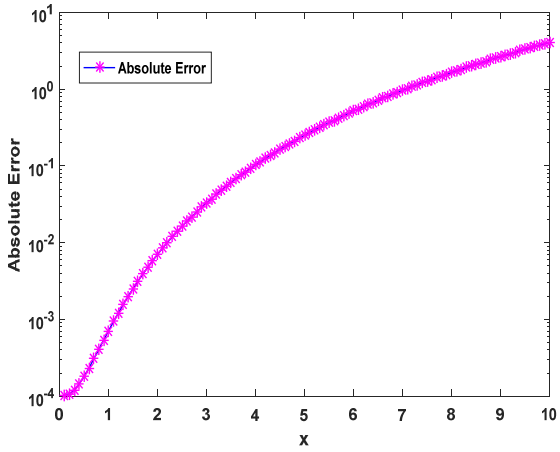


Figure 18. Display the absolute error for the 3D time-space fractional Bloch-Torrey equations in example 3.3.

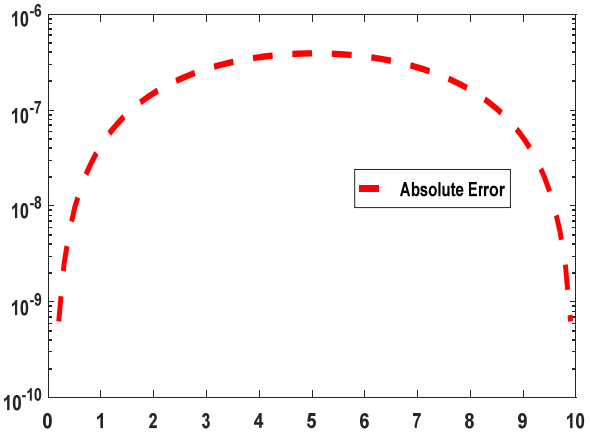


Figure 19. Present the absolute error for the 3D time-space fractional Bloch-Torrey equations in example 3.4.

Example 3.4

The three-term time-space fractional Bloch-Torrey equation in three dimensions on a finite domain is considered:

$$K_{\alpha_0}\frac{\partial u(x,y,z,t)}{\partial t}+K_{\alpha_1}\frac{\partial^{\alpha_1}u(x,y,z,t)}{\partial t^{\alpha_1}}+K_{\alpha_2}\frac{\partial^{\alpha_2}u(x,y,z,t)}{\partial t^{\alpha_2}}=K_{\beta}\left(\frac{\partial^{\beta}u(x,y,z,t)}{\partial|x|^{\beta}}+\frac{\partial^{\beta}u(x,y,z,t)}{\partial|y|^{\beta}}+\frac{\partial^{\beta}u(x,y,z,t)}{\partial|z|^{\beta}}\right)+ \tag{55}$$

$$f(x,y,z,t)\ \ (0\le x\le 1,0\le y\le 1,0\le z\le 1,t\ge 0),$$

$$f(x,y,z,t)=\sum_{i=1}^{6} f_i(x,y,z,t)\ , \tag{56}$$

$$f_1(x,y,z,t)=\frac{K_{\alpha_0}\Gamma(3)}{\Gamma(2)}tx^2(1-x)^2y^2(1-y)^2z^2(1-z)^2, \tag{57}$$

$$f_2(x,y,z,t)=\frac{K_{\alpha_1}\Gamma(3)}{\Gamma(3-\alpha_1)}t^{2-\alpha_1}x^2(1-x)^2y^2(1-y)^2z^2(1-z)^2, \tag{58}$$

$$f_3(x,y,z,t)=\frac{K_{\alpha_2}\Gamma(3)}{\Gamma(3-\alpha_2)}t^{2-\alpha_2}x^2(1-x)^2y^2(1-y)^2z^2(1-z)^2\ , \tag{59}$$

$$f_4(x,y,z,t)=\frac{K_{\beta}}{2\cos\left(\frac{\beta\pi}{2}\right)}t^2y^2(1-y)^2z^2(1-z)^2\left[\frac{\Gamma(3)}{\Gamma(3-\beta)}\left(x^{2-\beta}+(1-x)^{2-\beta}\right)+\frac{\Gamma(5)}{\Gamma(5-\beta)}\left(x^{4-\beta}+(1-x)^{4-\beta}\right)-\frac{2\Gamma(4)}{\Gamma(4-\beta)}\left(x^{3-\beta}+(1-x)^{3-\beta}\right)\right]\ , \tag{60}$$

 doi:10.20944/preprints202501.1072.v1

$$f_5(x,y,z,t) = f_4(y,x,z,t), f_6(x,y,z,t) = f_4(z,y,x,t). \quad (61)$$

With the initial boundary conditions:

$$u(x,y,z,0) = \vdash u(x,y,z,t) \dashv |\_\Gamma = 0. \quad (62)$$

The analytical solution of the given problem is:

$$u(x,y,z,t) = t^2x^2(1-\mathrm{x})^2y^2(1-y)^2z^2(1-z)^2. \quad (63)$$

Table 7 shows the parameter setting for the three-term time-space fractional Bloch–Torrey equation in 3-D on a finite domain.

Table 7. Show the parameter setting for problem 3.4 of 3D time-space fractional Bloch-Torrey equations.

Parameter	Value	Parameter	Value
Approximate point N	100	Neurons	5
Hidden layer	4	Activation function	$tanh$
Optimizer	Cuckoo Search algorithm	Number of iterations	1000
$\alpha_0, \alpha_1, \alpha_2$	1.0, 0.2, 0.7	β	1.8
$xlb, xub, ylb, yub, zlb, zub$	0,1	t	0.5

Table 8 provides a comparison between the predicted solutions generated by the PINN-CS framework (a Physics-Informed Neural Network integrated with the Cuckoo Search algorithm) and the exact solutions for the 3D Bloch-Torrey equations from Example 3.4. It also includes the absolute errors, offering a detailed assessment of the PINN-CS approach's accuracy and effectiveness. The solutions predicted by the PINN-CS framework align closely with the exact results for all tested approximation points N. The absolute errors are remarkably low, ranging between 10^{-10} and 10^{-8}. This level of precision highlights the reliability of the PINN-CS method in delivering accurate results across the entire computational domain, even as the problem complexity varies.

Table 8. Show the predicted solution of PINN-CS and the exact solution of 3D Bloch Torrey equations for example 3.4.

N	PINN-CS	Exact Solution	Absolute Error
0	0.030370213	0.030370213	0
1	0.029611863	0.029611863	6.25×10^{-10}
2	0.028889991	0.028889989	2.45×10^{-09}
3	0.028203346	0.02820334	5.39×10^{-09}
4	0.027552016	0.027552007	9.39×10^{-09}
5	0.026935346	0.026935332	1.44×10^{-08}
6	0.026352458	0.026352437	2.03×10^{-08}
7	0.025802799	0.025802772	2.70×10^{-08}
8	0.025285102	0.025285068	3.45×10^{-08}
9	0.024799831	0.024799789	4.27×10^{-08}
10	0.02434539	0.024345339	5.15×10^{-08}

 doi:10.20944/preprints202501.1072.v1

4. Conclusion

In this study, we introduced a novel framework combining Monte Carlo-based Physics-Informed Neural Networks (PINNs) with the Cuckoo Search (CS) algorithm, referred to as PINN-CS, for efficiently solving fractional partial differential equations (fPDEs). The proposed framework capitalizes on the advantages of applying QMC approaches for approximations of the fractional derivatives and the optimization ability of the CS algorithm to reach high total precision and fast convergence rate. Several numerical simulations were performed and the algorithm was tested to confirm that PINN-CS method works for the 2D space-fractional Poisson equations, the 2D time-space fractional diffusion equations, and the 3D time-space fractional Bloch-Torrey equations. The results consist that the proposed PINN-CS framework is more accurate and efficient than the fPINN and Monte Carlo fPINN techniques. The usage of the structured sampling nodes and block-sampled approximations saves computer space and reduces computational time considerably however, the precision of the solution estimate is still high. Further, the performance of proposed PINN-CS in approximating the solutions of irregular domains and the corresponding fuzzy boundary conditions clearly shows that the deep learning based PINNs are suited well for applied problems. The convergence analysis indicates fast convergence of the network parameters with the help of the CS algorithm thereby making the approximated solution to be in correct physical approximation with governing equation. The best absolute errors and near perfect overlaps with exact solutions testify towards the effectiveness and efficiency of the proposed framework. Further, the results show the significance of proper distribution of nodes and further advanced optimization for achieving increased accuracy while solving fractional PDEs of high dimensions.

The PINN-CS framework defines new base lines for solving fractional PDEs especially for issues that arise with irregular geometries, fuzzy boundaries, and high dimensional systems. The approach described in this research is easily transferable to computational science and engineering including biomechanics, materials design, and heated fluids. The PINN-CS framework defines new base lines for solving fractional PDEs especially for issues that arise with irregular geometries, fuzzy boundaries, and high dimensional systems. The approach described in this research is easily transferable to computational science and engineering including biomechanics, materials design, and heated fluids.

Funding: This study is supported via funding from Prince Sattam bin Abdulaziz University project number (PSAU/2024/R/1446).

Availability of Data and Materials: The source code of this study will be provided on GitHub after the acceptance of this manuscript.

Acknowledgment: Not applicable.

Competing Interests: The authors of this manuscript declare no competing interest.

References

1. Al-Refai, M. and Baleanu, D., 2024. On new solutions of the normalized fractional differential equations. Fractals, 32(06), p.2450115.
2. Yu, Y., Perdikaris, P., & Karniadakis, G. E. (2016). Fractional modeling of viscoelasticity in 3D cerebral arteries and aneurysms. Journal of computational physics, 323, 219-242.
3. Gutleb, T. S., & Carrillo, J. A. (2023). A static memory sparse spectral method for time-fractional PDEs in arbitrary dimensions. arXiv e-prints, arXiv-2304.
4. D'Elia, M., & Gunzburger, M. (2013). The fractional Laplacian operator on bounded domains as a special case of the nonlocal diffusion operator. Computers & Mathematics with Applications, 66(7), 1245-1260.
5. Tadjeran, C., Meerschaert, M. M., & Scheffler, H. P. (2006). A second-order accurate numerical approximation for the fractional diffusion equation. Journal of computational physics, 213(1), 205-213.

6. Zhang, H., Jiang, X., Wang, C., & Fan, W. (2018). Galerkin-Legendre spectral schemes for nonlinear space fractional Schrödinger equation. Numerical Algorithms, 79, 337-356.
7. Zhang, H., & Jiang, X. (2019). Unconditionally convergent numerical method for the two-dimensional nonlinear time fractional diffusion-wave equation. Applied Numerical Mathematics, 146, 1-12.
8. Hafeez, M.B. and Krawczuk, M., 2024. Fractional Spectral and Fractional Finite Element Methods: A Comprehensive Review and Future Prospects. Archives of Computational Methods in Engineering, pp.1-12.
9. Zhang, H., Jiang, X., & Yang, X. (2018). A time-space spectral method for the time-space fractional Fokker–Planck equation and its inverse problem. Applied Mathematics and Computation, 320, 302-318.
10. Jia, J., & Jiang, X. (2023). Improved uniform error bounds of exponential wave integrator method for long-time dynamics of the space fractional Klein-Gordon equation with weak nonlinearity. Journal of Scientific Computing, 97(3), 58.
11. Odeh, A., Al-Shugaa, M.A., Al-Gahtani, H.J. and Mukhtar, F., 2024. Analysis of Laminated Composite Plates: A Comprehensive Bibliometric Review. Buildings, 14(6), p.1574.
12. Lin, Z., Liu, F., Wang, D., & Gu, Y. (2018). Reproducing kernel particle method for two-dimensional time-space fractional diffusion equations in irregular domains. Engineering Analysis with Boundary Elements, 97, 131-143.
13. Liu, F., Zhuang, P., Turner, I., Anh, V., & Burrage, K. (2015). A semi-alternating direction method for a 2-D fractional FitzHugh–Nagumo monodomain model on an approximate irregular domain. Journal of Computational Physics, 293, 252-263.
14. Pang, G., Chen, W., & Fu, Z. (2015). Space-fractional advection–dispersion equations by the Kansa method. Journal of Computational Physics, 293, 280-296.
15. Zayernouri, M., Ainsworth, M., & Karniadakis, G. E. (2015). A unified Petrov–Galerkin spectral method for fractional PDEs. Computer Methods in Applied Mechanics and Engineering, 283, 1545-1569.
16. Fan, W., Liu, F., Jiang, X., & Turner, I. (2017). A novel unstructured mesh finite element method for solving the time-space fractional wave equation on a two-dimensional irregular convex domain. Fractional Calculus and Applied Analysis, 20(2), 352-383.
17. Xu, Y., Li, J., Chen, X., & Pang, G. (2020). Solving fractional Laplacian visco-acoustic wave equations on complex-geometry domains using Grünwald-formula based radial basis collocation method. Computers & Mathematics with Applications, 79(8), 2153-2167.
18. Guo, L., Zeng, F., Turner, I., Burrage, K., & Karniadakis, G. E. (2019). Efficient multistep methods for tempered fractional calculus: Algorithms and simulations. SIAM Journal on Scientific Computing, 41(4), A2510-A2535.
19. Sun, H., Liu, X., Zhang, Y., Pang, G., & Garrard, R. (2017). A fast semi-discrete Kansa method to solve the two-dimensional spatiotemporal fractional diffusion equation. Journal of Computational Physics, 345, 74-90.
20. Jia, J., Zhang, H., Xu, H., & Jiang, X. (2021). An efficient second order stabilized scheme for the two-dimensional time fractional Allen-Cahn equation. Applied Numerical Mathematics, 165, 216-231.
21. Zhang, J., Lv, J., Huang, J., & Tang, Y. (2023). A fast Euler–Maruyama method for Riemann–Liouville stochastic fractional nonlinear differential equations. Physica D: Nonlinear Phenomena, 446, 133685.
22. Bu, W., Yang, H., & Tang, Y. (2023). Two fast numerical methods for a generalized Oldroyd-B fluid model. Communications in Nonlinear Science and Numerical Simulation, 117, 106963.
23. Leonenko, N., & Podlubny, I. (2022). Monte Carlo method for fractional-order differentiation extended to higher orders. Fractional Calculus and Applied Analysis, 25(3), 841-857.
24. Leonenko, N., & Podlubny, I. (2022). Monte Carlo method for fractional-order differentiation extended to higher orders. Fractional Calculus and Applied Analysis, 25(3), 841-857.
25. Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics, 378, 686-707.
26. Raissi, M., Yazdani, A., & Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science, 367(6481), 1026-1030.

 doi:10.20944/preprints202501.1072.v1

27. Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. Computer Methods in Applied Mechanics and Engineering, 360, 112789.
28. Jin, X., Cai, S., Li, H., & Karniadakis, G. E. (2021). NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. Journal of Computational Physics, 426, 109951.
29. Pang, G., Lu, L., & Karniadakis, G. E. (2019). fPINNs: Fractional physics-informed neural networks. SIAM Journal on Scientific Computing, 41(4), A2603-A2626.
30. Wang, S., Zhang, H., & Jiang, X. (2023). Physics-informed neural network algorithm for solving forward and inverse problems of variable-order space-fractional advection–diffusion equations. Neurocomputing, 535, 64-82.
31. Guo, L., Wu, H., Yu, X., & Zhou, T. (2022). Monte Carlo fPINNs: Deep learning method for forward and inverse problems involving high dimensional fractional partial differential equations. Computer Methods in Applied Mechanics and Engineering, 400, 115523.
32. Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256). JMLR Workshop and Conference Proceedings.
33. Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
34. Yang, Z., Liu, F., Nie, Y., & Turner, I. (2020). An unstructured mesh finite difference/finite element method for the three-dimensional time-space fractional Bloch-Torrey equations on irregular domains. Journal of Computational Physics, 408, 109284.
35. Magin, R. L., Abdullah, O., Baleanu, D., & Zhou, X. J. (2008). Anomalous diffusion expressed through fractional order differential operators in the Bloch–Torrey equation. Journal of Magnetic Resonance, 190(2), 255-270.
36. Anagnostopoulos, S. J., Toscano, J. D., Stergiopulos, N., & Karniadakis, G. E. (2024). Residual-based attention in physics-informed neural networks. Computer Methods in Applied Mechanics and Engineering, 421, 116805.
37. Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with computers, 29, 17-35