Mathematics

https://www.aimspress.com/journal/Math

AIMS Mathematics, 10(6): 13721–13740.
DOI: 10.3934/math.2025618
Received: 18 February 2025
Revised: 30 May 2025
Accepted: 04 June 2025
Published: 13 June 2025

Research article

Deep learning approaches for $\beta-$conformable fractional differential equations: A PINN, NRPINN-s, and NRPINN-un based solutions

Sadullah Bulut and Muhammed Yiğider*

Department of Mathematics, Erzurum Technical University, Yakutiye, Erzurum 25050, Turkey

* **Correspondence:** Email: myigider@erzurum.edu.tr.

Abstract: In this paper, we present a numerical approach for solving $\beta-$conformable fractional differential equations (FDEs) using physics-informed neural networks (PINNs) and their enhanced versions, NRPINN-s and NRPINN-un. The proposed method combines the flexibility of artificial neural networks with the inherent structure of fractional calculus, allowing the solution of complex initial and boundary value problems without domain discretization. The loss function in the model includes initial and boundary conditions and is constructed using modifiable parameters (weights and biases). The efficiency of the proposed methods is demonstrated by numerical experiments on heat and wave equations. The obtained results show that NR-PINNs are superior to classical PINNs in improving convergence, reducing local errors and showing good performance. Visual and tabular comparisons of the solutions obtained from the method with analytical solutions confirm the effectiveness of the approach.

Keywords: Physics-Informed Neural Networks (PINNs); $\beta-$conformable fractional differential equations; NR-PINN; deep learning
Mathematics Subject Classification: 65M50, 35R11, 68T07

1. Introduction

Fractional partial differential equations (FPDEs), which are of interest to many scientists around the world, have been frequently used in the modeling of complex systems in recent years due to their effectiveness in solving problems in daily life. When the literature is examined, there are numerous methods such as the variable separation [1], the extended auxiliary equation mapping method [2], extended Fan sub equation [3], a generalized exponential rational function method [4], backlund transformation [5], the extended direct algebraic method [6], the Lie symmetry [7], the lumped Galerkin approach [8], the fourier pseudo-spectral method [9], extended direct algebraic method [10], extended modified rational expansion method [11], symbolic computations method [12], semi-inverse variational principle [13], the simplified Hirota's method [14], the Darboux transformation [15], the Hirota's bilinear method [16], the Jacobi elliptic function method [17], the sine-Gordon expansion method [18], the homogeneous balance method [19], the extended homoclinic test function method [20], the improved Bernoulli sub-equation function method [21], the modified simple equation method [22], the shooting method [23] and many other methods [24–26] for the solution of ordinary differential equation (ODE), partial differential equation (PDE) and fractional differential equations (FDEs) obtained from the modeling of daily life problems. In equations where these methods are applied, although the existence of a solution is known, in cases where it is not possible to obtain analytical solutions, iterative methods are used to approach the solution numerically.The first step of conventional numerical methods such as Runge−Kutta and Finite Difference Method (FDM) is always discretization of the domain. Using conventional methods, numerical solutions of relative equations can only be obtained at discretization nodes on the interval or domain. Interpolation or curve fitting methods are generally used to obtain solutions other than nodes. While solving the problem, the interpolation process performed in addition to the numerical method causes an increase in the amount of error. Moreover, when the domain is discretized, as the number of iterations increases, the cumulative error amount resulting from such methods also increases. In addition, Lee and Kang proposed artificial neural networks (ANN) as an alternative to iteration-based classical methods for obtaining numerical solutions of differential equations.

In artificial neural networks, the training of the network is carried out by using the nodes obtained by discretizing the domain. After the training of the network is completed, the neural network can generate a numerical solution for each point of the domain. Although artificial neural networks cannot theoretically guarantee the global optimal solution in approximating functions, they can reach the optimal solution to a large extent. Therefore, artificial neural networks have been one of the preferred methods for obtaining numerical solutions of differential equations in the last decade. In the first study in the literature on obtaining the numerical solutions of differential equations with artificial neural networks, the solution of finite difference equations was carried out with the Hopfield Neural Network model [27].

Physics-informed neural networks (PINNs), a breakthrough in recent years, have been applied in the field of computational science and engineering and become a powerful method [28]. PINNs are different from artificial neural networks (ANN) only in terms of the data learning combined with learning the physical rules in the process of their integration which has become the hallmark of their distinction. [30–32]. The innovation has even allowed for the correct modeling of the physical systems when the data was limited. PINNs have not only ensured consistency with the observational data, but also have been consistent with the basic laws of physics by minimizing a loss function that contains data, physical law, and boundary condition losses. [33, 34] Some of the main advantages of the approach also are higher accuracy, a decrease in the need for large datasets and the suitability for solving inverse problems. Due to providing a flexible structure through the solution of complex and physics-based problems, successful investigations have been carried out in many areas such as fluid mechanics, material science, climatic simulations, biomechanics and quantum mechanics, among many others [35]. Especially in high-dimensional, nonlinear systems, it has significantly become a cost-effective option compared to conventional numerical methods such as finite differences and finite elements. Hence, PINNs as an efficient and promising machine learning approach to scientific computing are capable of increasing the predictive power of scientific computation by unifying both data-driven and physics-based modeling.

Artificial neural networks (ANNs) and physics-informed neural networks (PINNs) represent two significant approaches in machine learning and computational science. While ANNs learn patterns from large datasets for tasks such as classification, prediction, and optimization, PINNs integrate physical laws into the learning process, making them particularly effective for systems governed by differential equations. Unlike ANNs, which rely heavily on extensive data, PINNs leverage prior physical knowledge to construct models with greater accuracy and reliability, even with limited data. Consequently, ANNs are widely applied in fields such as image processing, natural language processing, and financial forecasting, whereas PINNs are particularly advantageous in domains like fluid mechanics, materials science, and engineering systems. Therefore, the choice between ANNs and PINNs depends on the problem at hand, with each offering distinct advantages based on data availability and the underlying system's physical constraints.

In this study, three different deep learning-based approaches are considered for solving $\beta-$conformable fractional differential equations: Physics-Informed Neural Networks (PINN), Nonlocal Residual PINN-s (NR-PINN-s), and Nonlocal Residual PINN-un (NR-PINN-un). PINN is an innovative method that aims to solve the differential equations of physical systems by combining them with data and blending physics-based knowledge with neural networks. However, this approach may encounter limitations such as instability or low accuracy in some cases. Therefore, the nonlocal residual PINN (NR-PINN) structure was developed to offer more advanced solutions.

NR-PINN stands for "Nonlocal Residual-Based Physically Aware Neural Network" and was first proposed by Pang and colleagues [36]. Unlike classical PINN, it evaluates the residuals of differential equations not only at specific points but also holistically across certain regions of the solution domain. This nonlocal residual term formulation enhances the stability and accuracy of the solution, particularly for problems involving complex geometries and strong nonlinearities.

Two main types of NR-PINN have been defined in the literature: NR-PINN-s and NR-PINN-un. NR-PINN-s, proposed by Pang et al. [37], directly calculate the residual terms from the predicted solution and incorporate their spatial average into the loss function. The NR-PINN-un approach was developed by Lu et al. [38] and aims to learn the spatial average rather than the solution itself. This structure provides more stable and consistent results, especially for sparse data problems.

In this study, the PINN, NR-PINN-s, and NR-PINN-un approaches were compared in solving $\beta-$conformable fractional differential equations; it was demonstrated that NR-PINN-based methods offer significant advantages over the classical PINN approach in terms of accuracy, stability, and overall performance.

2. Materials and methods

2.1. Preliminaries

In this section, we introduce some definitions and general concepts related to the fractional derivative that can be used later in this article [39, 40].

Definition 2.1. *Let $a \in \mathbb{R}$ and h be a function such that $h : [a, \infty) \to \mathbb{R}$. Then, the β-derivative of h is defined as:*

$$ {}_{0}^{A}D_{t}^{\beta}h(t) = \begin{cases} \lim\limits_{\varepsilon\to 0} \frac{h\left(t+\varepsilon\left(t+\frac{1}{\Gamma(\beta)}\right)^{1-\beta}\right)-h(t)}{\varepsilon} & \textit{for all } t \geq 0, 0 < \beta \leq 1 \\ h(t) & \textit{for all } t \geq 0, \beta = 0, \end{cases} $$

where h is a function such that h : $[0, \infty) \to \mathbb{R}$ *and* Γ *the Gamma-function*

$$ \Gamma(\zeta) = \int_0^\infty t^{\zeta-1}\mathrm{e}^{-1}\,\mathrm{d}t. $$

If the above limit exists, then h is said to be β-differentiable. Note that for $\beta = 1$, we have ${}_{0}^{A}D_{t}^{\beta}h(t) = \frac{\mathrm{d}}{\mathrm{d}t}h(t)$. In addition, unlike other fractional derivatives, the β-derivative of a function can be defined locally at a given point, like the first-order derivative.

When applying this derivative operator in the PINN, PINN-s, and PINN-un approaches, the theoretically defined limit must be calculated using a numerical approximation. Therefore, instead of taking the $\varepsilon \to 0$ limit directly, an approximate derivative is obtained by selecting a fixed and small ε value. In the code, this value is typically defined as epsilon inf s = np sqit(np.finfo(np.float32).eps), which is determined in accordance with the computational precision of 32 -bit floating-point numbers as specified by the IEEE 754 standard. This ensures that a sufficiently small ε value is used while maintaining numerical stability.

Definition 2.2. *Let us consider a function g* : $[0, \infty) \to \mathbb{R}$ *and* $\alpha \in (0, 1]$. *Then the conformable fractional derivative of g of order* α *is defined as*

$$ T_\alpha(g(t)) = \lim_{\varepsilon\to 0} \frac{g\left(t+\varepsilon t^{1-\alpha}\right) - g(t)}{\varepsilon} $$

Some important properties of the conformal fractional derivative can be given as follows:

$$ \begin{aligned} T_\alpha\left(t^p\right) &= pt^{p-\alpha} \quad \text{for all } p \in \mathbb{R}, \\ T_a(\eta) &= 0 \quad \text{for all constant function } \eta, \\ T_\alpha(fg) &= fT_\alpha(g) + gT_\alpha(f), \\ T_\alpha(f(t)) &= t^{1-\alpha}\frac{df}{dt}(t) \text{ where} f \text{ is differentiable}, \\ T\left(\frac{1}{\alpha}t^\alpha\right) &= 1, \\ T_\alpha\left(e^{cx}\right) &= cx^{1-\alpha}e^{cx}, \quad c \in \mathbb{R}, \\ T_\alpha(\sin bx) &= bx^{1-\alpha}\cos bx \quad , \end{aligned} $$

see for detail [40].

2.2. Solution method

In recent years, artificial neural network (ANN) models have begun to be used in widespread situations to solve many real-world problems due to their professional and expert skills. We do not prefer to frame ANNs to solve fractional differential equations (FDEs). Compared to traditional distributed solutions, ANN-based solutions offer several significant advantages: The approximate solutions obtained exhibit a continuous structure throughout the entire integration domain, and the programming complexity does not increase significantly with the number of attention points. These features make ANNs particularly attractive for thin files or intensive manual problems.

Unlike traditional methods that produce a fixed step shape and solutions only at certain points, ANN-based models (especially physics informed neural networks-PINNs) can produce smooth and differentiable solutions throughout the entire domain. Obtaining continuous results in traditional perturbations often requires interpolation, which can lead to additional digital errors. However, ANNs are essentially integrated directly into the loss function of the neural network, completely eliminating its separation from the domain. This model provides a significant advantage over classical solvers, allowing accurate and physical solutions to be produced even for large, high-dimensional or geometrically complex problems.

Let us consider FDE of order $\beta \in [0, 1]$. This is defined as:

$$\begin{aligned}
{}_0^A D_t^{\beta} u &= F\left(u, {}_0^A D_x^{\beta} u, {}_0^A D_x^{2\beta} u, \ldots, r(x,t)\right) \\
u(0,t) &= -Ltf(t) \quad t \geqslant 0, \\
u(L,t) &= Lth(t) \quad t \geqslant 0, \\
u(x,0) &= x(x-L)g(x) \quad 0 \leqslant x \leqslant L.
\end{aligned} \tag{2.1}$$

Consider the time fractional initial-boundary value problem (2.1). Let $\vec{p} = \vec{p}(\vec{\alpha}, \vec{\beta}, \vec{w}, \vec{\psi})$ be the parameters of the feed-forward neural network with $\vec{\alpha}, \vec{\beta}, \vec{\psi}$ and $\vec{w} \in R$, where m is the number of neurons in the hidden layer. For $j = 1, 2, \ldots, n$, where n is the total number of inputs, the output of the neural network at $x_j, t_j \in [a, b]$ is denoted by $DNN\left(x_j, t_j, \vec{p}\right)$ for parameter values $\vec{p}$. In general, the points obtained from the fragmentation of the interval $[a, b]$ such that $x_j = t_j = a + j.h$ for a fixed step length of $h > 0$ are used for training the network.

For the solution of the given problem, the trial function satisfying the initial and boundary conditions can be expressed as [28]

$$u_T(x, t, \vec{p}) = t(x-L)f(t) + xth(t) + x(x-L)g(x) + x(x-L)tNN(x, t; \vec{p}).$$

$i = 1, 2, 3, \ldots, m$ where m represents the number of neurons in the hidden layer and z_i represents the ith neuron in the hidden layer (as seen in Figure 1). For inputs x_j, t_j, the output of the z_i neuron is calculated as $z_i = \psi_i x_j + w_i t_j + \beta_i$ when w_i is the weight and β_i is the threshold. The output of each neuron processed by the activation function is then weighted. The output of the neural network is determined as the sum of the weighted outputs. That is, when $\vec{p} = \vec{p}\ (\ \vec{\alpha},\ \vec{w}, \vec{\psi}, \vec{\beta}\ )$ represents the unknown parameters of the network, the weighted sum value obtained with

$$DNN\left(x_j, t_j, \vec{p}\right) = \sum_{i=1}^{m} \alpha_i \sigma\left(z_i\right)$$

constitutes the output of the feedforward neural network. The sigmoid function

$$\sigma\left(z_i\right) = \frac{1}{1 + e^{(-z_i)}}$$

is preferred as the activation function given in $NN\left(x_j, t_j, \vec{p}\right)$. Derivatives of the sigmoid function can be written using the sigmoid function as follows:

$$\begin{aligned}
\sigma' &= -\sigma^2 + \sigma, \\
\sigma'' &= 2\sigma^3 - 3\sigma^2 + \sigma, \\
\sigma''' &= -6\sigma^4 + 12\sigma^3 - 7\sigma^2 + \sigma, \\
&\;\;\vdots
\end{aligned}$$

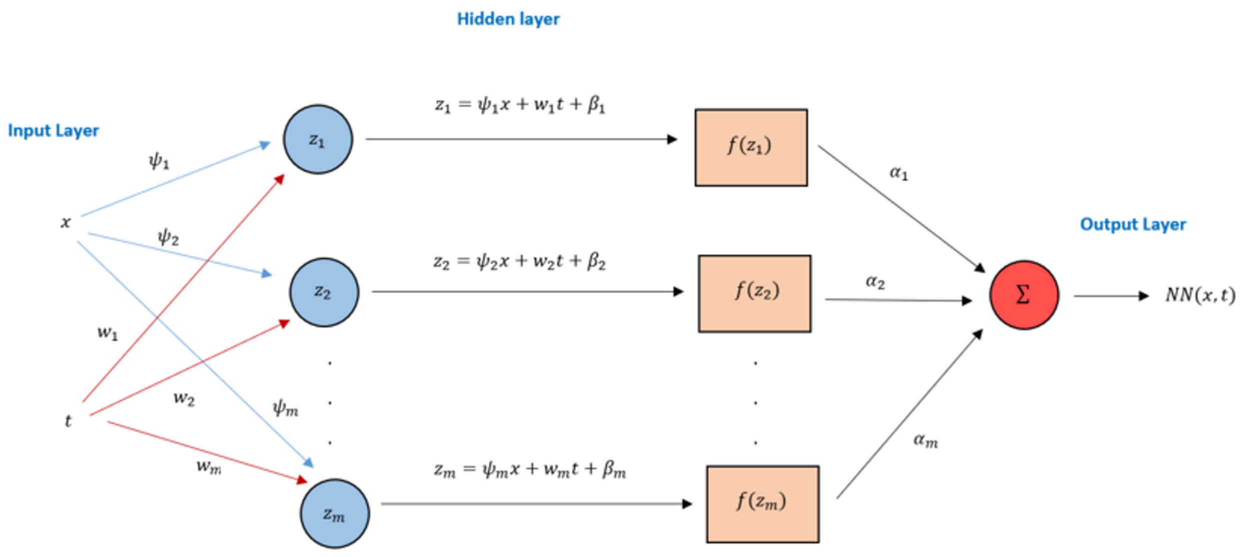

Figure 1. Topological structure of feedforward artificial neural network for numerical solutions of fractional partial differential equations.

A different activation function can be used other than the sigmoid function. The most preferred activation functions in the literature are the hyperbolic tangent and sigmoid functions.

$$\varphi(z_i) = \frac{e^{z_i} - e^{-z_i}}{e^{z_i} + e^{-z_i}}$$

The derivatives of the above hyperbolic tangent activation function are

$$\begin{aligned}
\varphi' &= 1 - \varphi^2, \\
\varphi'' &= 2\varphi^3 - 2\varphi, \\
\varphi''' &= 24\varphi^5 - 36\varphi^3 + 12\varphi, \\
&\;\;\vdots
\end{aligned}$$

On the other hand, choosing the activation function differently in general does not change the result much. The main reason for this is to adjust the parameter values by minimizing the error caused by the output of the network calculated based on the unknown parameters. Therefore, it is more important how the parameters of the network are determined rather than which activation function is used. In this study, sigmoid and tanh activation functions are used because these functions are continuously differentiable and exhibit universal approximation properties. This allows the error-free use of derivative information in the solution of differential equations. Modern activation functions such as ReLU, Swish and GELU have also been tried, but stability problems were observed in the PINN structure due to derivative discontinuities. Therefore, sigmoid and tanh were preferred.

At the beginning, the output of the network is calculated with randomly determined parameter values. As a result of the output obtained, an error will occur in the network. In order to minimize the error that occurs, the back propagation algorithm is generally used and the error is spread to the whole network by updating the $\vec{p}$ unknown parameter values. When n is considered as the number of inputs, the error in a feed forward neural network is calculated using

$$E = \frac{1}{2}\sum_{j=1}^{m}\left(d_j - u_j\right)^2.$$

In the given equation, d_j; is the expected output value of the network for inputs t_j, x_j and y_j is the output value that the network actually produces. On the other hand, in order to calculate the error in the solution of the ordinary differential equation, the cost function is first calculated using the equation

$$e_i = \frac{1}{2}\left({}_0^A D_t^\beta u_T - f\left(x_j, t_j, u_j\right)\right)^2$$

for the input values t_j, x_j Accordingly, the cost function, which is the sum of the error values for each input, is given by

$$E = \sum_{j=1}^{n} e_i = \sum_{j=1}^{n} \frac{1}{2}\left({}_0^A D_t^\beta u_T - f\left(x_j, t_j, u_j\right)\right)^2$$

Our goal is to minimize the cost function to solve the differential equation. μ is the learning coefficient, which usually takes a value between 0 and 1 ; gradient descent algorithm is used to update α_i, w_i, β_i and $\vec{\psi}$ values for $i = 1, 2, \ldots, n$.

In this approach, parameter values are updated to

$$\alpha_i = \alpha_i - \mu\frac{\partial E}{\partial \alpha_i} \quad w_i = w_i - \mu\frac{\partial E}{\partial w_i} \quad \beta_i = \beta_i - \mu\frac{\partial E}{\partial \beta_i} \quad \psi_i = \psi_i - \mu\frac{\partial E}{\partial \psi_i}$$

respectively.

When updating the parameter values, the partial derivatives of the cost function with respect to the unknown parameters for $i = 1, 2, 3, \ldots, m$

$$\frac{\partial u_T}{\partial t_j} = \frac{\partial}{\partial t_j}\left[t_j(x_j - l)f(t) + x_j t_j h(t) + x_j(x_j - l)g(x)\right] + x_j t_j (x_j - l) DNN(x_j, t_j; \overrightarrow{p})$$

$$\frac{\partial DNN(x_j, t_j; \overrightarrow{p})}{\partial t_j} = \sum_{i=1}^{m} \alpha_i w_i \sigma(z_i)\sigma'(z_i)$$

$$= \sum_{i=1}^{m} \alpha_i w_i \sigma(z_i)(1-\sigma(z_i))$$

$$\frac{\partial^2 DNN(x_j, t_j; \overrightarrow{p})}{\partial^2 t_j} = \sum_{i=1}^{m} \alpha_i w_i^2 \sigma(z_i)(1-\sigma(z_i))(1-2\sigma(z_i))$$

Derivatives of the error function with respect to weights are in the form

$$\frac{\partial E}{\partial \alpha_i} = \sum_{j=1}^{n} \left(\frac{\partial u_T}{\partial t_j} - f\left(x_j, t_j, u_j\right)\right)\left(\frac{\partial^2 u_T}{\partial \alpha_j \partial t_j} - \frac{\partial f\left(x_j, t_j, u_j\right)}{\partial \alpha_j}\right)$$

$$\frac{\partial E}{\partial \beta_i} = \sum_{j=1}^{n} \left(\frac{\partial u_T}{\partial t_j} - f\left(x_j, t_j, u_j\right)\right)\left(\frac{\partial^2 u_T}{\partial \beta_j \partial t_j} - \frac{\partial f\left(x_j, t_j, u_j\right)}{\partial \beta_j}\right)$$

$$\frac{\partial E}{\partial w_i} = \sum_{j=1}^{n} \left(\frac{\partial u_T}{\partial t_j} - f\left(x_j, t_j, u_j\right)\right)\left(\frac{\partial^2 u_T}{\partial w_j \partial t_j} - \frac{\partial f\left(x_j, t_j, u_j\right)}{\partial w_j}\right)$$

$$\frac{\partial E}{\partial \psi_i} = \sum_{j=1}^{n} \left(\frac{\partial u_T}{\partial t_j} - f\left(x_j, t_j, u_j\right)\right)\left(\frac{\partial^2 u_T}{\partial \psi_j \partial t_j} - \frac{\partial f\left(x_j, t_j, u_j\right)}{\partial \psi_j}\right)$$

The $\frac{\partial^2 DNN(x_j,t_j,\overrightarrow{p})}{\partial \alpha_i \partial t_j}$, $\frac{\partial^2 DNN(x_j,t_j,\overrightarrow{p})}{\partial w_i \partial t_j}$,$\frac{\partial^2 DNN(x_j,t_j,\overrightarrow{p})}{\partial \beta_i \partial t_j}$ values in the given partial derivatives are calculated respectively as follows:

$$\frac{\partial DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial \alpha_i} = \sigma(z_i),$$

$$\frac{\partial^2 DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial \alpha_i \partial t_j} = w_i \sigma(z_i)(1-\sigma(z_i))$$

$$\frac{\partial DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial w_i} = \alpha_i t_j \sigma(z_i)(1-\sigma(z_i)),$$

$$\frac{\partial^2 DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial w_i \partial t_j} = \alpha_i \sigma(z_i)(1-\sigma(z_i)) + \alpha_i t_j w_i \left[\sigma(z_i)(1-\sigma(z_i))(1-2\sigma(z_i))\right]$$

$$\frac{\partial DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial \psi_i} = \alpha_i x_j \sigma(z_i)(1-\sigma(z_i)),$$

$$\frac{\partial^2 DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial \psi_i \partial t_j} = \alpha_i x_j w_i \left[\sigma(z_i)(1-\sigma(z_i))(1-2\sigma(z_i))\right]$$

$$\frac{\partial DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial \beta_i} = \alpha_i \sigma(z_i)(1-\sigma(z_i)),$$

$$\frac{\partial^2 DNN\left(x_j, t_j, \overrightarrow{p}\right)}{\partial \beta_i \partial t_j} = \alpha_i \sigma(z_i)(1 - \sigma(z_i))(1 - 2\sigma(z_i))$$

Physics-informed neural networks (PINNs) offer a novel approach for solving differential equations by integrating data-driven learning with physical constraints. In this method, the solution of the differential equation, denoted as $u(x,t)$, is approximated by a fully connected deep neural network (DNN). The network takes the independent variables, such as x and t, as inputs and produces the predicted solution $u_\theta(x,t)$ as the output, where θ represents the neural network's weights and biases. Unlike purely datadriven approaches, PINNs incorporate the governing differential equation into the training process through the loss function, ensuring that the learned solution adheres to the underlying physics.

The loss function in PINNs consists of multiple components. First, the residual of the differential equation is minimized, ensuring that the network-generated solution satisfies the governing equation. Mathematically, this is formulated as $\mathcal{L}_{\text{PDE}} = \|\mathcal{N}[u_\theta(x,t)] - f(x,t)\|^2$, where $\mathcal{N}$ represents the differential operator and $f(x,t)$ is the source term. Additionally, the loss function incorporates boundary and initial conditions, expressed as $\mathcal{L}_{\text{BC/IC}} = \|u_\theta(x,t) - g(x,t)\|^2$, to ensure that the solution satisfies the required constraints. The total loss function is then formulated as $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \lambda\mathcal{L}_{\text{BC/IC}}$, where λ is a weighting parameter that balances the contributions of the equation residual and the imposed constraints. One of the key advantages of PINNs is their ability to leverage automatic differentiation (AD) to compute derivatives, enabling the efficient handling of high-order derivatives and fractional derivatives. This feature makes PINNs particularly well-suited for solving fractional differential equations (FDEs), which are often challenging for conventional numerical methods. The training process involves optimizing the neural network parameters using gradient-based methods, such as stochastic gradient descent (SGD) or Adam optimization, to minimize the total loss function. Once trained, the PINN model provides a continuous and differentiable solution that satisfies the differential equation across the entire domain.

Compared to traditional numerical methods, PINNs offer several advantages. They do not require explicit discretization of the problem domain, making them particularly useful for high-dimensional problems and complex geometries. Additionally, PINNs can produce smooth and differentiable solutions, which are beneficial for applications involving higher-order derivatives. However, PINNs also have some limitations, such as high computational costs during training and sensitivity to hyperparameter choices. To address these challenges, enhanced versions such as NR-PINNs have been proposed to improve accuracy and convergence speed.

In summary, PINNs provide a powerful framework for solving differential equations, especially for problems involving fractional derivatives. By embedding physical laws directly into neural networks, they offer an effective alternative to traditional numerical solvers while maintaining flexibility and scalability. The development of improved architectures, such as NR-PINNs, further enhances their applicability to a wide range of scientific and engineering problems.

3. Application

In this section, we present example problems to demonstrate the effectiveness of the proposed method.

Let the inhomogeneous heat equations be given

$$\begin{array}{lll} \text{FDE} & {}_0^A D_t^\beta u & = u_{xx} + \sin x, \quad 0 < x < \pi, t > 0, \\ \text{BC} & u(0,t) & = e^{-t}, \quad t \geqslant 0, \\ & u(\pi,t) & = -e^{-t}, \quad t \geqslant 0, \\ \text{IC} & u(x,0) & = cosx, \quad 0 \leqslant x \leqslant \pi. \end{array} \tag{3.1}$$

Exact solution of fractional differential equation (FDE) (3.1) satisfying BC and IC conditions is

$$u(x,t) = (1 - e^{-t}) \sin x + e^{-t} \cos x$$

and can be seen in Figure 2.

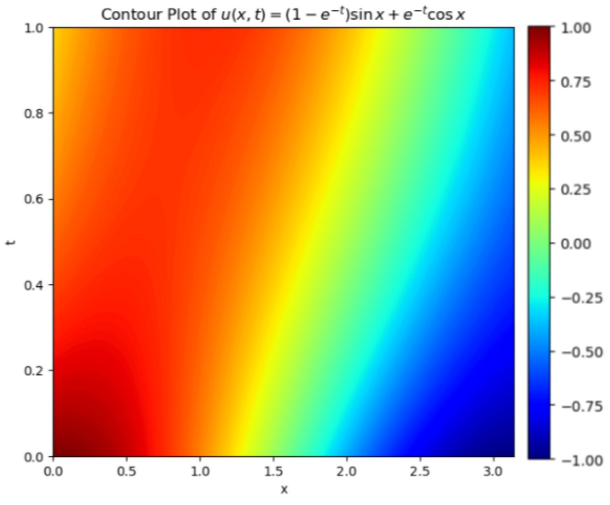

Figure 2. Graphs of the analytical solution of the $\beta-$conformable time fractional derivative Inhomogeneous Heat Equation.

For the solution of the given initial value problem, depending on the output of the neural network, the trial function that satisfies the BC and IC conditions

$$u_T(x,t;\vec{p}) = e^{-t} \cos x + xt(x - \pi) DNN(x,t;\vec{p}).$$

The model training was conducted by generating a training dataset of size 100 × 100 in the domain $(x,t) \in [0,\pi] \times [0,1]$. The neural network consists of three hidden layers, each with 20 neurons, and utilizes the hyperbolic tangent (tanh) activation function. The learning rate was set to 0.001. During the training process, the mean squared error (MSE) loss function was used, and each model was trained for 2000 epochs.

Figure 3 compares the solutions obtained by the PINN and two different NR-PINN methods (NRPINN-s and NRPINN-un), along with their point-wise error distributions. The top row presents the predicted solutions for each method, showing that all models achieve a similar overall solution structure. The bottom row illustrates the point-wise error distribution between the predicted and exact solutions. The standard PINN method exhibits a more widespread error distribution, whereas the NR-PINN-based approaches localize the errors, effectively reducing the overall error magnitude. Notably, the NRPINN-s method appears to be the most effective in minimizing errors, indicating its superior accuracy in solution approximation. These results demonstrate that NR-PINN methods provide more reliable and accurate predictions compared to the classical PINN model.

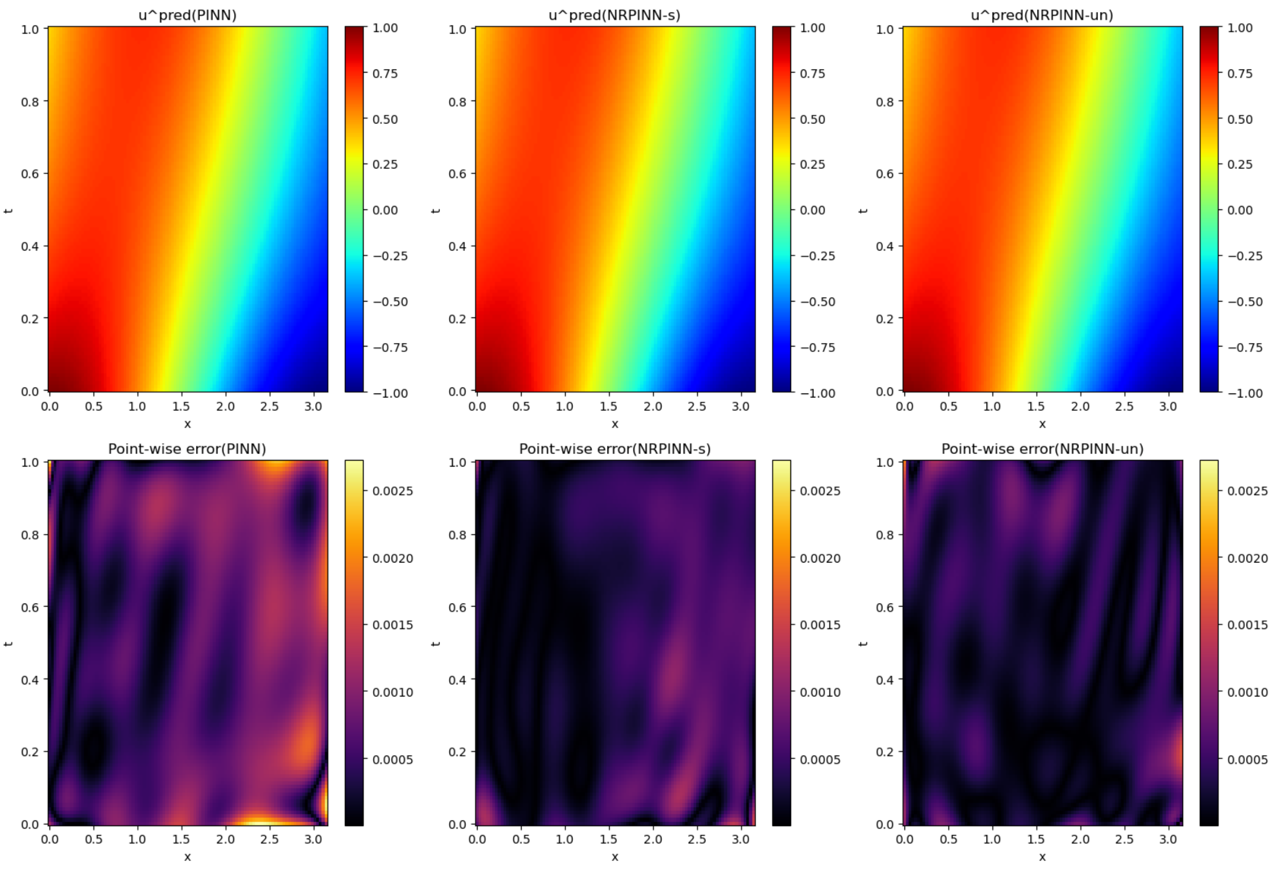

Figure 3. Results for the inhomogeneous heat equation with $\beta = 0.8$. The top row shows the predicted solutions u^{Pred} obtained using PINN, NRPINN-s and NRPINN-un methods. The bottom row presents the corresponding point-wise errors.

Figure 4 visualizes the differences between the solutions obtained by the PINN model and two different NR-PINN methods (NRPINN-s and NRPINN-un).

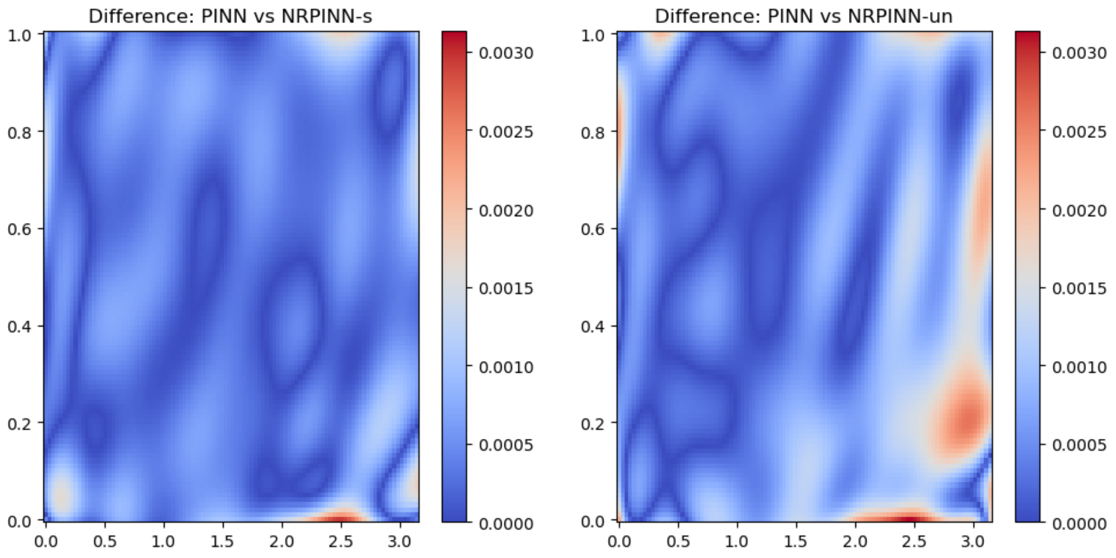

Figure 4. Visualization of the difference between the results of the PINN model and different NR-PINN approaches (NRPINN-s and NRPINN-un). The left panel shows the difference between PINN and NRPINN-s, while the right panel shows the difference between PINN and NRPINN-un. The color scale represents the magnitude of the error.

The left panel shows the differences between PINN and NRPINN-s. Overall, the differences remain at low levels, with a maximum error of approximately 0.003. The most significant deviations are concentrated in the lower regions and the right corner. However, since the overall error distribution is homogeneous and remains small, it can be concluded that the NRPINN-s method produces results that are highly consistent with the PINN model.

The right panel presents the differences between PINN and NRPINN-un. Here, the error distribution appears to be more fluctuating and locally variable. Notably, the right boundary and lower region exhibit slightly larger discrepancies. Nevertheless, the error magnitude remains below 0.003, indicating that the PINN and NRPINN-un models still produce similar solution structures.

This analysis suggests that the NRPINN-s method yields solutions that are more closely aligned with PINN, whereas the NRPINN-un method exhibits slightly larger local differences. However, both NR-PINN approaches maintain low error levels and provide physically consistent solutions compared to the standard PINN model.

As seen in Figure 5, the NRPINN-s model provides the lowest error values in both metrics and shows higher solution performance compared to the standard PINN and NRPINN-s approaches.

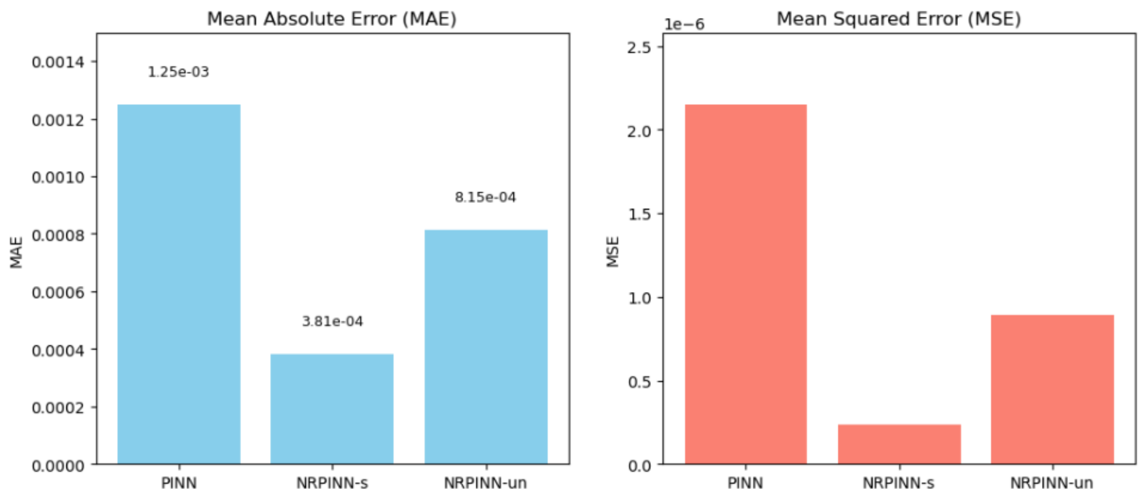

Figure 5. Bar charts of the mean absolute error (MAE) and mean squared error (MSE), illustrating the accuracy performance of different models.

Let the inhomogeneous wave equations be given

$$\begin{array}{lll} \text{FDE} & {}_0^AD_{2t}^{\beta}u & = u_{xx} + \sin x, \quad 0 < x < \pi, t > 0, \\ \text{BC} & u(0,t) & = 0, \quad u(\pi,t) = 0, t \geqslant 0, \\ \text{IC} & u(x,0) & = \sin x, \quad u_t(x,0) = \sin x. \end{array} \tag{3.2}$$

This equation often describes water waves, vibrations of a membrane or a string, the propagation of electromagnetic and sound waves, as well as the transmission of electrical signals in a cable. Here the function $u(x,t)$ describes a small displacement of any point of a vibrating string at position x at time t. Unlike the previous equation, the wave equation includes the term ${}_0^AD_{2t}^{\beta}u$ due to the tension in the string, which represents the vertical acceleration of a vibrating string at point x. Exact solution of FDE (3.2) satisfying BC and IC conditions is

$$u(x,t) = \sin x + \sin x \sin t$$

and can be seen in Figure 6.

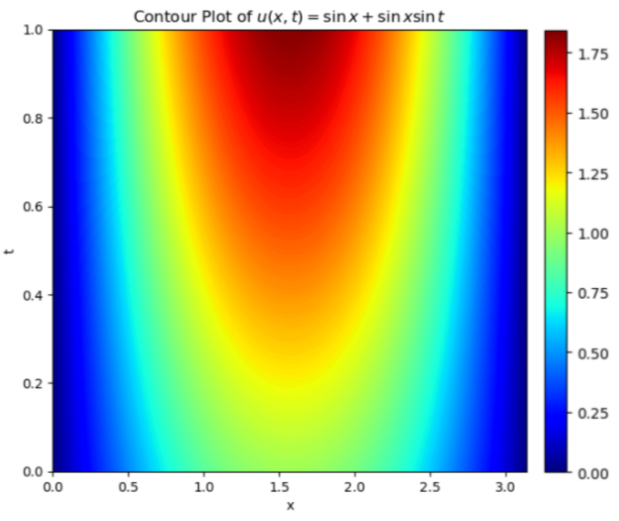

Figure 6. Graphs of the analytical solution of the $\beta-$conformable time fractional derivative Inhomogeneous Wave Equation.

For the solution of the given initial value problem, depending on the output of the neural network, the trial function that satisfies the BC and IC conditions

$$u_T(x,t;\vec{p}) = \sin x + t\sin x + t^2 x(x-\pi)DNN(x,t;\vec{p}).$$

Derivative of the trial solution with respect to the variable t

$$\frac{\partial u_T(x,t,\vec{p})}{\partial t} = \sin x + 2tx(x-\pi)DNN(x,t;\vec{p}).$$

It is clear that the derivative satisfies the IC condition.

The model training was conducted by generating a training dataset of size 100×100 in the domain $(x,t) \in [0,\pi] \times [0,1]$. The neural network consists of three hidden layers, each with 20 neurons, and utilizes the hyperbolic tangent (tanh) activation function. The learning rate was set to 0.001. During the training process, the mean squared error (MSE) loss function was used, and each model was trained for 2000 epochs.

Figure 7 presents a comparative analysis of the predicted solutions and point-wise errors for different neural network approaches in solving the $\beta-$conformable fractional differential equation. The top row illustrates the predicted solutions obtained using PINN, NRPINN-s, and NRPINN-un, demonstrating a high degree of similarity across all methods. The bottom row displays the corresponding point-wise errors, indicating that NRPINN-based approaches exhibit lower error magnitudes compared to the standard PINN, particularly in the central region. This suggests that incorporating NRPINN modifications enhances the accuracy and stability of the solution.

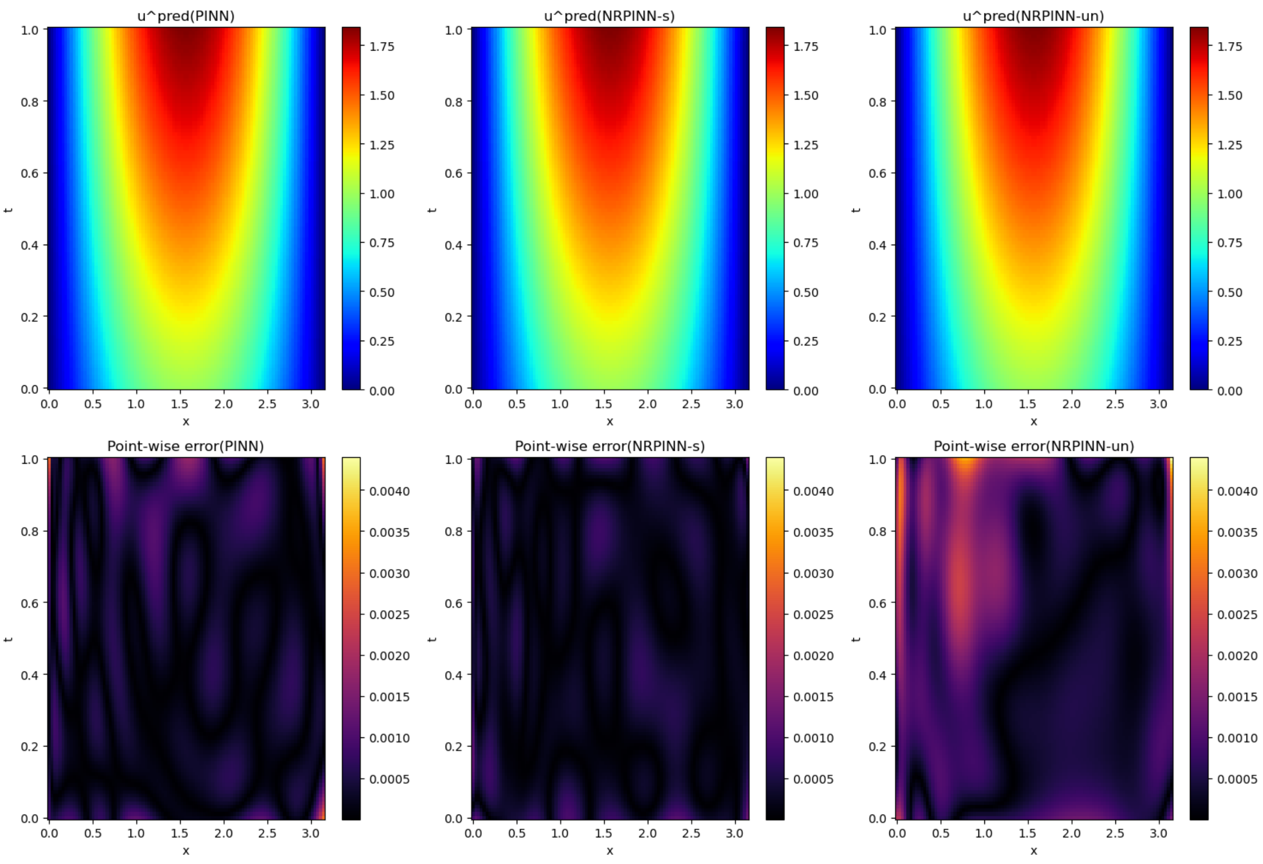

Figure 7. Results for the inhomogeneous wave equation with $\beta = 0.8$. The top row shows the predicted solutions u^{Pred} obtained using PINN, NRPINN-s and NRPINN-un methods. The bottom row presents the corresponding point-wise errors.

Figure 8 visualizes the differences between the PINN method and the NRPINN-s and NRPINN-un approaches. The color scale indicates that the overall differences are relatively low, but there are noticeable increases, particularly in the boundary regions and certain localized areas. This suggests that the NRPINN approaches generally provide solutions consistent with PINN, though some regional discrepancies exist. Furthermore, the more uniform distribution of differences in the NRPINN-un method implies that this approach may exhibit a more balanced error profile.

As seen in Figure 9, the NRPINN-s model provides the lowest error values in both metrics and shows higher solution performance compared to the standard PINN and NRPINN-s approaches.

Finally, the NR-PINN-s method provides a more stable and precise learning process by recording the spectral features of the solution. The NR-PINN-un method includes a special reporting to estimate the unknown coefficients or functional forms of the solution with less error. The methods with this efficiency aim to improve not only the digital accuracy perspective of the proposed NR-PINNs but also the learning dynamics perspective from the classical PINNs in a different way. In the various test problems in the study, the proposed methods have much better performance than the classical common PINNs.

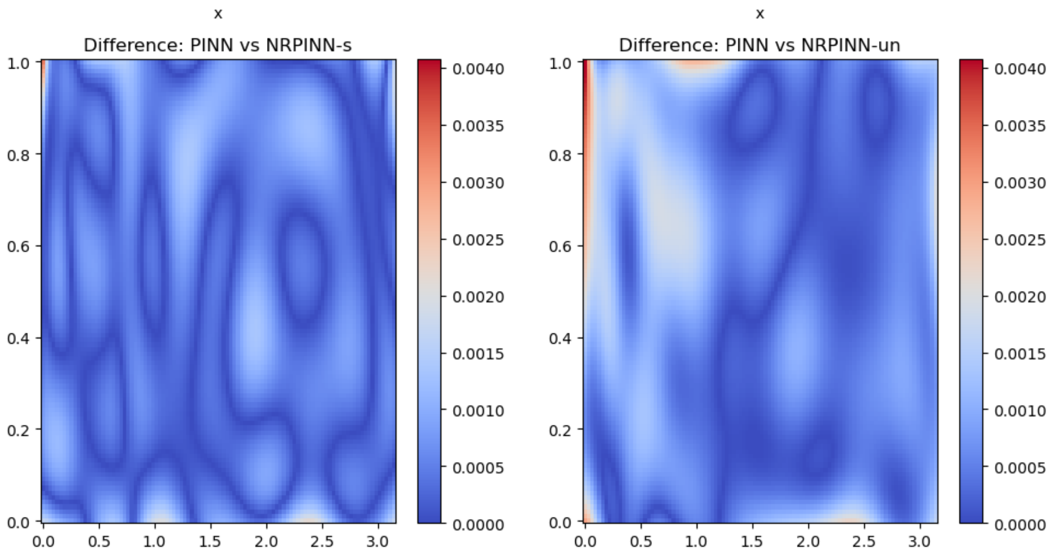

Figure 8. Visualization of the difference between the results of the PINN model and different NR-PINN approaches (NRPINN-s and NRPINN-un). The left panel shows the difference between PINN and NRPINN-s, while the right panel shows the difference between PINN and NRPINN-un. The color scale represents the magnitude of the error.

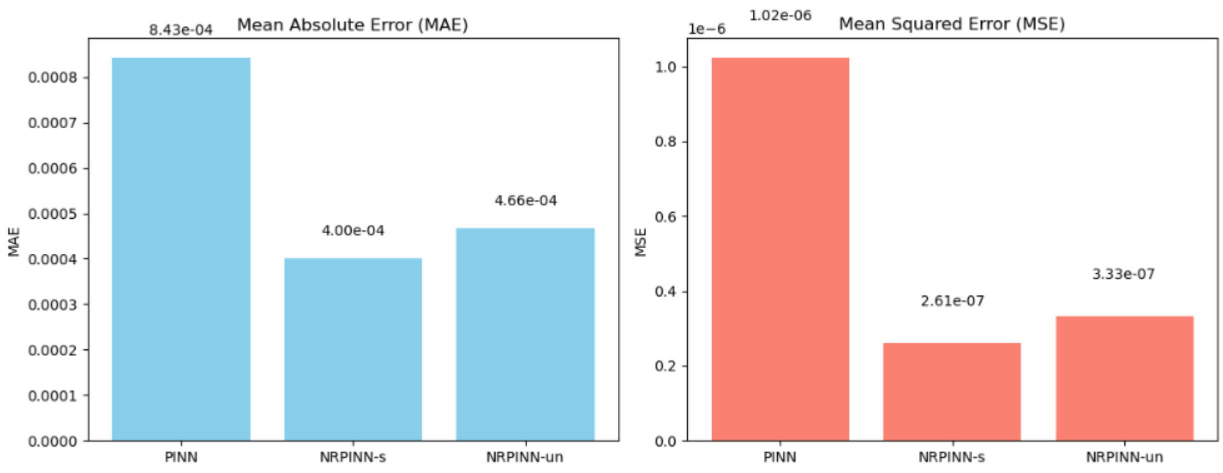

Figure 9. Bar charts of the mean absolute error (MAE) and mean squared error (MSE), illustrating the accuracy performance of different models.

4. Conclusions

In this study, we demonstrate the efficiency of PINNs and their variants, namely NRPINN-s and NRPINN-un, for solving $\beta-$conformable fractional differential equations. Since the loss function includes the governing equation, the proposed methods can effectively model the analytical solution and satisfy the physical requirements. From the numerical results, it is observed that NR-PINNs exhibit larger convergence rate with smaller pointwise error compared to traditional PINNs. The potential of PINNs to provide continuous and differentiable solutions without domain discretization makes them a capable candidate as an alternative method to traditional numerical methods. Intelligent learning algorithms and composite models should be investigated in future studies to improve the accuracy and reduce the computational effort for more complex fractional systems.

Author contributions

S. BULUT: conceptualization, methodology, writing original draft, editing; M. YİĞİDER: supervision, validation. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The author would like to thank the referees for their careful readings and remarkable comments for the improvement of the paper.

Conflict of interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Y. Zhao, J. Xia, X. Lü, The variable separation solution, fractal and chaos in an extended coupled (2+ 1)-dimensional Burgers system, *Nonlinear Dynam.,* **108** (2022), 4195–4205. https://doi.org/10.1007/s11071-021-07100-z
2. A. Akbulut, S. Islam, Study on the Biswas–Arshed equation with the beta time derivative, *Int. J. Appl. Computat. Math.,* **8** (2022), article number 167. https://doi.org/10.1007/s40819-022-01350-0
3. M. Cinar, A. Secer, M. Ozisik, M. Bayram, Derivation of optical solitons of dimensionless Fokas-Lenells equation with perturbation term using Sardar sub-equation method, *Opt. Quant. Electron.*, **54** (2022), article number 402. https://doi.org/10.1007/s11082-022-03819-0
4. S. Kumar, M. Niwas, New optical soliton solutions of Biswas–Arshed equation using the generalised exponential rational function approach and Kudryashov's simplest equation approach, *Pramana*, **96** (2022), 204. https://doi.org/10.1007/s12043-022-02450-8
5. K. Wang, Bäcklund transformation and diverse exact explicit solutions of the fractal combined Kdv–mkdv equation, *Fractals*, **30** (2022), 2250189. https://doi.org/10.1142/S0218348X22501894
6. M. Ghayad, N. Badra, H. Ahmed, W. Rabie, Derivation of optical solitons and other solutions for nonlinear Schrödinger equation using modified extended direct algebraic method, *Alex. Eng. J.*, **64** (2023), 801–811. https://doi.org/10.1016/j.aej.2022.10.054
7. M. Hashemi, M. Mirzazadeh, Optical solitons of the perturbed nonlinear Schrödinger equation using Lie symmetry method, *Optik*, **281** (2023), 170816. https://doi.org/10.1016/j.ijleo.2023.170816

8. J. Martin, S. Duncan, A Galerkin approach for modelling the pantograph-catenary interaction, *Advances In Dynamics Of Vehicles On Roads And Tracks: Proceedings Of The 26th Symposium Of The International Association Of Vehicle System Dynamics, IAVSD 2019, August 12–16, 2019, Gothenburg, Sweden*, 230–241, (2020). https://doi.org/10.1007/978-3-030-38077-9

9. F. Mariano, L. Moreira, A. Nascimento, A. Silveira-Neto, An improved immersed boundary method by coupling of the multi-direct forcing and Fourier pseudo-spectral methods, *J. Braz. Soc. Mech. Sci.*, **44** (2022), 388. https://doi.org/10.1007/s40430-022-03679-5

10. A. Hussain, M. Junaid-U-Rehman, F. Jabeen, I. Khan, Optical solitons of NLS-type differential equations by extended direct algebraic method. *Int. J. Geom. Methods M.*, **19** (2022), 2250075. https://doi.org/https://doi.org/10.1142/S021988782250075X

11. M. Ali, H. El-Owaidy, H. Ahmed, A. El-Deeb, I. Samir, Optical solitons and complexitons for generalized Schrödinger–Hirota model by the modified extended direct algebraic method, *Opt. Quant. Electron.*, **55** (2023), 675. https://doi.org/10.1007/s11082-023-04962-y

12. W. Ma, Y. Zhang, Y. Tang, Symbolic computation of lump solutions to a combined equation involving three types of nonlinear terms, *East Asian J. Appl. Math.*, **10** (2020), 732–745. https://doi.org/10.4208/eajam.151019.110420

13. Y. Gu, S. Zia, M. Isam, J. Manafian, A. Hajar, M. Abotaleb, Bilinear method and semi-inverse variational principle approach to the generalized (2+ 1)-dimensional shallow water wave equation, *Results Phys.*, **45** (2023), 106213. https://doi.org/10.1016/j.rinp.2023.106213

14. A. Yokus, M. Isah, Dynamical behaviors of different wave structures to the Korteweg–de Vries equation with the Hirota bilinear technique, *Physica A*, **622** (2023), 128819. https://doi.org/10.1016/j.physa.2023.128819

15. Y. Shen, B. Tian, T. Zhou, X. Gao, N-fold Darboux transformation and solitonic interactions for the Kraenkel–Manna–Merle system in a saturated ferromagnetic material, *Nonlinear Dynam.*, **111** (2023), 2641–2649. https://doi.org/10.1007/s11071-022-07959-6

16. T. Yin, Z. Xing, J. Pang, Modified Hirota bilinear method to (3+ 1)-D variable coefficients generalized shallow water wave equation, *Nonlinear Dynam.*, **111** (2023), 9741–9752. https://doi.org/10.1007/s11071-023-08356-3

17. A. Hussain, Y. Chahlaoui, F. Zaman, T. Parveen, A. Hassan, The Jacobi elliptic function method and its application for the stochastic NNV system, *Alex. Eng. J.*, **81** (2023), 347–359. https://doi.org/10.1016/j.aej.2023.09.017

18. P. Das, S. Mirhosseini-Alizamini, D. Gholami, H. Rezazadeh, A comparative study between obtained solutions of the coupled Fokas–Lenells equations by Sine-Gordon expansion method and rapidly convergent approximation method, *Optik*, **283** (2023), 170888. https://doi.org/10.1016/j.ijleo.2023.170888

19. E. Fan, H. A. Zhang, Note on the homogeneous balance method, *Phys. Lett. A*, **246** (1998), 403–406. https://doi.org/https://doi.org/10.1016/S0375-9601(98)00547-7

20. A. Yokus, M. Isah, Stability analysis and solutions of (2+ 1)-Kadomtsev–Petviashvili equation by homoclinic technique based on Hirota bilinear form, *Nonlinear Dynam.*, **109** (2022), 3029–3040. https://doi.org/10.1007/s11071-022-07568-3

21. K. Fatema, M. Islam, M. Akhter, M. Akbar, M. Inc, Transcendental surface wave to the symmetric regularized long-wave equation, *Phys. Lett. A*, **439** (2022), 128123. https://doi.org/https://doi.org/10.1016/j.physleta.2022.128123

22. M. Kaplan, A. Bekir, A. Akbulut, E. Aksoy, The modified simple equation method for nonlinear fractional differential equations, *Rom. J. Phys.,* **60** (2015), 1374–1383.

23. G. Gie, C. Jung, H. Lee, Semi-analytic shooting methods for Burgers' equation, *J. Comput. Appl. Math.*, **418** (2023), 114694. https://doi.org/https://doi.org/10.1016/j.cam.2022.114694

24. S. Kiliç, E. Çelik, Complex solutions to the higher-order nonlinear boussinesq type wave equation transform, *Ric. Mat.,* (2022), 1–8. https://doi.org/10.1007/s11587-022-00698-1

25. S. Kılıç, E. Çelik, H. Bulut, Solitary wave solutions to some nonlinear conformable partial differential equations, *Opt. Quant. Electron.*, **55** (2023), 693. https://doi.org/10.1007/s11082-023-04983-7

26. T. Yazgan, E. Ilhan, E. Çelik, H. Bulut, On the new hyperbolic wave solutions to Wu-Zhang system models, *Opt. Quant. Electron.*, **54** (2022), 298. https://doi.org/10.21203/rs.3.rs-1184408/v1

27. H. Lee, I. Kang, Neural algorithm for solving differential equations, *J. Comput. Phys.*, **91** (1990), 110–131. https://doi.org/https://doi.org/10.1016/0021-9991(90)90007-N

28. R. Gladstone, M. Nabian, N. Sukumar, A. Srivastava, H. Meidani, FO-PINN: A First-Order formulation for Physics-Informed Neural Networks, *Eng. Anal. Bound. Elem.*, **174** (2025), 106161. https://doi.org/10.1016/j.enganabound.2025.106161

29. M. Vellappandi, S. Lee, Physics-informed Neural Fractional Differential Equations, *Appl. Math. Model.*, (2025), 116127. https://doi.org/10.1016/j.apm.2025.116127

30. M. Zhong, S. Gong, S. Tian, Z. Yan, Data-driven rogue waves and parameters discovery in nearly integrable PT-symmetric Gross–Pitaevskii equations via PINNs deep learning, *Physica D*, **439** (2022), 133430. https://doi.org/10.1016/j.physd.2022.133430

31. K. Eshkofti, S. Hosseini, A new modified deep learning technique based on physics-informed neural networks (PINNs) for the shock-induced coupled thermoelasticity analysis in a porous material, *J. Therm. Stresses*, **47** (2024), 798–825. https://doi.org/10.1080/01495739.2024.2321205

32. M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

33. Z. Zhou, Z. Yan, Deep learning neural networks for the third-order nonlinear Schrödinger equation: bright solitons, breathers, and rogue waves, *Commun. Theor. Phys.*, **73** (2021), 105006. https://doi.org/10.48550/arXiv.2104.14809

34. J. Song, Z. Yan, Deep learning soliton dynamics and complex potentials recognition for 1D and 2D PT-symmetric saturable nonlinear Schrödinger equations, *Physica D*, **448** (2023), 133729. https://doi.org/10.1016/j.physd.2023.133729

35. B. Zimmering, C. Coelho, O. Niggemann, Optimising neural fractional differential equations for performance and efficiency, *Proc. Mach. Learn. Res.*, **255** (2024), 23.

36. G. Pang, M. D'Elia, M. Parks, G. Karniadakis, nPINNs: Nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator, Algorithms and applications, *J. Comput. Phys.*, **422** (2020), 109760. https://doi.org/10.1016/j.jcp.2020.109760

37. M. D'Elia, M. Parks, G. Pang, G. Karniadakis, Nonlocal Physics-Informed Neural Networks: A unified theoretical and computational framework for nonlocal models, (Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020). https://doi.org/10.48550/arXiv.2004.04276

38. L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, et al., Comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, *Comput. Meth. Appl. Mech. Eng.*, **393** (2022), 114778. https://doi.org/10.1016/j.cma.2022.114778

39. A. Atangana, Derivative with a new parameter: Theory, methods and applications, (Academic Press, 2015).

40. R. Khalil, M. Al Horani, A. Yousef, M. Sababheh, A new definition of fractional derivative, *J. Comput. Appl. Math.* **264** (2014), 65–70. https://doi.org/10.1016/j.cam.2014.01.002