Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2025 (43) 1** : 1–13.
ISSN-0037-8712
doi:10.5269/bspm.78713

A Novel Numerical Treatment for Solving Fractional Order Delay Integro-Differential Algebraic Equations Using Fractional Physics-Informed Neural Networks Method

Haitham M. Khazeem* and Osama H. Mohammed

ABSTRACT: This paper introduces a novel computational framework based on Physics-Informed Neural Networks (PINNs) for the numerical solution of Fractional-Delay Integro-Differential-Algebraic Equations (FDIDAEs). In the proposed approach, we employ the standard feedforward neural networks (NNs) with the delay integro-differential algebraic equations are straightforward encoded into the NN using fractional order differentiation. While the sum of the mean squared fractional order delay integro differential algebraic equations-residuals and the mean squared error in initial conditions is minimized with respect to the NN parameters.

Key Words: Fractional order delay integro-differential algebraic equations, physics-informed neural networks, backpropagation, feedforward neural networks, Adam method.

Contents

1. Introduction

Many scientific and real-life phenomena or natural problems modeled by fractional calculus topics [1,2,3,4,5,6,7,8,9,10]. Fractional-order derivatives, integrals, differential equations (FDEs), integral equations (FIEs), integro-differential equations (FIDEs), or systems of these equations have been recognized as a powerful tools for accurately describing the properties of complex dynamical processes, surpassing the capabilities of standard integer-order derivatives and integrals [11,12]. Therefore, researchers dedicate considerable effort to developing appropriate methods for solving problems involving fractional calculus, including fractional derivatives and integrals. Given the complexity of obtaining analytical solutions for fractional differential equations, fractional integral equations, and fractional integro-differential equations, numerical and approximate methods are commonly employed in most research studies see [11,13,14,15,16,17,18,19,20,21,22].

Fractional order integro-differential algebraic equations (FIDAEs) is a kind of FIDEs where in a system of integro-differential equations, there exist at least one equation does not contain any term with

* Corresponding author.
2010 *Mathematics Subject Classification:* 34K37, 26A33, 34A08, 45J05.
Submitted August 31, 2025. Published December 19, 2025

derivative or integral [23,24]. FIDAEs are commonly employed to study dynamic changes over a time interval or along a distance, as seen in applications like electric circuits, hydraulic circuits, chemical reactions [30].

When at least one term of the FIDAE contains a fixed lag $\tau > 0$, the equation transforms into delay integro-differential algebraic equation of fractional order.

Physics-Informed Neural Networks method, first proposed in [36] are highly effective in solving integer-order partial differential equations (PDEs) even in the presence of scattered and noisy data. PINNs utilize conventional feedforward neural networks (NNs), where partial differential equations are directly incorporated through automatic differentiation. The training process involves minimizing a loss function that combines the mean-squared residuals of the PDE and the mean-squared errors associated with the initial and boundary conditions, optimizing the NNs parameters accordingly [37]. In this work, we modify certain foundational aspects of the PINNs framework to better suit our problem. Although we apply the method to a different class of equations, one of the most significant modifications involves replacing the conventional neural network architecture with a representation based on a weighted power series. In this approach, the series coefficients are treated as trainable parameters, which are optimized to approximate the solution. This modification is particularly advantageous when dealing with fractional-order derivatives, which pose challenges for standard automatic differentiation techniques commonly used in PINNs. To address this, we employ a tailored approach to fractional-order differentiation, replacing automatic differentiation with manually handled or analytically derived expressions. Despite these changes, the core idea of incorporating the governing FIDAEs directly into the loss function remains consistent with the original PINNs methodology.

2. Preliminaries

This section presents fundamental definitions and preliminary notes related to fractional calculus.

Definition 2.1 *The left Reimann-Liouville (R-L) fractional order integral of a function u is given by [17]*

$$ {}_0I_t^{\alpha}u(t) = \frac{1}{\Gamma(\alpha)}\int_0^t (t-\tau)^{\alpha-1}u(\tau)d\tau,\ t>0,\ \alpha\in\mathbf{R}^+ $$

Definition 2.2 *Let $u \in C[\alpha, m]$, $\quad \alpha \geq -1$, the fractional order derivative in the Caputo sense [1,10]*

$$ {}_0^cD_t^{\alpha}u(t) = \begin{cases} \dfrac{1}{\Gamma(m-\alpha)}\int_0^t \dfrac{u^{(m)}(s)}{(t-s)^{\alpha+1-m}}ds, & m-1<\alpha<m, s>0, m\in N \\ \dfrac{d^m}{dt^m}u(t), & \alpha = m \end{cases} $$

Remark 2.1 *The following are some properties of the (R-L) fractional order integral and the Caputo fractional order derivative*

(1) ${}_0I_t^{\alpha}t^{\mu} = \dfrac{\Gamma(\mu+1)}{\Gamma(\mu+\alpha+1)}t^{\alpha+\mu}\ ,\ \mu>-1\ ,\ \alpha>0$

(2) ${}_0^cD_t^{\alpha}t^{\mu} = \dfrac{\Gamma(\mu+1)}{\Gamma(\mu-\alpha+1)}t^{\mu-\alpha}\ ,\ \mu>-1\ ,\ \alpha>0$

(3) ${}_0^cD_t^{\alpha}({}_0I_t^{\alpha}u(t)) = u(t)$

(4) ${}_0I_t^{\alpha}({}_0^cD_t^{\alpha}u(t)) = u(t) - \sum_{k=0}^{n-1}\dfrac{u^{(k)}(0)}{k!}t^k\ ,\ t\geq 0\ ,\ n-1<\alpha<n$

Theorem 2.1 *[38] Let $u : [a,b] \to \mathbf{R}$ be a continuous function and an arbitrary $\zeta > 0$, there exist an algebraic polynomial p such that*

$$ |u(t)-p(t)| \leq \zeta\ ,\ \forall t\in[a,b] \tag{2.1} $$

3. Analysis of Physics-Informed Neural Networks Method

Recently, physics-informed deep learning models have gained attention as effective tools for solving complex mathematical equations, especially differential equations commonly found in scientific and engineering applications. These methods utilize neural networks to approximate solutions which we use here a weighted power series named "trial solutions" and embedding physical laws and constraints into the learning process. Unlike traditional numerical methods, which can be computationally demanding and struggle with high-dimensional problems, physics-informed neural networks provide a more flexible and scalable alternative [32].

In this work, we construct fractional PINNs (fPINNs) using a multilayer feedforward architecture consisting of at least three layers. The input layer receives a set of input variables, each represented by a corresponding node. These nodes forward weighted signals to the hidden layer, where a nonlinear activation function is applied. In our formulation, the hidden layer serves to construct the trial solution of the fractional integro-differential algebraic equation under consideration. The output layer yields the network's approximation to the desired solution. Training involves optimizing the network weights so that the output satisfies the governing equations and associated conditions, typically through minimization of a suitable loss functional. The backpropagation algorithm is employed to compute gradients and update weights during the training process.

The architecture of the fractional PINNs employed in this work consists of three layers. The first is the input layer, which corresponds to a set of equidistant collocation points sampled from the domain of interest. The second layer is the hidden layer, comprising trainable parameters (weights) and a nonlinear activation function. Together, these define a trial solution, which will be examined in detail in the subsequent section. This trial solution is then used to construct a loss function, typically formulated as the average of the residuals associated with the governing equations and boundary conditions. Upon training—by minimizing the loss function through weight optimization—an approximate solution to the problem is obtained, see figure (1).

4. Trial Solution Analysis

The main aim in this section is to construct an assumed solution to the FDIDAE. There are several ways to construct a trial solution depending on the problem. One of these ways [33] is given by

$$u_{trial}(t) = A(t) + F(t, N(t, \Omega))$$

Let $A(t)$ denote a function that satisfies the given initial conditions. The term $N(t, \Omega)$ represents the output of the NN, where Ω is the vector of adjustable weights. By invoking **Theorem (2.1)** and utilizing the weighted power series representation, we construct the trial solution in a form that inherently satisfies the initial conditions.

To achieve this, we define an additional term $F(t, N(t, \Omega))$, which is designed so that it does not affect the initial conditions. Specifically, we set

$$F(t, N(t, \Omega)) = \Psi(t) N(t, \Omega)$$

where $\Psi(t)$ is a differentiable function satisfying

$$\lim_{t \to t_0} \Psi(t) = 0$$

Consequently, as $t \to t_0$, it follows that

$$F(t, N(t, \Omega)) \to 0 \quad \text{and} \quad u_{trial} \to A(t)$$

Therefore, the trial solution $u_{trial}(t)$ automatically satisfies the initial conditions by construction.

5. Gradient Descent Method

The gradient descent (GD) method is an optimization procedure used to minimize or maximize objective functions and is widely applied in various learning techniques. Consider a differentiable function $E(\Omega)$, where $\Omega = (w_0, w_1, ..., w_m)^T$. The goal is to minimize $E(\Omega)$ such that it approaches zero (though not necessarily reaching zero), using the gradient descent method. This involves iteratively updating the variables w_l according to the following form:

$$w_{l+1} = w_l - \eta \Delta E(\Omega)$$

Where l is the step, $\Delta E(\Omega) = \dfrac{\partial E}{\partial w_l}$, w_l is the current position and w_{l+1} is the next value and η is the learning rate.

In [34], an alternative form of the gradient descent method was used, given by:

$$w_{l+1} = w_l + \Delta w_l$$

$$\Delta w_l = -\eta \frac{\partial E}{\partial w_l} + \gamma \Delta w_{l-1}$$

$$\frac{\partial E}{\partial w_l} = \sum_{r=0}^{R} \frac{\partial E_r}{\partial w_l}$$

Where η and γ are the learning rate and momentum term respectively.

Since the gradient descent method is an iterative procedure that operates in the domain of learnable parameters [7], we employ it in one instance to optimize the weights. In another instance, we use the Adam optimization method—described in detail in the following section—to improve the performance of the backpropagation algorithm.

6. Adam Method

While the gradient descent method is the basic method which first introduced in the previous section, Adam method based on the gradient descent method, the momentum method and variation of the interval is defined by the following [35]

The first momentum defined by

$$m_l = \beta_1 m_{l-1} + (1 - \beta_1) \frac{\partial E}{\partial w_l}$$

The second momentum is defined by

$$v_l = \beta_2 v_{l-1} + (1 - \beta_2) (\frac{\partial E}{\partial w_l})^2$$

And finally

$$w_{l+1} = w_l - \eta \frac{\hat{m}_l}{\sqrt{\hat{v}_l} + \epsilon}$$

$$\hat{m}_l = m_l / (1 - \beta_1^{l+1}), \; \hat{v}_l = v_l / (1 - \beta_2^{l+1})$$

Where β_1,β_2 are exponentially decreasing rates for moment estimates, η is the training rate and ϵ is the stability term.

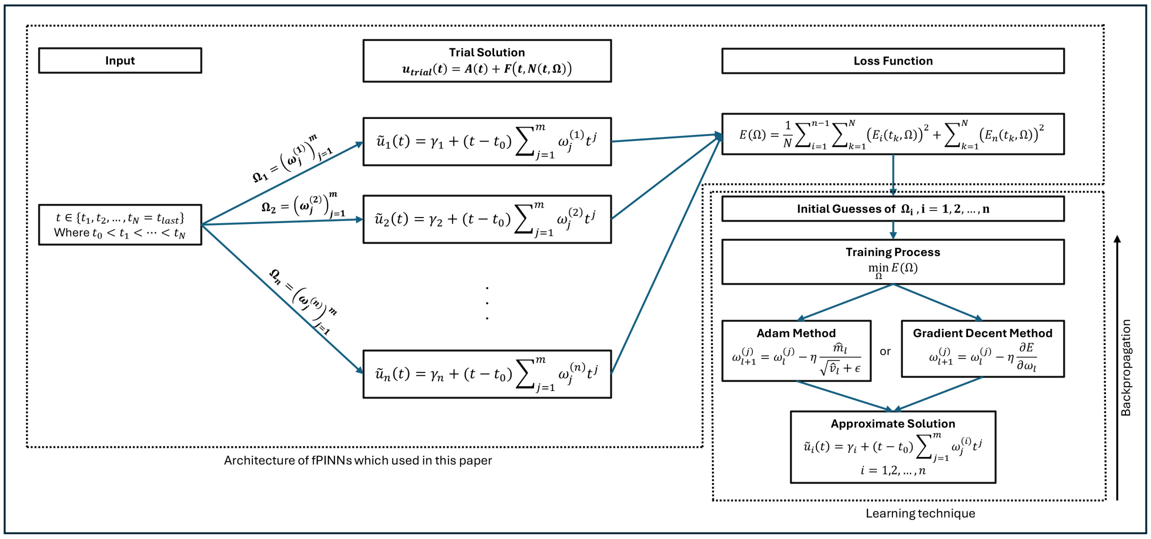
Input
$t \in \{t_1, t_2, \dots, t_N = t_{last}\}$ Where $t_0 < t_1 < \dots < t_N$
Trial Solution $u_{trial}(t) = A(t) + F(t, N(t,\Omega))$
$\Omega_1 = (\omega_j^{(1)})_{j=1}^m$
$\Omega_2 = (\omega_j^{(2)})_{j=1}^m$
$\Omega_n = (\omega_j^{(n)})_{j=1}^m$
$\tilde{u}_1(t) = \gamma_1 + (t-t_0)\sum_{j=1}^m \omega_j^{(1)} t^j$
$\tilde{u}_2(t) = \gamma_2 + (t-t_0)\sum_{j=1}^m \omega_j^{(2)} t^j$
$\tilde{u}_n(t) = \gamma_n + (t-t_0)\sum_{j=1}^m \omega_j^{(n)} t^j$
Loss Function
$E(\Omega) = \frac{1}{N}\sum_{i=1}^{n-1}\sum_{k=1}^{N}(E_i(t_k,\Omega))^2 + \sum_{k=1}^{N}(E_n(t_k,\Omega))^2$
Initial Guesses of Ω_i, $i = 1, 2, \dots, n$
Training Process $\min_\Omega E(\Omega)$
Adam Method $\omega_{l+1}^{(j)} = \omega_l^{(j)} - \eta \frac{\hat{m}_l}{\sqrt{\hat{v}_l} + \epsilon}$
or
Gradient Decent Method $\omega_{l+1}^{(j)} = \omega_l^{(j)} - \eta \frac{\partial E}{\partial \omega_l}$
Approximate Solution $\tilde{u}_i(t) = \gamma_i + (t-t_0)\sum_{j=1}^m \omega_j^{(i)} t^j$ $i = 1,2,\dots,n$
Backpropagation
Architecture of fPINNs which used in this paper
Learning technique

Figure 1: Overview of the fPINNs Architecture Along with the Learning Strategy Implemented in This Paper.

7. The Approach

Let $\tau > 0$ and $L^2[t_0, t_0+\tau]$ denote the space of squared integrable functions on the interval $[t_0, t_0+\tau]$. For $i = 1, 2, ..., n$, let $u_i(t) \in L^2[t_0, t_0+\tau]$, and define the vector functions $U(t) = (u_i(t))_{i=1}^n$ and $\Phi(t) = (\varphi_i(t))_{i=1}^n$, $\varphi_i(t) : \mathbf{R} \to \mathbf{R}$ for all i. We now consider the following FDIDAE

$$ {}_0^cD_t^{\alpha_i} u_i(t) = \sum_{i=1}^{n} \int_0^t k_i(t,s)u_i(s)ds + F_i\big(t, U(t), U(t-\tau)\big) \tag{7.1}$$

$$0 < \alpha_i \le 1,\ i = 1, 2, ..., n-1,\ \tau > 0,\ t > t_0$$

Subject to

$$g(t, U(t)) = 0 \tag{7.2}$$

With initial conditions

$$u_{0,i}(t) = \varphi_i(t),\ i = 1, 2, ..., n,\ t \in [t_0 - \tau, t_0] \tag{7.3}$$

The solution of problem (7.1)-(7.3) on the interval $[t_0, t_0+\tau]$ will be as follows:

First, substituting (7.3) into (7.1) transforms the FDIDAE into FIDAE

$$ {}_0^cD_t^{\alpha_i} u_i(t) = \sum_{i=1}^{n} \int_0^t k_i(t,s)u_i(s)ds + F_i\big(t, U(t), \Phi(t-\tau)\big) \tag{7.4}$$

$$,\ 0 < \alpha_i \le 1,\ i = 1, 2, ..., n-1$$

Subject to

$$g(t, U(t)) = 0$$

And initial conditions $u_i(t_0) = \varphi_i(t_0) = \gamma_i$, $\gamma_i \in \mathbf{R}$, $i = 1, 2, ..., n$, $t \in [t_0, t_0+\tau]$

We now introduce trial solutions $\tilde{u}$ to approximate the unknown functions

$$\tilde{u}_i(t) = A_i(t) + \Psi_i(t) N_i(t, \Omega_i),\ i = 1, 2, ..., n \tag{7.5}$$

Where $\Omega_i = (w_0^{(i)}, w_1^{(i)}, ..., w_m^{(i)})^T$ is the vector of weights to be determined for each $\tilde{u}_i(t), i = 1, 2, ..., n$. The functions $A_i(t)$ are assumed to guarantee the initial conditions, and $\Psi_i(t)$ are differentiable functions such that $\lim_{t\to t_0} \Psi_i(t) = 0$. Moreover, $N_i(t, \Omega_i)$ are smooth functions. Here, the terms $A_i(t) = u_i(t_0) = \gamma_i$ guarantees the initial conditions, and by choosing $\Psi_i(t) = t - t_0$ for all i we ensure that $\lim_{t\to t_0} \Psi_i(t) = 0$ for all i. As pointed out in the previous sections, the choice of $N_i(t, \Omega_i)$, will be a power series, thus,

$$N_i(t, \Omega_i) = \sum_{j=0}^{m} w_j^{(i)} t^j,\ i = 1, 2, ..., n$$

Therefore, the trial solution (7.5) will be as follows

$$\tilde{u}_i(t) = \gamma_i + (t - t_0) \sum_{j=0}^{m} w_j^{(i)} t^j,\quad i = 1, 2, ..., n \tag{7.6}$$

It is clear that (7.6) is always sufficient to enforce the initial conditions, since we have $\lim_{t\to t_0} \tilde{u}_i(t) = \gamma_i$, and in fact $\tilde{u}_i(t_0) = \gamma_i$ for all i.

Next, applying the direct substitution of (7.6) into (7.4), yields:

${}_0^cD_t^{\alpha_i}(\gamma_i+(t-t_0)\sum_{j=0}^m w_j^{(i)} t^j) = \sum_{i=1}^n \int_0^t k_i(t,s)(\gamma_i+(s-t_0)\sum_{j=0}^m w_j^{(i)} s^j)ds + F_i(t, A(t)+\Psi(t)N(t,\Omega), \Phi(t-\tau))$,
$0 < \alpha_i \leq 1,\ i = 1, 2, ..., n-1$

where $A = (A_i)_{i=1}^n$, $\Psi = (\Psi_i)_{i=1}^n$, $N = (N_i)_{i=1}^n$

subject to $g\big(t, \gamma_1 + (t-t_0)\sum_{j=0}^m w_j^{(1)} t^j, \gamma_2 + (t-t_0)\sum_{j=0}^m w_j^{(2)} t^j, ..., \gamma_n + (t-t_0)\sum_{j=0}^m w_j^{(n)} t^j\big)$

Now to construct the loss function, the residuals of the given problem becomes:

$E_i(t,\Omega) = {}_0^cD_t^{\alpha_i}(\gamma_i + (t-t_0)\sum_{j=0}^m w_j^{(i)} t^j) - (\sum_{i=1}^n \int_0^t k_i(t,s)(\gamma_i + (s-t_0)\sum_{j=0}^m w_j^{(i)} s^j)ds + F_i(t, A(t) + \Psi(t)N(t,\Omega), \Phi(t-\tau)))$, $i = 1, 2, ..., n-1$

$E_n(t,\Omega) = g\big(t, \gamma_1 + (t-t_0)\sum_{j=0}^m w_j^{(1)} t^j, \gamma_2 + (t-t_0)\sum_{j=0}^m w_j^{(2)} t^j, ..., \gamma_n + (t-t_0)\sum_{j=0}^m w_j^{(n)} t^j\big)$

It is important to note that, since the initial conditions are embedded within the trial solutions, the residuals associated with these initial conditions may be neglected.

The total loss function employed in this paper is the mean squared error (MSE), defined as follows:

$$E(\Omega) = \frac{1}{N} \sum_{i=1}^{n-1} \sum_{k=1}^{N} (E_i(t_k, \Omega))^2 + \sum_{k=1}^{N} (E_n(t_k, \Omega))^2 \tag{7.7}$$

Where $\Omega = (\Omega_1, \Omega_2, ..., \Omega_n)$, see Algorithm (1)

Algorithm 1 (Solve FDIDAE)
```
1: Transform the FDIDAE into FIDAE, equation (7.4) using (7.3).
2: Assume the trial solutions $\tilde{u}_i(t) = A_i(t) + \Psi_i(t)N_i(t,\Omega_i)$,   $i = 1, 2, ..., n$.
3: Discretize the given interval with equidistant step sizes $\{t_0 < t_1 < t_2, ..., t_N = t_{last}\}$.
4: Construct the loss function $E(\Omega)$.
5: Set the initial values of $w_j^{(i)}$ randomly, for all $j = 1, 2, ..., m$ and $i = 1, 2, ..., n$.
```

```
6:  while Number of iterations < Maximum of iterations do
    Update parameters using Adam method $w_{l+1} = w_l - \eta\frac{\hat{m}_l}{\sqrt{\hat{v}_l}+\epsilon}$ with $\beta_1 = 0.9,\ \beta_2 = 0.999,\ \epsilon = 10^{-8}$.
7:      Substitute the updated parameters into $E(\Omega)$.
8:      if $E(\Omega) \leq \delta,\ \delta > 0$ then
9:          break
10:     end if
11: end while
12: The approximate solutions are $\tilde{u}_i(t) = A_i(t)+\Psi_i(t)N_i(t,\Omega_i),\ i = 1,2,...,n$, and according to equation (2.1), $|u_i(t) - \tilde{u}_i(t)| \leq \zeta,\ \forall\ t \in [t_0, t_{last}]$.
```

8. Illustrative Examples

In this section, an illustrative examples are given to prove the efficiency of the suggested method. All computations are done by MATLAB R2024a on a computer with a 12th Gen Intel Core i9-12900 H processor and 20 CPUs. In all examples we gave the dimension of Ω_i, i.e the value of m, the collocation of the domain interval, the training rate η, and the elapsed time to obtain the wanted result.

Example 8.1 *Consider the following FDIDAE*

$$ {}_0^cD_t^{0.5}u_1(t) = u_2(t) + \int_0^t u_1(s)ds - sin(t) - u_1(t-1) + 1 + \sqrt{t}E_{1,1.5} \tag{8.1}$$

subject to

$$4\big(u_1(t)\big)^2 + e^{2it} + e^{-2it} - 4e^{2t} = 2 - 4\big(u_2(t)\big)^2 \tag{8.2}$$

$t \geq 0$ *with initial conditions*

$$u_1(t) = e^{t+1} \quad on \quad t \in [-1, 0] \tag{8.3}$$

, $u_2(0) = 0$, *where* $E_{1,1.5}$ *is the Mittag-Leffler function. The closed-form solution of the problem given by equations (8.1) and (8.2) is*

$$u_1(t) = e^t, \quad u_2(t) = \sin(t).$$

In Example 8.1, we set $m = 4$ and $N = 10$. The learning rate of the Adam method was chosen as $\eta = 0.001$, and the initial guesses for the weights were selected randomly. The elapsed time to obtain the results was 1.170120 seconds.

Figure (2) illustrates a comparison between the exact and approximate solutions in the t–u plane for both functions, the relative errors of $u_1(t)$ and $u_2(t)$, and the total error convergence. Additionally, Table (1) provides a selection of results, including the exact and approximate values of $u_1(t)$ and $u_2(t)$, along with their corresponding relative errors.

Table 1: The exact and approximate solutions u and $\tilde{u}$ and the relative errors $\tilde{E}$ for example (8.1).

t	u_1	$\tilde{u}_1$	$\tilde{E}_1$	u_2	$\tilde{u}_2$	$\tilde{E}_2$
0.22222	1.2488	1.2492	0.00024684	0.2204	0.21907	0.0060063
0.44444	1.5596	1.5596	3.54e-05	0.42996	0.42904	0.0021366
0.67677	1.9675	1.9677	8.76e-05	0.62628	0.62669	0.00066136
0.86869	2.3838	2.3836	7.71e-05	0.76348	0.76336	0.00016508
1	2.7183	2.7182	2.96e-05	0.84147	0.84208	0.00072278

Example 8.2 *Consider the following FIDAE*

$$ {}_0^cD_t^{0.5}u_1(t) = u_3(t) + \int_0^t (t-s)u_2(s)ds + \sqrt{t}E_{1,1.5} - sin(t) - \frac{t^4}{12} - cos(t) \tag{8.4}$$

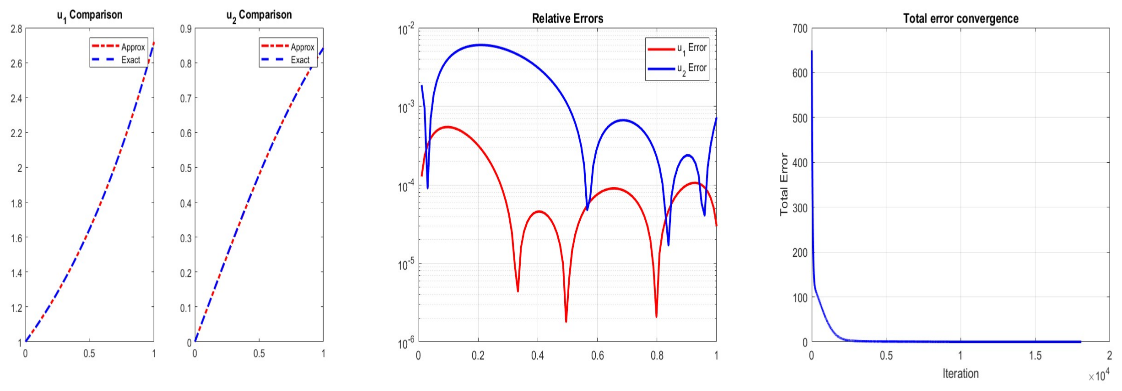

Figure 2: A comparison between the exact and approximate solutions, the relative errors $|u - \tilde{u}|/|u|$ of $u_1(t)$ and $u_2(t)$, and the total error for example (8.1).

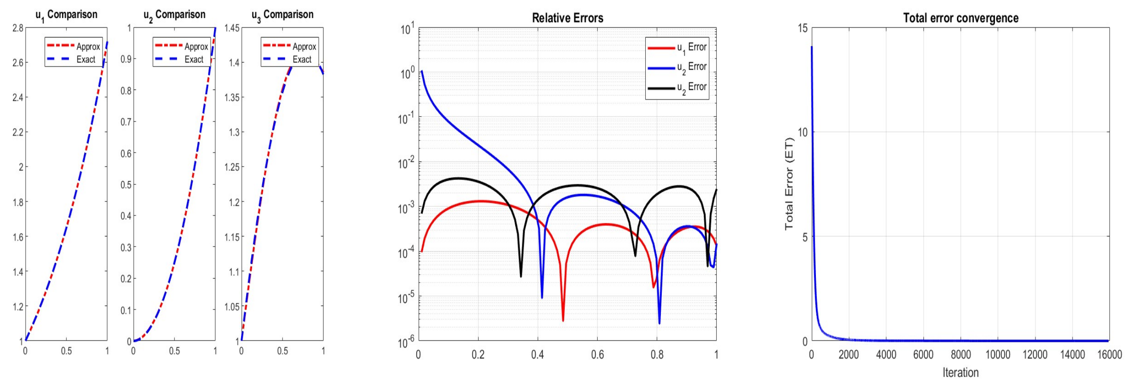

Figure 3: A comparison between the exact and approximate solutions, the relative errors $|u - \tilde{u}|/|u|$ of $u_1(t)$, $u_2(t)$ and $u_3(t)$, and the total error for example (8.2).

$$ {}_0^cD_t^{0.5}u_2(t) = \int_0^t (t-s)u_3(s)ds + \frac{\Gamma(3)}{\Gamma(2.5)} - t - 1 + cos(t) + sin(t) \tag{8.5}$$

subject to

$$u_1(t) + u_2(t) + u_3(t) - e^t - t^2 - cos(t) - sin(t) = 0 \tag{8.6}$$

with $t \in [0,1]$, $u_1(0) = u_3(0) = 1$, $u_2(0) = 0$, *the exact solution of (8.4)-(8.6) is given by* $u_1(t) = e^t$, $u_2(t) = t^2$, $u_3(t) = cos(t) + sin(t)$.

In example (8.2), we set $m = 4$ and $N = 10$, the training rate of the Adam method was $\eta = 0.001$ and the initial guesses of the weights were chosen randomly. The elapsed time to obtain the results was 1.216274 seconds.

Figure (3) illustrates a comparison between the exact and approximate solutions in the $t - u$ plane for both functions, the relative errors of $u_1(t)$, $u_2(t)$ and $u_3(t)$, and the total error convergence. Additionally, Table (2) provides a selection of results, including the exact and approximate solutions for $u_1(t)$, $u_2(t)$ and $u_3(t)$ along with their corresponding relative errors.

Example 8.3 *Consider the following FIDAE*

$$ {}_0^cD_t^{0.5}u_1(t) = \int_0^t u_2(s)ds - \frac{1}{2}(1 - cos(t)) + \sum_{k=0}^{\infty} \frac{\Gamma(k+3)t^{k+1.5}}{k!\Gamma(k+2.5)} \tag{8.7}$$

Table 2: The exact and approximate solutions u and $\tilde{u}$ and the relative errors $\tilde{E}$ for example (8.2).

t	u_1	$\tilde{u}_1$	$\tilde{E}_1$	u_2	$\tilde{u}_2$	$\tilde{E}_2$	u_3	$\tilde{u}_3$	$\tilde{E}_3$
0.22222	1.2488	1.2472	0.0013003	0.049383	0.050273	0.018033	1.1958	1.1921	0.0031346
0.44444	1.5596	1.5593	0.00022395	0.19753	0.19737	0.00080118	1.3328	1.3358	0.0022385
0.67677	1.9675	1.9682	0.00035811	0.45801	0.4575	0.0011338	1.4059	1.4075	0.0011763
0.86869	2.3838	2.3831	0.00027162	0.75462	0.75485	0.00030882	1.4093	1.4054	0.0028003
1	2.7183	2.7179	0.00013044	1	0.99985	0.00014969	1.3818	1.3852	0.0024532

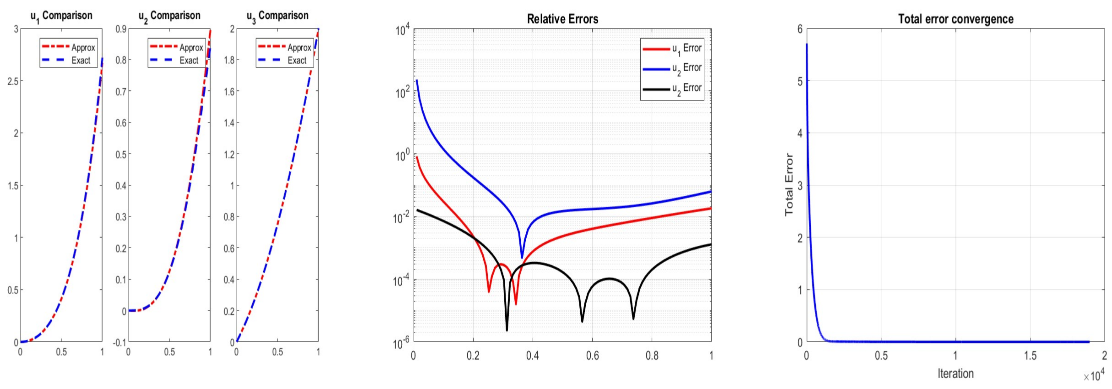

Figure 4: A comparison between the exact and approximate solutions, the relative errors $|u-\tilde{u}|/|u|$ of $u_1(t)$, $u_2(t)$ and $u_3(t)$, and the total error for example (8.3).

$$ {}_{0}^{c}D_t^{0.5}u_3(t) = \int_0^t u_1(s)ds + \frac{\Gamma(2)}{\Gamma(1.5)}\sqrt{t} + \frac{\Gamma(3)}{\Gamma(2.5)}t\sqrt{t} - ((t^2-2t+2)e^t - 2) \tag{8.8}$$

Subject to

$$u_1(t) + u_2(t) + u_3(t) - t^2e^t - t - t^2 - tsin(t^2) = 0 \tag{8.9}$$

With $t \in [0,1]$, $u_1(0) = u_2(0) = u_3(0) = 0$ *and exact solution* $u_1(t) = t^2e^t$, $u_2(t) = tsin(t^2)$ $u_3(t) = t+t^2$.

In example (8.3), we set $m = 4$ and $N = 10$, the training rate of the Adam method was $\eta = 0.001$ and the initial guesses of the weights were chosen randomly. The elapsed time to obtain the results was 1.092856 seconds.

Figure (4) illustrates a comparison between the exact and approximate solutions in the $t-u$ plane for both functions, the relative errors of $u_1(t)$, $u_2(t)$ and $u_3(t)$, and the total error convergence. Additionally, Table (3) provides a selection of results, including the exact and approximate solutions for $u_1(t)$, $u_2(t)$ and $u_3(t)$ along with their corresponding relative errors.

8.1. Comparison

In this subsection, a comparison between the Adam optimizer and the GD method is presented. The results shown in Tables (4), (5), and (6) correspond to Examples 8.1, 8.2, and 8.3, respectively. These results allow for a comparison of the efficiency of the proposed approach when using the Adam optimizer versus GD.

Table 3: The exact and approximate solutions u and $\tilde{u}$ and the relative error $\tilde{E}$ for example (8.3)

t	u_1	$\tilde{u}_1$	$\tilde{E}_1$	u_2	$\tilde{u}_2$	$\tilde{E}_2$
0.25253	0.082088	0.082118	0.00036247	0.016092	0.014711	0.085817
0.47475	0.36233	0.36145	0.0024379	0.1061	0.10819	0.019754
0.69697	0.97525	0.96941	0.0059882	0.32541	0.33175	0.019485
0.87879	1.8596	1.8374	0.011942	0.61318	0.63482	0.035277
1	2.7183	2.669	0.018123	0.84147	0.89422	0.062681

		u_3	$\tilde{u}_3$	$\tilde{E}_3$		
		0.31629	0.3166	0.00095842		
		0.70013	0.70003	0.00014358		
		1.1827	1.1826	0.0001507		
		1.6511	1.6502	0.00050309		
		2	1.9975	0.0012505		

To ensure a fair comparison, the initial values of the weights were fixed as $\Omega = (0.1, 0.2, \ldots, 1)$ for Example 8.1, and $\Omega = \left(\frac{1}{15}, \frac{2}{15}, \ldots, 1\right)$ for Examples 8.2 and 8.3.

Table 4: A comparison of the relative errors $\tilde{E}$ of u_1 and u_2 between Adam method and GD for example (8.1).
Note: The elapsed time to obtain the results using Adam method was 1.393165 seconds while the elapsed time using GD was 101.998194 seconds.

t	**Adam** $\tilde{E}_1$	**GD** $\tilde{E}_1$	**Adam** $\tilde{E}_2$	**GD** $\tilde{E}_2$
0	0	0	Undefined	Undefined
0.11111	5.78e-04	2.79e-04	5.05e-02	2.58e-02
0.22222	1.02e-03	4.49e-04	3.81e-02	1.84e-02
0.33333	1.15e-03	4.76e-04	2.57e-02	1.16e-02
0.44444	9.15e-04	3.62e-04	1.42e-02	5.78e-03
0.55556	4.04e-04	1.49e-04	4.36e-03	1.31e-03
0.66667	2.03e-04	8.94e-05	2.96e-03	1.55e-03
0.77778	6.44e-04	2.56e-05	6.66e-03	2.67e-03
0.88889	6.07e-04	2.40e-04	5.44e-03	1.95e-03
1	2.34e-04	6.85e-05	2.31e-03	5.82e-04

Table 5: A comparison of the relative errors $\tilde{E}$ of u_1, u_2 and u_3 between Adam method and GD for example (8.2).
Note: The elapsed time to obtain the results using Adam method was 1.018777 seconds while the elapsed time using GD was 59.160156 seconds.

t	**Adam** $\tilde{E}_1$	**GD** $\tilde{E}_1$	**Adam** $\tilde{E}_2$	**GD** $\tilde{E}_2$	**Adam** $\tilde{E}_3$	**GD** $\tilde{E}_3$
0	0	0	Undefined	Undefined	0	0
0.11111	4.02e-04	4.69e-04	4.51e-01	2.01e-01	5.53e-03	2.98e-03
0.22222	9.92e-04	6.25e-04	1.15e-01	4.71e-02	5.77e-03	2.54e-03
0.33333	1.37e-03	5.65e-04	2.74e-02	9.20e-03	3.45e-03	7.83e-04
0.44444	1.36e-03	3.90e-04	8.21e-05	1.53e-03	5.05e-04	8.74e-04
0.55556	9.59e-04	1.85e-04	5.86e-03	3.03e-03	1.75e-03	1.57e-03
0.66667	3.19e-04	1.53e-05	3.48e-03	1.44e-03	2.52e-03	1.03e-03
0.77778	2.84e-04	7.34e-05	6.43e-04	5.13e-04	1.59e-03	3.92e-04
0.88889	4.93e-04	5.24e-05	2.79e-03	1.35e-03	5.45e-04	1.45e-03
1	9.26e-05	9.42e-05	4.25e-04	1.75e-04	2.43e-03	3.11e-04

Table 6: A comparison of the relative errors $\tilde{E}$ of u_1, u_2 and u_3 between Adam method and GD for example (8.3).
Note: The elapsed time to obtain the results using Adam method was 1.592705 seconds while the elapsed time using GD was 161.459063 seconds.

t	**Adam** $\tilde{E}_1$	**GD** $\tilde{E}_1$	**Adam** $\tilde{E}_2$	**GD** $\tilde{E}_2$	**Adam** $\tilde{E}_3$	**GD** $\tilde{E}_3$
0	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
0.11111	5.39e-03	6.31e-03	1.41e-01	4.37	9.50e-04	1.17e-02
0.22222	4.46e-04	9.64e-04	1.10e-01	4.67e-01	1.50e-04	4.67e-03
0.33333	1.72e-04	2.97e-03	3.09e-02	2.45e-02	8.74e-05	1.07e-03
0.44444	9.99e-04	3.72e-03	9.25e-04	4.23e-02	7.49e-05	3.70e-04
0.55556	2.51e-03	4.45e-03	8.05e-03	3.93e-02	1.55e-05	6.16e-04
0.66667	4.78e-03	5.87e-03	1.57e-02	2.65e-02	3.93e-05	3.59e-04
0.77778	8.00e-03	8.45e-03	2.49e-02	2.14e-02	2.26e-04	1.26e-04
0.88889	1.24e-02	1.25e-02	3.86e-02	3.19e-02	6.21e-04	3.19e-04
1	1.80e-02	1.83e-02	6.14e-02	6.55e-02	1.24e-03	1.25e-03

9. Conclusions

In this paper, a novel numerical approach using fractional physics-informed neural networks (fPINNs) method was presented in order to solve fractional order delay integro-differential algebraic equations. The trial solution that we constructed was efficient to approximate the exact solution and it is easy for fractional order differentiation. For the learning technique we used Adam method and this decreases the iterations of the learning process and the elapsed time, while the Gradient Descent method increases them. We conclude that the proposed approach is very powerful and solving the given problem efficiently. The approximate results generated by the proposed method closely align with the analytical solutions.

References

1. Dalir, Mehdi and Bashour, Majid. *Applications of fractional calculus.* Applied Mathematical Sciences 4, 21, 1021–1032, (2010).
2. Hilfer, Rudolf. *Applications of Fractional Calculus in Physics*, World scientific (2000).
3. Tenreiro Machado, JA and Silva, Manuel F and Barbosa, Ramiro S and Jesus, Isabel S and Reis, Cecília M and Marcos, Maria G and Galhano, Alexandra F. *Some applications of fractional calculus in engineering.* Mathematical problems in engineering, 2010, 1, 639801. Wiley Online Library (2010).
4. Sun, HongGuang and Zhang, Yong and Baleanu, Dumitru and Chen, Wen and Chen, YangQuan. *A new collection of real world applications of fractional calculus in science and engineering.* Communications in Nonlinear Science and Numerical Simulation 64, 213–231. Elsevier (2018).
5. Tarasov, Vasily E. *On history of mathematical economics: Application of fractional calculus.* Mathematics, 7, 6, 509. MDPI (2019).
6. Arora, Sugandha and Mathur, Trilok and Agarwal, Shivi and Tiwari, Kamlesh and Gupta, Phalguni. *Applications of fractional calculus in computer vision: a survey.* Neurocomputing 489, 407–428. Elsevier (2022).
7. Singh, Abhaya Pal and Bingi, Kishore. *Applications of fractional-order calculus in robotics.* Fractal and Fractional 8, 7, 403. MDPI (2024).
8. Miranda-Valdez, Isaac Y and Puente-Córdova, Jesús G and Rentería-Baltiérrez, Flor Y and Fliri, Lukas and Hummel, Michael and Puisto, Antti and Koivisto, Juha and Alava, Mikko J. *Viscoelastic phenomena in methylcellulose aqueous systems: Application of fractional calculus.* Food Hydrocolloids 147, 109334. Elsevier (2024).
9. Meng, Ruifan. *Application of fractional calculus to modeling the non-linear behaviors of ferroelectric polymer composites: viscoelasticity and dielectricity.* Membranes, 11, 6, 409. MDPI (2021).
10. Cheow, Yi Herng and Ng, Kok Haur and Phang, Chang and Ng, Kooi Huat. *The application of fractional calculus in economic growth modelling: An approach based on regression analysis.* Heliyon, 10, 15. Elsevier (2024).
11. Farhood, Adnan Khalaf and Mohammed, Osama H and Taha, Bushra A. *Solving fractional time-delay diffusion equation with variable-order derivative based on shifted Legendre–Laguerre operational matrices.* Arabian Journal of Mathematics, 12, 3, 529–539. Springer (2023).
12. Khan, Hasib and Ahmed, Saim and Alzabut, Jehad and Azar, Ahmad Taher. *A generalized coupled system of fractional differential equations with application to finite time sliding mode control for Leukemia therapy.* Chaos, Solitons & Fractals, 174, 113901. Elsevier (2023).

13. Mohammed, Osama H and Mohsin, Ahmed K. *Approximate methods for solving one-dimensional partial integro-differential equations of fractional order.* Italian Journal of Pure and Applied Mathematics, 205, (2021).

14. Mohammed, OH and Jaleel, DHA. *Legendre-adomian-homotopy analysis method for solving multi-term nonlinear differential equations of fractional order.* Italian Journal of Pure and Applied Mathematics, 581, (2021).

15. Farhood, Adnan K and Mohammed, Osama H. *Homotopy perturbation method for solving time-fractional nonlinear Variable-Order Delay Partial Differential Equations.* Partial Differential Equations in Applied Mathematics, 7, 100513. Elsevier (2023).

16. Farhood, Adnan K and Mohammed, Osama H. *Shifted Chebyshev operational matrices to solve the fractional time-delay diffusion equation.* Partial Differential Equations in Applied Mathematics, 8, 100538. Elsevier (2023).

17. Abdulhameed, Noor A and Mohammed, Osama H and Yousif, Ahmed A. *A hybrid technique for approximating the solution of fractional order integro differential equations.* Partial Differential Equations in Applied Mathematics, 8, 100552. Elsevier (2023).

18. Yousif, Ahmed A and AbdulKhaleq, Fajir A and Mohsin, Ahmed K and Mohammed, Osama H and Malik, Adyan M. *A developed technique of homotopy analysis method for solving nonlinear systems of Volterra integro-differential equations of fractional order.* Partial Differential Equations in Applied Mathematics, 8, 100548. Elsevier (2023).

19. Mohammed, Hasnaa F and Mohammed, Osama H. *A hybrid technique for solving fractional delay variational problems by the shifted Legendre polynomials.* Partial Differential Equations in Applied Mathematics, 9, 100635. Elsevier (2024).

20. Salim, Huda A and Abdali, Bashaer M and Abdulkhaleq, Fajir A and Mohammed, Osama H. *Perturbation iteration transform method for solving fractional order integro-differential equation.* Partial Differential Equations in Applied Mathematics, 11, 100874. Elsevier (2024).

21. Amin, Rohul and Shah, Kamal and Asif, Muhammad and Khan, Imran and Ullah, Faheem. *An efficient algorithm for numerical solution of fractional integro-differential equations via Haar wavelet.* Journal of Computational and Applied Mathematics, 381, 113028. Elsevier (2021).

22. Brugnano, Luigi and Burrage, Kevin and Burrage, Pamela and Iavernaro, Felice. *A spectrally accurate step-by-step method for the numerical solution of fractional differential equations.* Journal of Scientific Computing, 99, 2, 48. Springer (2024).

23. Shiri, Babak and Baleanu, Dumitru. *System of fractional differential algebraic equations with applications.* Chaos, Solitons & Fractals, 120, 203–212. Elsevier (2019).

24. Xing, Baixue and Liu, Haixia and Liu, Hongliang and Tang, Xiao. *Chebyshev Neural Network Method for Solving Delay-Integro-Differential-Algebraic Equations Based on Variable Transformation*, (2023).

25. Cao, Wenbo and Zhang, Weiwei. *An analysis and solution of ill-conditioning in physics-informed neural networks.* Journal of Computational Physics, 520, 113494. Elsevier (2025).

26. Karniadakis, George Em and Kevrekidis, Ioannis G and Lu, Lu and Perdikaris, Paris and Wang, Sifan and Yang, Liu. *Physics-informed machine learning.* Nature Reviews Physics, 3, 6, 422–440. Nature Publishing Group (2021).

27. Ruder, Sebastian. *An overview of gradient descent optimization algorithms.* arXiv preprint arXiv:1609.04747 (2016).

28. Qu, Haidong and She, Zihang and Liu, Xuan. *Homotopy analysis method for three types of fractional partial differential equations.* Complexity, 2020, 1, 7232907. Wiley Online Library (2020).

29. Aitbrahim, Aabdessamad and El Ghordaf, J and El Hajaji, A and Hilal, K and Valdes, JE Nápoles. *A comparative analysis of Conformable, non-conformable, Riemann-Liouville, and Caputo fractional derivatives.* European Journal of Pure and Applied Mathematics, 17, 3, 1842–1854, (2024).

30. Wenqiang Yang and Wenyuan Wu and Greg Reid. *Structural Analysis by Modified Signature Matrix for Integro-differential-algebraic Equations*, (2023).

31. Rihan, Fathalla A and others. *Delay Differential Equations and Applications to Biology.* Springer (2021).

32. Aghaei, Alireza Afzal and Moghaddam, Mahdi Movahedian and Parand, Kourosh. *PINNIES: An Efficient Physics-Informed Neural Network Framework to Integral Operator Problems.* arXiv preprint arXiv:2409.01899, (2024).

33. Westrin, Mimmi. *Solving Ordinary Differential Equations and Systems using Neural Network Methods*, (2023).

34. Shloof, AM and Senu, Norazak and Ahmadian, Ali and Pakdaman, Morteza and Salahshour, Soheil. *A new iterative technique for solving fractal-fractional differential equations based on artificial neural network in the new generalized Caputo sense.* Engineering with Computers, 39, 1, 505–515. Springer (2023).

35. Yi, Dokkyun and Ahn, Jaehyun and Ji, Sangmin. *An effective optimization method for machine learning based on ADAM.* Applied Sciences, 10, 3, 1073. MDPI (2020).

36. Raissi, Maziar and Perdikaris, Paris and Karniadakis, George E. *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. Elsevier (2019).

37. Pang, Guofei and Lu, Lu and Karniadakis, George Em. fPINNs: Fractional physics-informed neural networks. SIAM Journal on Scientific Computing, 41, 4, A2603–A2626. SIAM (2019).

38. Pérez, Dilcia and Quintana, Yamilet. *A survey on the Weierstrass approximation theorem*. arXiv preprint math/0611038, (2006).

Haitham M. Khazeem,
Department of Mathematics and Computer Applications,
College of Science, Al-Nahrain University,
Baghdad, Iraq.
E-mail address: haithem.mms23@ced.nahrainuniv.edu.iq

and

Osama H. Mohammed,
Department of Mathematics and Computer Applications,
College of Science, Al-Nahrain University,
Baghdad, Iraq.
E-mail address: osama.hameed@nahrainuniv.edu.iq