Journal of Engineering and Basic Sciences

e-ISSN: 3023-6460

2026, 05, 1836617, 03.04.2026 (https://doi.org/10.54709/joebs.1836617)

RESEARCH

Neural Network Solutions for First-Order Differential Equations: A Physics-Informed Perspective

K. J. Audu[1*] A. O. Kayode[1 2] S. O. Fajuyi[1] M. O. Inuolaji[1] Y. A. Yahaya[2]

[1] Department of Mathematics, Federal University of Technology, Minna, Nigeria
[2]Department of Mathematics, North-Eastern University Gombe, Nigeria

ABSTRACT

In this study, a Physics-Informed Neural Network (PINN) model was developed and used to solve first-order Ordinary Differential Equations (ODEs). The proposed model implements the initial conditions and incorporates physical laws through its differential equation into the neural network training process to further improve its accuracy and solution. The main interest of this work is to test accuracy, evaluate, and compare the performance of this model with established Artificial Neural Network (ANN) solutions in solving first-order ODEs. We validate the effectiveness and accuracy of these models through the conclusions drawn from the six numerical tests carried out after close evaluation of the results presented by the models which show that the developed PINN model consistently achieves high accuracy with absolute errors in the range of 10^{-8} and 10^{-10} compared to the established ANN model. This also illustrates their weakness and lets researchers make wise decisions in selecting the suitable method of addressing certain ODE problems.

Keywords: First-Order ODEs, Physics-Informed Neural Network, Artificial Neural Network, Performance, Accuracy

Received :05.12.2025
Accepted :12.02.2026

How to cite this article: K. J. Audu, A. O. Kayode, S. O. Fajuyi, M. O. Inuolaji, Y. A. Yahaya, Neural Network Solutions for First-Order Differential Equations: A Physics-Informed Perspective, Toros University Journal of Engineering and Basic Sciences, JOEBS, 2026, 05, 1836617

1. INTRODUCTION

Ordinary Differential Equations (ODEs) are equations which include functions and derivatives of functions with respect to their variables. In different fields, ODEs are important in the modeling of dynamic systems. [1-3]. Among the types of ODEs, first-order equations represent the most basic and widely applicable differential equation which model various phenomena such as population growth, chemical reactions, electrical circuits and mechanics [4-7]. The first order ODEs are those equations where a function and the first derivative are involved. Conventional numerical approaches, such as Runge-kutta methods are widely used when solving equations involving first-order ODE due to their simplicity and stability [8-10]. Nevertheless, these approaches are applicable but may be computationally expensive when dealing with complicated nonlinear ODEs and can regularly face difficulties with nonlinear problems, constrained datasets, restrictions and other constraints. [11-15].

The limitation of these traditional methods led to the rise of Artificial Intelligence particularly Neural Network to obtain absolute solutions to these equations [16-19]. The importance of neural network techniques in providing solutions to problems of ODEs cannot be overstated, so, the demand for an accurate method is required by researchers. Artificial Neural Networks (*ANNs*) model is effective but only it learns from the data and does not necessarily validate the solution of the equation which can alter accuracy [20-22]. To overcome this limitation, this study develops a Physics-Informed Neural Networks (*PINNs*) as an alternative to addresses these problems by embedding conser-

*k.james@futminna.edu.ng

vation laws, such as the residuals of the governing equations and the initial conditions, as constraints within the neural network's loss function and utilizing automatic differentiation techniques for precise solutions which enables the network to model in respect to the physical constraints during training and by this improve its accuracy, efficiency, generalization and consistency. This paper provides researchers with guidance when choosing between the ANN model and the developed PINN model for solving first-order ODEs of the form;

$$f(n, \underline{m}, \underline{m}') = 0 \quad (1)$$

where f is the function that defines the relationship between the independent variable n, the dependent variable $\underline{m}$ and its first derivative $\underline{m}'$.

Numerous studies have been done regarding the PINNs and ANNs such as solving ordinary differential equations using an optimization technique based on training improved artificial neural networks [23], a novel method for solving ordinary differential equations with artificial neural networks [24, 25] carried out an overview on artificial neural network, using of multilayer neural networks for solving systems of differential equations [26, 27] conducted research on utilizing the artificial neural network approach for the resolution of first-order ordinary differential equations and [28] conducted a review of Physics-Informed Neural Network. Although considerable research has been carried out in respect to first-order ODEs, there remains a need for further investigation into the application of the ANN and PINN model [29-32]. But, in spite of the vast application of these methods, a literature gap can be found where the extensive comparative studies have not been conducted to assess their performance over a variety of problems on first-order ODEs. Previous studies have primarily focused on one model which brings uncertainty on the model that is more appropriate to solve first-order ODEs. Therefore, this study seeks to bridge the gap by developing a PINN model for solving first-order ODE prioritizing on validating its capacity in embedding physical laws into the neural network while maintaining accuracy and conducting a comparison between the PINN approach with the established ANN approach.

The proposed study will provide a clear understanding of circumstances in which each of the models performs better through a series of numerical experiments. The results obtained during such assessments are supposed to guide the practitioners and researchers to make sound decision on the choice of the best model to apply in their respective applications.

2. METHODOLOGY

In this section, we described how the models are being used to obtain the solution to the general form of a first-order ODEs equation in the form (1) with its initial condition specified as:

$$\underline{m}(n_0) = \underline{m}_0 \quad (2)$$

2.1 Description of Artificial Neural Network

A computational model based on the structure and functions of biological neural networks is known as an *ANN* [33-35]. The neural network contains the input layer as $n \in \mathrm{R}$, the hidden layer which comprises of the weight and bias parameters with the activation function and output $\hat{\underline{m}}(n) \in \mathrm{R}$ of the information that goes through the network. The neural network architecture is shown in Figure 1.

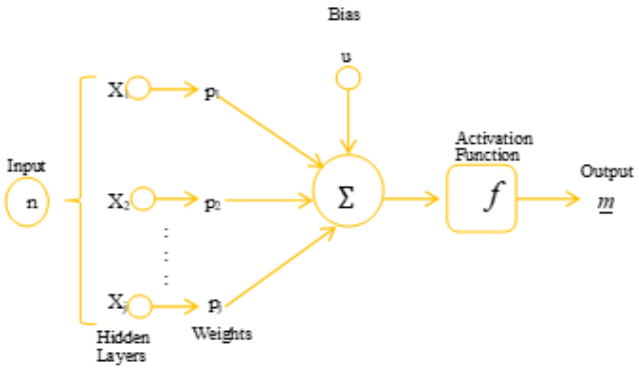

Figure 1: Architecture of an Artificial Neural Network

An ANN with j^{th} hidden layers is equivalently expressed as:

$$\begin{aligned} x^{(1)} &= \sigma\left(P^{(1)}x^{(1)} + u^{(1)}\right) \\ x^{(2)} &= \sigma\left(P^{(2)}x^{(2)} + u^{(2)}\right) \\ &\mathrm{M} \\ x^{(j)} &= \sigma\left(P^{(j)}x^{(j-1)} + u^{(j)}\right) \\ \hat{\underline{m}} &= P^{(j+1)}x^{(j)} + u^{(j+1)} \end{aligned} \quad (3)$$

where

$x^{(1)}, x^{(2)}$ and $x^{(j)}$ are the 1st, 2nd and j^{th} hidden layers respectively

σ is the non-linear activation function (tanh) applied on each nodes

$P^{(j)}$ is the weight parameter acting on the j^{th} layer

$\hat{\underline{m}}$ is the predicted output of the neural network

$u^{(j)}$ is the bias tensor of the j^{th} layer

2.2 Description of the Proposed PINN-FODE

The developed PINN model used to solve (1) is formulated as:

$$NN(n,\underline{m}) = \sigma\ (Pn\ +\ \mathrm{u}) \qquad (4)$$

where the neural network is denoted as NN, n is the input of the network, σ is the activation function of the neural network, P is the weight parameter of the function and u denotes the bias that acts on each neuron of the neural network. The architecture of the developed PINN model used to solve first-order ODEs in this paper is diagrammatically displayed in Figure 2.

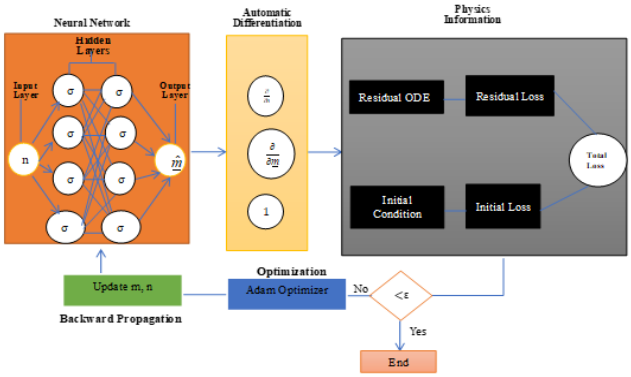

Figure 2:.The Developed PINN-FODE Model Architecture

The developed PINN model architecture shown in Figure 2 takes the independent variable n as an input into the input layer, and then passes it into the hidden layers each composed of neurons, and these neurons in the hidden layer apply a non-linear activation function which aids the weight P and bias u to act on the input and approximate the solution of the first-order ODE through the training process which then gives the solution as output $\underline{m}$ through the output layer. During the process of training the neural network, the developed model integrates the physical laws into its loss function through the differential equation which allows the network to learn solution that satisfies both the data and physical laws enabling better accuracy. The parameter used to determine the difference between the predicted and actual output during training is the loss function.

2.3 The Loss Function

The loss function is obtained using the Mean Square Error (MSE) which is mathematically represented as:

$$MSE = \frac{1}{K}\sum_{i=1}^{K}\left(\underline{m}_i - \underline{\hat{m}}_i\right)^2 \qquad (5)$$

where $\underline{m}_i$ is the actual output , $\underline{\hat{m}}_i$ is the predicted output and K is the number of nodes. The total loss function applied during training is denoted as:

$$L_{Total}(\theta) = L_{ODE}(\theta) + L_{IC}(\theta) \qquad (6)$$

where θ is the parameter of the neural network, L_{ODE} is the ODE residual loss and the L_{IC} represent the initial condition loss. The primary notion of computing the PINNs loss function is to define a residual loss function from the first-order ODE $\frac{d\underline{m}}{dn} = f\left(n,\underline{m}(n)\right)$. The residual loss function is defined as:

$$Loss_{FODE} = R(n;\theta) = \frac{1}{K}\sum_{i=1}^{K}\left(\frac{d\underline{\hat{m}}}{dn}(n_i) - f(n_i,\underline{\hat{m}}(n_i))\right)^2 \qquad (7)$$

where K is the number of residual points, $\frac{d\underline{\hat{m}}}{dn}(n_i)$ represents the predicted output and $f(n_i,\underline{\hat{m}}(n_i))$ is the actual output of the ODE. This is implemented in the neural network while computing the loss function with its initial conditions as part of the PINN's loss function. For a first-order ODE with an initial condition (2), this is incorporated as a MSE term denoted as:

$$Loss(\theta)_{IC} = \frac{1}{K}\sum_{i=1}^{K}\left(\underline{\hat{m}}(n_0) - \underline{m}(n_0)\right)^2 \qquad (8)$$

where K is the number of initial points, $\underline{\hat{m}}(n_0)$ is the prediction of the initial condition in the network and $\underline{m}(n_0)$ is the actual initial condition.

2.4 Training Process of the Model

In this section, the training of the developed PINN model is discussed with its structure, which is an iterative optimization procedure to reduce the loss. The developed PINN model takes n as the input into the input layer, then transfers it to the hidden layer and yields an output $\underline{\hat{m}}$ which is the solution of the network through the output layer. The network output is then differentiated using the automatic differentiation which is then implemented in the residual ODE and initial condition to obtain the total loss of the network. Fig. 3 illustrates the training process of the developed PINN model for solving first-order ODEs.

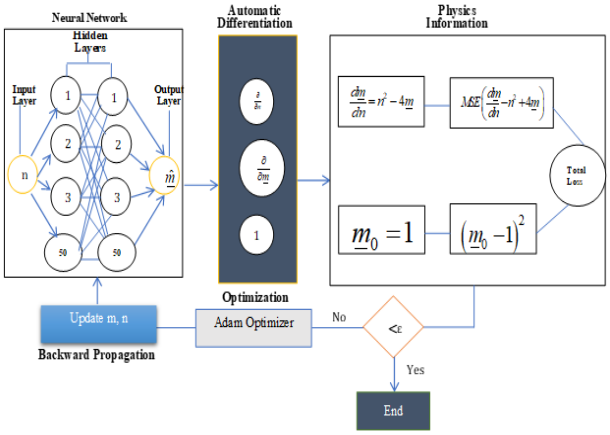

Figure 3. The Developed PINN Model Training Process

As shown in Figure 3 above, the network uses the Adam optimizer to optimize the parameters during the backward propagation before transferring it back into the neural network to retrain in other to obtain an absolute solution. The Adam optimizer used is mathematically expressed as:

$$\theta_{j+1} = \theta_j - \frac{\eta}{\sqrt{n_{j+1} + \omega}} \times \left[\hat{\underline{m}}_{j+1}\right] \tag{9}$$

such that

$$\underline{m}_{j+1} = \alpha_1 . \underline{m}_j + \left(1 - \alpha_1\right) . \nabla Loss(\theta)$$
$$n_{j+1} = \alpha_1 . n_j + \left(1 - \alpha_2\right) . \left(\nabla Loss(\theta)\right)^2 \tag{10}$$

and the bias correction denoted as:

$$\underline{m}_{j+1} = \frac{\underline{m}_{j+1}}{1 - \underline{m}_1^{j+1}}$$
$$n_{j+1} = \frac{n_{j+1}}{1 - \alpha_2^{j+1}} \tag{11}$$

where θ_j is the model parameter, η is the learning rate, ω is a constant that prevents division by zero and $\nabla Loss(\theta)$ is the gradient of the loss with respect to network parameters. The construction of the loss function forms the framework of the developed PINN-FODE model algorithm. In the case of a first-order ODE (1) with initial condition (2), the neural network is trained to reduce the errors. The algorithm of solving these equations with the developed PINN model is described in Algorithm 1:

Algorithm for the Proposed PINN-FODE

1: The first-order ODE problem is defined along with its initial condition.

2: Configure the neural network from the input to the output with the non-linear activation function.

3: Generate training data points $n \in [0,1]$.

4: Define the loss functions:

Loss residual obtained from first- order ODEs (FODE):

$$Loss_{FODE} = R(n;\theta) = \frac{1}{K}\sum_{i=1}^{K}\left(\frac{d\hat{\underline{m}}}{dn}(n_i) - f\left(n_i, \hat{\underline{m}}(n_i)\right)\right)^2$$

Loss obtained from the constraints (initial criteria):

$$Loss(\theta)_{initial\ criteria} = \frac{1}{K}\sum_{i=1}^{K}\left[\hat{\underline{m}}(n_0) - \underline{m}(n_0)\right]^2$$

Total loss:

$$L_{Total}(\theta) = L_{FODE}(\theta) + L_{initial\ criteria}(\theta)$$

5: Train and optimize the network hyper-parameters using the Adam optimizer.

Optimizer (Adam):

$$\theta_{j+1} = \theta_j - \frac{\eta}{\sqrt{n_{j+1} + \omega}} \times \left[\hat{\underline{m}}_{j+1}\right]$$

6: Evaluate the trained model over domain

3. NUMERICAL EXPERIMENTS AND RESULTS

In this section, we present six numerical tests involving first-order ODEs. The results obtained from the developed Physics-Informed Neural Network model were compared with the established solutions obtained from [27] using the Artificial Neural Network model. All computations were done using Python software and the results are tabulated. To evaluate the models' accuracy, we consider the *MAE* (Mean Absolute Error) and the *RMSE* (Root Mean Squared) represented as:

$$MAE = \frac{1}{K}\sum_{i=1}^{K}\left|\underline{m}_i - \hat{\underline{m}}_i\right| \tag{12}$$

$$RSME = \sqrt{\frac{1}{K}\sum_{i=1}^{K}\left(\underline{m}_i - \hat{\underline{m}}_i\right)} \tag{13}$$

where K represent the total number of the data sets, $\underline{m}_i$ is the actual output of the network and $\hat{\underline{m}}_i$ is the predicted output of the network.

Test 1: We consider the developed PINN model to solve the first-order ODE

$$\frac{d\underline{m}}{dn} = n^2 - 4\underline{m}$$

Given the conditions (initial): $n_0 = 0, \quad \underline{m}_0 = 1$

Real Solution: $\frac{31}{32}e^{-4n}+\frac{1}{4}n^2+\frac{1}{8}n+\frac{1}{32}$

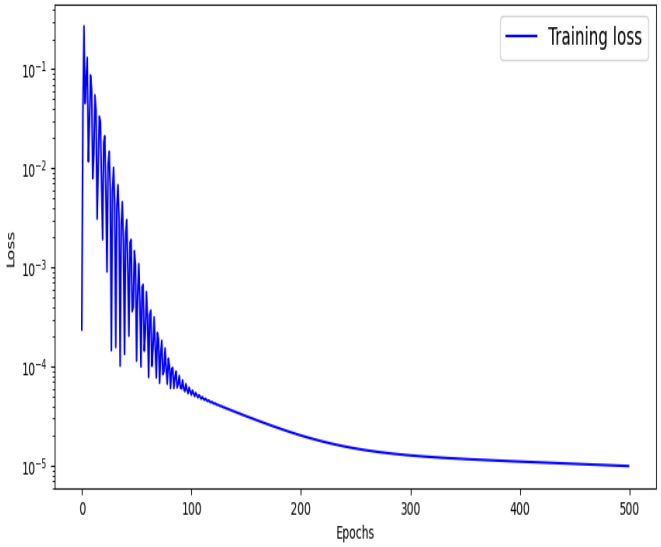

Figure 4: Training Loss Visual Representation for Test 1

Table 1: Computed Solution for Test 1

n	Real Solution	PINN Solution	ANN Solution
0.0	1.0000000000	1.0000000000	0.9975615740
0.1	0.6706225276	0.6706225406	0.6668983698
0.2	0.4515374005	0.4515374287	0.4484636486
0.3	0.3080318868	0.3080318792	0.3070761859
0.4	0.2168372571	0.2168372476	0.2169048488
0.5	0.1623560488	0.1623560556	0.1619767845
0.6	0.1341329962	0.1341330130	0.1327966154
0.7	0.1251597404	0.1251597483	0.1234643161
0.8	0.1307384074	0.1307383865	0.1297434270
0.9	0.1477198452	0.1477198509	0.1479196250
1.0	0.1739932895	0.1739932752	0.1742707193

Table 2: Computed Error for Test 1

n	Real Solution	|Real - PINN|	|Real - ANN|
0.0	1.0000000000	0.0000e+00	2.43843e-03
0.1	0.6706225276	1.30154e-08	3.72416e-03
0.2	0.4515374005	2.82409e-08	3.07375e-03
0.3	0.3080318868	7.64130e-09	9.55701e-04
0.4	0.2168372571	9.55564e-09	6.75917e-05
0.5	0.1623560488	6.81307e-09	3.79264e-04
0.6	0.1341329962	1.68388e-08	1.33638e-03
0.7	0.1251597404	7.84701e-09	1.69542e-03
0.8	0.1307384074	2.08741e-08	9.94980e-04
0.9	0.1477198452	5.72083e-09	1.99780e-04
1.0	0.1739932895	1.42972e-08	2.77430e-04

Table 3: ANN-FODE (Audu *et al*, 2024) Model Performance Evaluation for Test 1

Number of Nodes	MAE	RSME	Time(s)
40	5.57465e-04	7.15516e-04	9.82
80	5.50018e-04	8.03820e-04	3.76
160	7.50821e-04	1.11994e-03	4.50
320	6.23421e-04	9.85016e-04	3.99
500	6.79855e-04	9.82182e-04	4.27

Table 4: PINN-FODE Model Performance Evaluation for Test 1

Number of Nodes	MAE	RSME	Time(s)
40	1.23075e-08	1.50032e-08	7.32
80	3.18291e-08	3.40456e-08	11.12
160	1.01325e-08	1.15880e-08	9.04
320	1.32519e-08	1.53966e-08	14.15
500	1.03509e-08	1.15252e-08	31.80

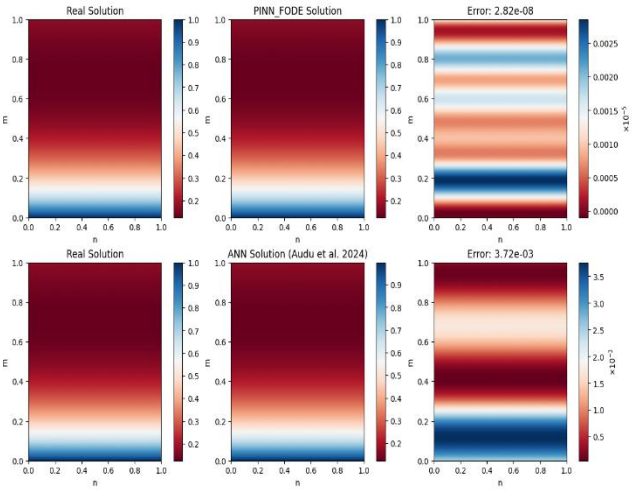

Figure 5: Heat-Map İllustration for Test 1

Test 2: We employ the developed PINN model approach to obtain solution to the first-order ODE

$$\frac{d\underline{m}}{dn}=e^{-2n}-5\underline{m}$$

Obeying the conditions (initial): $n_0=o, \underline{m}_0=1$,

Real Solution: $\frac{1}{3}\left(e^{-2n}+2e^{-5n}\right)$

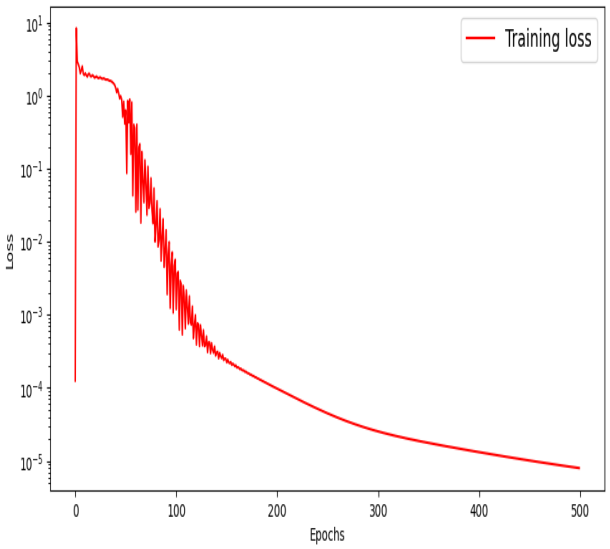

Figure 6: Training Loss Visual Representation for Test 2

Table 5: Computed Solution for Test 2

n	Real Solution	PINN Solution	ANN Solution
0.0	1.0000000000	1.0000000000	0.9966232181
0.1	0.6772640944	0.6772640203	0.6730039120
0.2	0.4686929882	0.4686929711	04664441943
0.3	0.3316906691	0.3316906389	0.3313499391
0.4	0.2399998605	0.2399998391	0.2397259176
0.5	0.1773498058	0.1773498128	0.1765649021
0.6	0.1335894465	0.1335894408	0.1327142864
0.7	0.1023305878	0.1023305801	0.1018679067
0.8	0.0795092732	0.0795092629	0.0795983225
0.9	0.0625056326	0.0625056306	0.0628914014
1.0	0.0496037267	0.0496037257	0.0497684479

Table 6: Computed Error for Test 2

n	Real Solution	|Real - PINN|	|Real - ANN|
0.0	1.0000000000	0.00000e+00	3.3768e-03
0.1	0.6772640944	7.40110e-08	4.2602e-03
0.2	0.4686929882	1.70175e-08	2.2488e-03
0.3	0.3316906691	3.01574e-08	3.4073e-04
0.4	0.2399998605	2.14693e-08	2.7394e-04
0.5	0.1773498058	6.97450e-09	7.8490e-04
0.6	0.1335894465	5.73941e-09	8.7516e-04
0.7	0.1023305878	7.71586e-09	4.6268e-04
0.8	0.0795092732	1.02464e-08	8.9049e-05
0.9	0.0625056326	2.02936e-09	3.8577e-04
1.0	0.0496037267	9.69922e-10	1.6472e-04

Table 7: ANN-FODE (Audu *et al*, 2024) Model Performance Evaluation for Test 2

Number Of Nodes	MAE	RSME	Time(s)
40	2.87856e-03	3.65093e-03	11.56
80	1.53313e-03	1.88205e-03	4.45
160	3.14159e-03	3.54243e-03	3.85
320	2.24916e-03	2.68689e-03	4.28
500	2.10997e-03	2.56384e-03	3.71

Table 8: PINN-FODE Model Performance Evaluation for Test 2

Number Of Nodes	MAE	RSME	Time(s)
40	2.00024e-08	2.34893e-08	2.57
80	2.52332e-08	3. 52516e-08	2.14
160	1.53896e-08	1.76348e-08	2.69
320	2.50953e-08	3.14821e-08	2.77
500	1.83280e-08	2.22532e-08	3.64

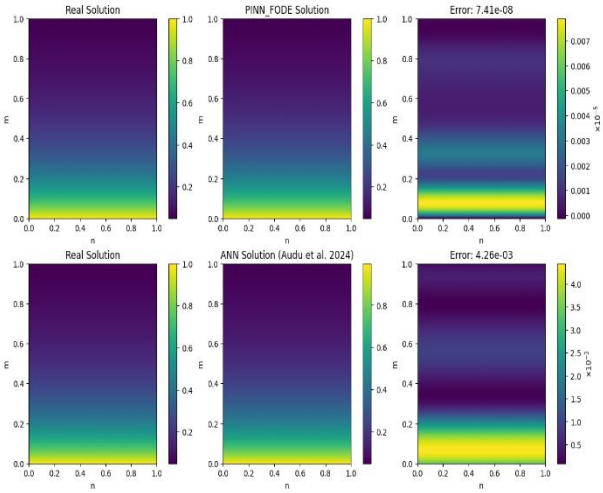

Figure 7: Heat-Map İllustration for Test 2

Test 3: Use the developed PINN-FODE model to obtain solution to the first-order initial value problem

$$\frac{d\underline{m}}{dn} = \underline{m}n^3 - 1.5\underline{m}$$

Satisfying the condition (initial): $n_0 = 0, \underline{m}_0 = 1$,

Real Solution: $e^{\frac{1}{4}n\left(n^3-6\right)}$

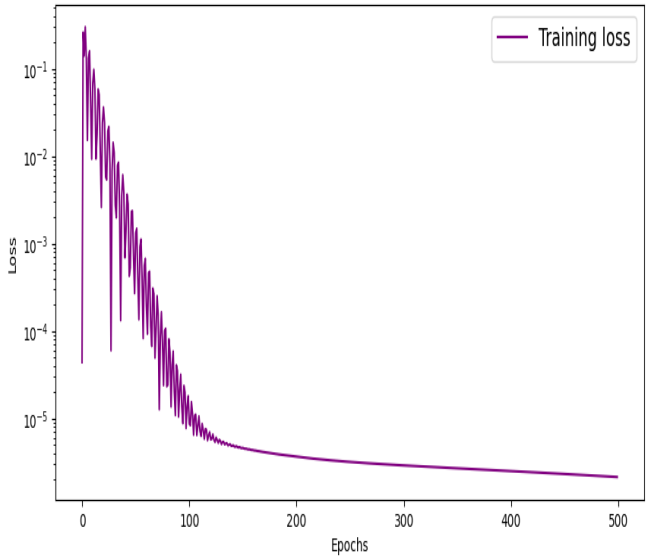

Figure 8: Training Loss Visual Representation for Test 3

Table 9: Computed Solution for Test 3

n	Real Solution	PINN Solution	ANN Solution
0.0	1.0000000000	1.0000000000	0.9985585213
0.1	0.8607295156	0.8607294925	0.8599051237
0.2	0.7411146164	0.7411146039	0.7379962206
0.3	0.6389206648	0.6389206456	0.6338128448
0.4	0.5523353219	0.5523352895	0.5468689799
0.5	0.4798052311	0.4798052436	0.4756994247
0.6	0.4199582338	0.4199582277	0.4183301330
0.7	0.3715859950	0.3715859857	0.3726402521
0.8	0.3336709440	0.3336709298	0.3365982175
0.9	0.3054482341	0.3054482279	0.3083880544
1.0	0.2865048051	0.2865047969	0.2864583731

Table 10: Computed Error for Test 3

n	Real Solution	|Real - PINN|	|Real - ANN|
0.0	1.0000000000	0.00000e+0	1.441479e-03
0.1	0.8607295156	2.30817e-08	8.243919e-04
0.2	0.7411146164	1.24461e-08	3.118396e-03
0.3	0.6389206648	1.91578e-08	5.107820e-03
0.4	0.5523353219	3.23871e-08	5.466342e-03
0.5	0.4798052311	1.24653e-08	4.105806e-03
0.6	0.4199582338	6.17733e-09	1.628101e-03
0.7	0.3715859950	9.27833e-09	1.054257e-03
0.8	0.3336709440	1.41703e-08	2.927273e-03
0.9	0.3054482341	6.21503e-09	2.939820e-03
1.0	0.2865048051	8.22785e-09	4.643200e-05

Table 11: ANN-FODE (Audu *et al*, 2024) model performance evaluation for Test 3

Number Of Nodes	MAE	RSME	Time(s)
40	2.09673e-03	2.46768e-03	1.61
80	1.69236e-03	2.01606e-03	1.15
160	1.75061e-03	2.08635e-03	1.28
320	1.71439e-03	2.09706e-03	1.09
500	1.86815e-03	2.27046e-03	1.01

Table 12: PINN-FODE Model Performance Evaluation for Test 3

Number Of Nodes	MAE	RSME	Time(s)
40	2.51063e-08	2.74804e-08	6.47
80	2.31753e-08	2.54411e-08	7.17
160	1.79785e-08	2.01207e-08	8.87
320	8.78145e-09	9.80727e-09	13.43
500	7.80371e-09	9.47628e-09	31.435

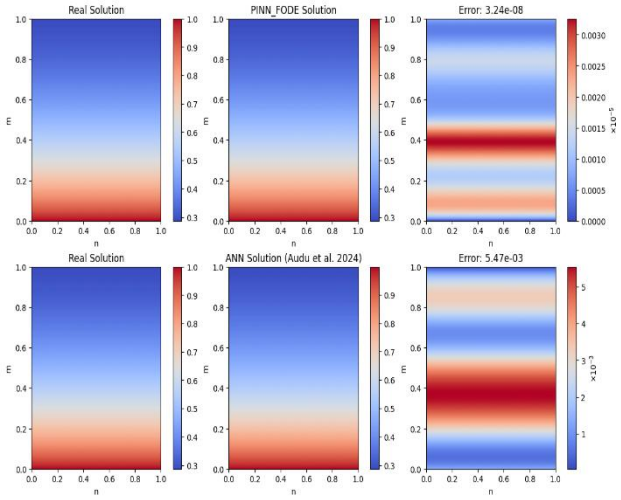

Figure 9: Heat-Map İllustration for Test 3

Test 4: Applying the developed PINN model to solve the first-order ODE

$$\frac{d\underline{m}}{dn} = -2n\underline{m}^2$$

Given the conditions (initial): $n_0 = 0, \underline{m}_0 = 1$

Real Solution: $\frac{1}{1+n^2}$

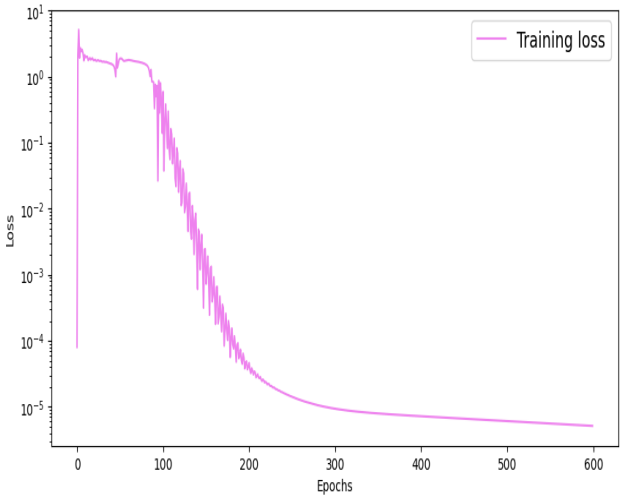

Figure 10 : Training Loss Visual Representation for Test 4

Table 13: Computed Solution for Test 4

n	Real Solution	PINN Solution	ANN Solution
0.0	1.000000000	1.0000000000	0.9995874763
0.1	0.990099012	0.9900990096	0.9895322323
0.2	0.961538493	0.9615384604	0.9612917304
0.3	0.917431175	0.9174311866	0.9176725149
0.4	0.862069010	0.8620689620	0.8626384735
0.5	0.800000011	0.8000000000	0.8005299568
0.6	0.735294103	0.7352941022	0.7354393005
0.7	0.671140909	0.6711409471	0.6707770824
0.8	0.609756052	0.6097560905	0.6090454459
0.9	0.552486181	0.5524862009	0.5518095493
1.0	0.500000000	0.5000000000	0.4998184443

Table 14: Computed Error for Test 4

n	Real Solution	|Real - PINN|	|Real - ANN|
0.0	1.0000000000	0.00000	4.12524e-04
0.1	0.9900990129	3.24288e-09	5.66781e-04
0.2	0.9615384936	3.31970e-08	2.46763e-04
0.3	0.9174311757	1.09316e-08	2.41339e-04
0.4	0.8620690107	4.87610e-08	5.69463e-04
0.5	0.8000000119	1.19209e-08	5.29945e-04
0.6	0.7352941036	1.44371e-09	1.45197e-04
0.7	0.6711409092	3.79197e-08	3.63827e-04
0.8	0.6097560525	3.79754e-08	7.10607e-04
0.9	0.5524861813	1.96857e-08	6.76632e-04
1.0	0.5000000000	0.00000e+0	1.81556e-04

Table 15: ANN-FODE (Audu *et al*, 2024) Model Performance Evaluation for Test 4

Number Of Nodes	MAE	RSME	Time(s)
40	1.78224e-03	2.20796e-03	7.89
80	1.78306e-03	2.21889e-03	1.29
160	1.62840e-03	2.10299e-03	1.04
320	1.70748e-03	2.21773e-03	1.51
500	1.37523e-03	1.72094e-03	1.18

Table 16: PINN-FODE model performance evaluation for Test 4

Number Of Nodes	MAE	RSME	Time(s)
40	5.98407e-08	7.19209e-08	3.89
80	2.66390e-08	3.55360e-08	4.48
160	1.81068e-08	2.18038e-08	4.25
320	1.59817e-08	1.99443e-08	6.98
500	2.68448e-08	3.25243e-08	22.21

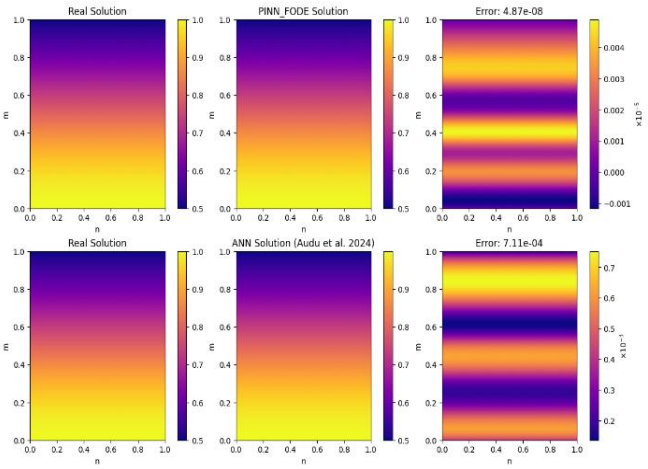

Figure 11: Heat-Map İllustration for Test 4

Test 5: We optimize the application of the developed PINN model to the first-order ODE

$$\frac{d\underline{m}}{dn} = \frac{3}{2} - \frac{1}{2}\underline{m}$$

Obeying the conditions (initial): $\underline{m}(n_0) = 4$

Real Solution: $3 + e^{-\frac{n}{2}}$

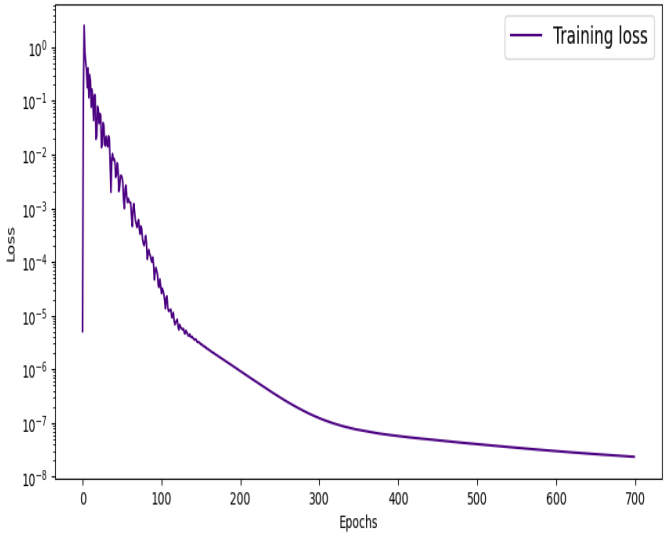

Figure 12: Training Loss Visual Representation for Test 5

Table 17: Computed Solution for Test 5

n	Real Solution	PINN Solution	ANN Solution
0.0	4.0000000000	4.0000000000	4.0002832413
0.1	3.9512293339	3.9512294238	3.9523916245
0.2	3.9048373699	3.9048374167	3.9059658051
0.3	3.8607079983	3.8607079713	3.8612987995
0.4	3.8187308311	3.8187307506	3.8186030388
0.5	3.7788007259	3.7788007831	3.7780096531
0.6	3.7408182621	3.7408182119	3.7395720482
0.7	3.7046880722	3.7046880939	3.7032728195
0.8	3.6703200340	3.6703200420	3.6690378189
0.9	3.6376280785	3.6376281592	3.6367480755
1.0	3.6065306664	3.6065306597	3.6062524319

Table 18: Computed Error for Test 5

n	Real Solution	|Real - PINN|	|Real - ANN|
0.0	4.0000000000	0.00000e+00	2.83241e-04
0.1	3.9512293339	8.99144e-08	1.16229e-03
0.2	3.9048373699	4.67688e-08	1.12844e-03
0.3	3.8607079983	2.69809e-08	5.90801e-04
0.4	3.8187308311	8.05083e-08	1.27792e-04
0.5	3.7788007259	5.71345e-08	7.91073e-04
0.6	3.7408182621	5.02497e-08	1.24621e-03
0.7	3.7046880722	2.17144e-08	1.41525e-03
0.8	3.6703200340	8.01312e-09	1.28222e-03
0.9	3.6376280785	8.07622e-08	8.80003e-04
1.0	3.6065306664	6.63869e-09	2.78234e-04

Table 19: ANN-FODE (Audu *et al*, 2024) Model Performance Evaluation for Test 5

Number Of Nodes	MAE	RSME	Time(s)
40	1.68965e-03	1.91179e-03	3.09
80	1.70129e-03	1.93147e-03	4.67
160	9.82436e-04	1.12452e-03	2.79
320	1.00528e-03	1.15369e-03	3.49
500	8.36221e-04	9.61375e-04	2.55

Table 20: PINN-FODE Model Performance Evaluation for Test 5

Number Of Nodes	MAE	RSME	Time(s)
40	3.07028e-08	4.87733e-08	2.20
80	4.41247e-08	6.08441e-08	2.72
160	4.85418e-08	6.67564e-08	2.36
320	4.57698e-08	6.11071e-08	2.14
500	4.26862e-08	4.83555e-08	2.31

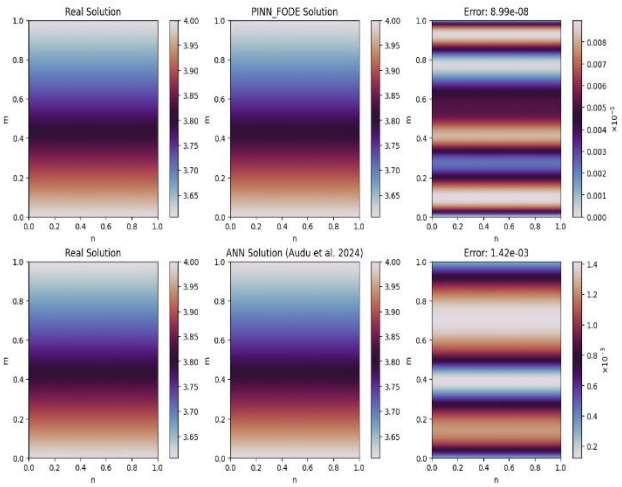

Figure 13: Heat-Map İllustration for Test 5

Test 6: A metal rod has initial temperature at 1000. 00 Kelvin, at time 500 seconds, what will its temperature be? The DE (differential equation) of the temperature of the metal rod is governed by $\frac{d\underline{m}}{dn} = -2.2067\times10^{-12}\left(\underline{m}^4 - 81\times10^8\right)$, that satisfies the initial conditions $n_0 = 0,\ \underline{m}_0 = 1000,\ h = 50\sec s$. We implement the developed PINN model to determine the temperature of the metal rod.

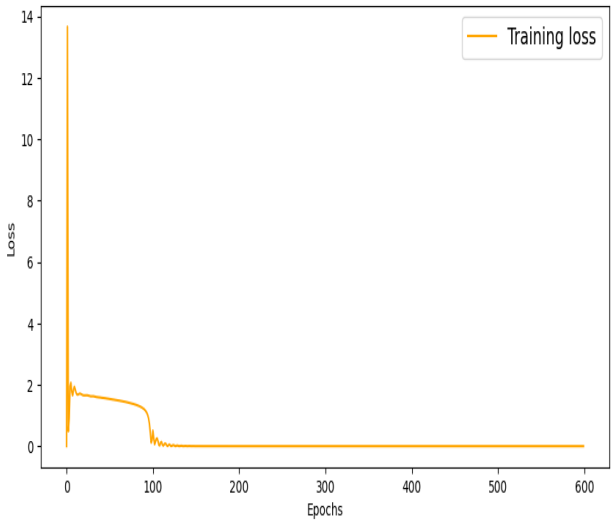

Figure 14: Training Loss Visual Representation for Test 6

Table 21: Computed Solution for Test 6

n	Real Solution	PINN Solution	ANN Solution
0	1000.000000	1000.000122	1000.000244
50	909.836070	909.880493	909.729736
100	845.551286	845.520020	846.158691
150	796.467377	796.120361	796.613464
200	757.261554	757.227844	756.757019
250	724.928388	724.914185	725.446655
300	697.618880	697.472351	697.427002
350	674.121546	674.300781	674.778076
400	653.603327	653.802734	653.534729
450	635.469029	635.298950	634.877075
500	619.279966	619.271484	619.168701

Table 22: Computed Error for Test 6

n	Real Solution	|Real - PINN|	|Real - ANN|
0	1000.000000	0.000122	0.000244
50	909.836070	0.044423	0.106334
100	845.551286	0.031266	0.607405
150	796.467377	0.347016	0.146087
200	757.261554	0.033710	0.504535
250	724.928388	0.014204	0.518267
300	697.618880	0.146529	0.191886
350	674.121546	0.179236	0.656530
400	653.603327	0.199408	0.068598
450	635.469029	0.170079	0.591954
500	619.279966	0.008482	0.111265

Table 23: ANN-FODE (Audu *et al*, 2024) Model Performance Evaluation for Test 6

Number Of Nodes	MAE	RSME	Time(s)
500	1.2350	1.6468	51.69
1500	1.4534	1.9360	64.54
2500	1.1668	1.6233	65.66
3500	1.1375	1.3130	78.43
5000	1.2348	1.0124	89.21

Table 24: PINN-FODE Model Performance Evaluation for Test 6

Number Of Nodes	MAE	RSME	Time(s)
500	0.0018	0.0023	3.56
1500	0.0027	0.0068	1.88
2500	0.0083	0.0012	4.50
3500	0.0069	0.0057	2.52
5000	0.0017	0.0006	3.59

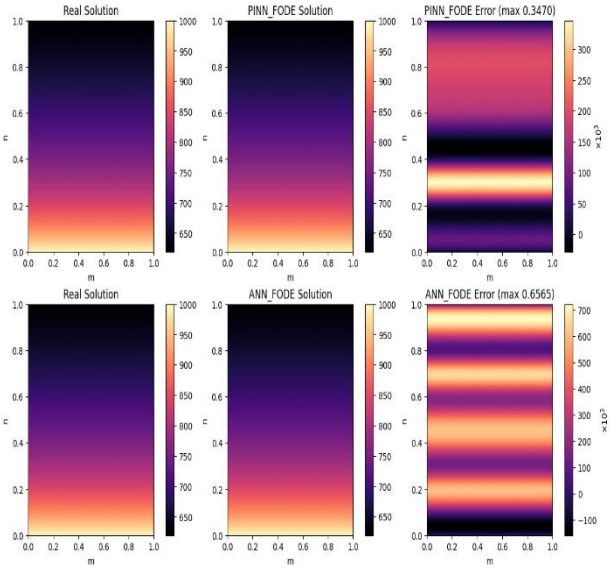

Figure 15: Heat-Map İllustration for Test 6

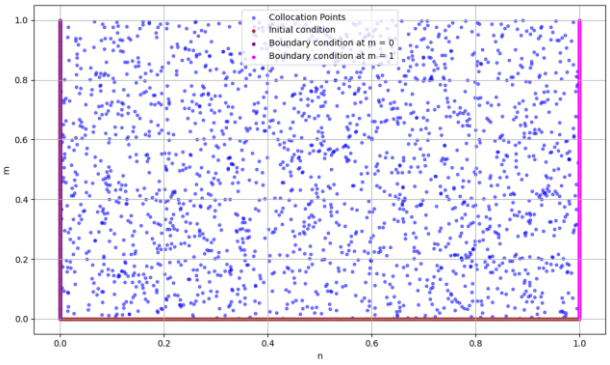

Figure 16: Illutration of Collocation point

The training configuration utilized for both ANN and PINN models is presented in Table 25.

Table 25: Training Configuration Parameters

Test	Collocation Point	Hidden Layers	Number of Neurons	Number of Epochs	Activation Function	Learning Rate
1	$n \in [0,1,100]$	4	50	5000	Tanh	10^{-4}
2	$n \in [0,1,100]$	4	40	2000	Tanh	10^{-4}
3	$n \in [0,1,100]$	4	50	2000	Tanh	10^{-4}
4	$n \in [0,1,100]$	4	80	5000	Tanh	10^{-4}
5	$n \in [0,1,100]$	4	60	2000	Tanh	10^{-4}
6	$n \in [0,1,100]$	4	50	2000	Tanh	10^{-4}

4. DISCUSSION

From the results acquired in this study, it can be noticed that the developed PINN model yields solutions that outperformed those obtained from the established ANN model (Audu *et al*, 2024) and a few crucial considerations should be mentioned. Firstly, the results of the ANN to be compared were obtained from the established ANN model (Audu *et al*, 2024) study and trained under the same training conditions like network structure, optimizing method, stopping criteria and computations as the PINN model. Secondly, their incredibly small values of errors 10^{-8} to 10^{-10} must be deduced as physics-informed on the PINN, where the governing differential equation is an effective regularization method that reduces the space of solutions and control overfitting. The current research, however, is concerned with the accuracy of solutions in the training field and the generalization ability of the model in another field was not directly explored. Moreover, this work is limited to first-order ODEs, it is widely understood that the extension of PINNs to higher-order ODEs and partial differential equations are associated with other challenges, such as numerical instability due to higher-order automatic differentiation, higher computational costs and convergence problems. Lastly, although the computation times reported show that PINNs can be more expensive per-epoch because they replace the application of different operators with the objective, they also show competitive scalability where high accuracy and physical consistency are needed.

It was observed from the tables that the developed PINN model gives more accurate and better results than the established ANN model by authors in [27]. Throughout all the 6 tests, the developed PINN model yields consistently low absolute errors between the range of 10^{-8} to 10^{-10} and that of the established ANN model ranging from 10^{-1} to 10^{-5}. Furthermore, this is clarified by the comparison between the errors which was represented by the heat-map solution where the developed PINN model maintains stability and nearly absolute error-free outcomes across all tests while the established ANN model displays irregular error patterns which remains significantly higher.

The results obtained from these models highlight the effectiveness and accuracy of the developed PINN model over the established ANN model in solving the presented problems. These findings provide a motivation to future research to integrate both models to be trained under exactly the same conditions so as to examine their generalization and robustness, and to enhance the scalability to higher-order and large-scale tasks.

5. CONCLUSION

In this study, we constructed and trained a Physics-Informed Neural Network model that aims at the solution of first-order ODEs by applying physical constraints to its training procedure via the loss function so that it can achieve better results. Six numerical tests were conducted to check its efficacy, and the results indicate that both the developed PINN model and the established ANN model [27] are effective, but the developed PINN model is always more effective at the expense of the established ANN model.

As the tables and heat-maps visualization indicate, the developed PINN model can produce credible solutions with only a few errors that present the accuracy and efficiency of the developed model to work with on various problems with first-order ODEs. This study shows the value of the chosen model in solving the problems of the first order of the ODEs. In future research, the developed PINN model can be extended to seek solutions to higher-order ODEs, which would provide a more essential contribution of artificial intelligence to neural network and further refine the knowledge of its contribution.

Overall, this study has made a substantial contribution to the PINN model, which helps to make an informed choice of the model in order to find the solutions to first-order ODEs.

REFERENCES

1. Linot, A. J., Burby, J. W., Tang, Q., Balaprakash, P., Graham, M. D., & Maulik, R. (2022). Stabilized neural ordinary differential equations for long-time forecasting of dynamical systems. *Journal of Computational Physics*, *474*, 111838, https://doi.org/10.1016/j.jcp.2022.111838
2. Omejc, N., Gec, B., Brence, J., Todorovski, L., & Džeroski, S. (2024). Probabilistic grammars for modeling dynamical systems from coarse, noisy, and partial data. *Machine Learning*, *113*(10), 7689–7721, https://doi.org/10.1007/s10994-024-06522-1
3. Jimoh, A. K., & Adewumi, A. O. (2022). A two-step block method with two hybrid points for the numerical solution of first order ordinary differential equations. *Open Journal of Mathematical Sciences*, *6*(1), 281–288, https://doi.org/10.30538/oms2022.0193
4. Shende, S. K. (2024). Application of first-order differential equations in R L Circuits. *International Journal for Research in Applied Science and Engineering Technology*, *12*(2), 1338–1343, https://doi.org/10.22214/ijraset.2024.58508
5. Rufai, M. A., Carpentieri, B., & Ramos, H. (2023). A new hybrid block method for solving First-Order differential system models in applied sciences and engineering. *Fractal and Fractional*, *7*(10), 703, https://doi.org/10.3390/fractalfract7100703
6. Shior, M., Agbata, B., Ezeafulukwe, A., Topman, N., Mathias, A., & Ekwoba, L. (2024). Applications of first-order ordinary differential equations to real life systems. *European Journal of Statistics and Probability*, *12*(2), 43–50, https://doi.org/10.37745/ejsp.2013/vol12n24350
7. Kek, S. L., Chen, C. Y., & Chan, S. Q. (2024). First-Order linear ordinary differential equation for regression modelling. In *Frontiers in artificial intelligence and applications*,*381,* 129–134, https://doi.org/10.3233/faia231184
8. Audu, K. J., Abubakar, T. A., Amuda, Y. Y., & Nkereuwem, J. E. (2024). Comparative numerical evaluation of some Runge-Kutta methods for solving first order systems of ODEs. *Journal of Engineering and Basic Sciences*, *03*, 20–28, https://doi.org/10.54709/joebs.1556269
9. Koroche, K. A. (2021). Numerical solution of first order ordinary differential equation by using Runge-Kutta method. *International Journal of Systems Science and Applied Mathematics*, *6*(1), 1, https://doi.org/10.11648/j.ijssam.20210601.11
10. Adams, N. O., Badmus, A. M., & Yahuza, M. (2025). Grid-Based and hybrid Five-Step Block Runge-Kutta methods for solving linear and non-linear First-Order ordinary differential equations. *Faculty of Natural and Applied Sciences Journal of Mathematical Modeling and Numerical Simulation*, *2*(3), 30–43, https://doi.org/10.63561/jmns.v2i3.863
11. Babaei, M. (2024). An efficient ODE-solving method based on heuristic and statistical computations: αII-(2 + 3)P method. *The Journal of Supercomputing*, *80*(14), 20302–20345, https://doi.org/10.1007/s11227-024-06137-2
12. Qin, O., & Xu, K. (2024). Solving nonlinear ODEs with the ultraspherical spectral method. *IMA Journal of Numerical Analysis*, *44*(6), 3749–3779, https://doi.org/10.1093/imanum/drad099
13. Shams, M., Kausar, N., Araci, S., & Kong, L. (2024). On the stability analysis of numerical schemes for solving non-linear polynomials arises in engineering problems. *AIMS Mathematics*, *9*(4), 8885–8903, https://doi.org/10.3934/math.2024433
14. Tai, H. S., & Basri, S. (2025). Numerical Integration approach for nonlinear differential equation in growth modelling. *International Journal on Robotics Automation and Sciences*, *7*(2), 29–35, https://doi.org/10.33093/ijoras.2025.7.2.4
15. Audu, K. J., & Babatunde, O. (2024). A comparative analysis of two semi analytic approaches in solving systems of First-Order differential equations. *Scientific Journal of Mehmet Akif Ersoy University.*, *7*(1), 8–24, https://doi.org/10.70030/sjmakeu.1433801
16. Abdal, S., Ashique, A., Afzal, U., Kumar, M. D., Masood, K., & Shah, N. A. (2025). A comprehensive review on artificial neural network techniques (Levenberg–Marquardt, Bayesian regularization, scaled conjugate gradient) for magnetohydrodynamic hybrid nanofluid flow with bio-convection and heat sources. *AIMS Mathematics*, *10*(10), 23084–23135, https://doi.org/10.3934/math.20251025
17. Han, D., & Temuer, C. (2024). Piecewise neural network method for solving large interval solutions to initial value problems of ordinary differential equations. *Symmetry*, *16*(11), 1490, https://doi.org/10.3390/sym16111490
18. Nam, H., Baek, K. R., & Bu, S. (2022). Error estimation using neural network technique for solving ordinary differential equations. *Advances in Continuous and Discrete Models*, *2022*(1), 45, https://doi.org/10.1186/s13662-022-03718-4
19. Yi, Z. (2023). nmODE: neural memory ordinary differential equation. *Artificial Intelligence Review*, *56*(12), 14403–14438, https://doi.org/10.1007/s10462-023-10496-2

20. **Carbone, M. R. (2022). When not to use machine learning: A perspective on potential and limitations. *MRS Bulletin*, *47*(9), 968–974, https://doi.org/10.1557/s43577-022-00417-z**
21. **Wen, L., Xi, J., Hu, H., Xiong, L., Lu, G., & Xiao, T. (2025). Neural ODE-Based dynamic modeling and predictive control for power regulation in distribution networks. *Energies*, *18*(13), 3419, https://doi.org/10.3390/en18133419**
22. **Demirkaya, A., Lockwood, K., Stratis, G., Imbiriba, T., Ilieş, I., Rampersad, S., Alhajjar, E., Guidoboni, G., Danziger, Z. C., & Erdoğmuş, D. (2024). A hybrid ODE-NN framework for modeling incomplete physiological systems. *IEEE Transactions on Biomedical Engineering*, *72*(4), 1377–1386, https://doi.org/10.1109/tbme.2024.3505796**
23. **Li, S., & Wang, X. (2020). Solving ordinary differential equations using an optimization technique based on training improved artificial neural networks. *Soft Computing*, *25*(5), 3713–3723, https://doi.org/10.1007/s00500-020-05401-w**
24. **Okereke, R. N., Maliki, O. S., & Oruh, B. I. (2021). A novel method for solving Ordinary Differential Equations with Artificial Neural Networks. *Applied Mathematics*, *12*(10), 900–918, https://doi.org/10.4236/am.2021.1210059**
25. **Qamar, R., & Zardari, B. A. (2023). Artificial Neural Networks: An Overview. *Mesopotamian Journal of Computer Science*, 130–139, https://doi.org/10.58496/mjcsc/2023/015**
26. **Marchenko, N., Sydorenko, G., & Rudenko, R. (2021). Using of Multilayer Neural Networks for the solving systems of differential equations. *Bulletin of National Technical University KhPI Series System Analysis Control and Information Technologies*, *2 (6)*, 81–88, https://doi.org/10.20998/2079-0023.2021.02.13**
27. **Audu, K. J., Benjamin, M., Mohammed, U., & Yahaya, Y. A. (2024). Utilizing the artificial neural network approach for the resolution of First-Order ordinary differential equations. *Malaysian Journal of Science and Advanced Technology*, *4*(3), 210–216. https://doi.org/10.56532/mjsat.v4i3.265**
28. **Li, Z. (2025). A review of Physics-Informed Neural Networks. *Applied and Computational Engineering*, *133*(1), 165–173, https://doi.org/10.54254/2755-2721/2025.20636**
29. **Venkatachalapathy, P., & Mallikarjunaiah, S. M. (2022). A feedforward neural network framework for approximating the solutions to nonlinear ordinary differential equations. *Neural Computing and Applications*, *35*(2), 1661–1673, https://doi.org/10.1007/s00521-022-07855-5**
30. **Almqvist, A. (2021). Fundamentals of Physics-Informed neural networks applied to solve the Reynolds boundary value problem. *Lubricants*, *9*(8), 82, https://doi.org/10.3390/lubricants9080082**
31. **Dong, C. (2025). Solving Differential Equations with Physics-Informed Neural Networks. *Theoretical and Natural Science*, *87*(1), 137–146, https://doi.org/10.54254/2753-8818/2025.20346**
32. **Wasif, K., & Shahbaz, A. (2024). Comparative analysis of Finite Difference Method (FDM) and Physics-Informed Neural Networks (PINNs). *I-manager's Journal on Mathematics*, *13*(1), 27–37, https://doi.org/10.26634/jmat.13.1.20383**
33. **Park, S., & Sin, Y. (2021). Artificial Neural Network (ANN) Modeling Analysis of Algal Blooms in an Estuary with Episodic and Anthropogenic Freshwater Inputs. *Applied Sciences*, *11*(15), 6921, https://doi.org/10.3390/app11156921**
34. **Franjić, S., & Galić, D. (2021). In Shortly about Neural Networks. *Advances in Computer and Communication*, *2*(1), 15–19, https://doi.org/10.26855/acc.2021.08.001**
35. **Kariri, E., Louati, H., Louati, A., & Masmoudi, F. (2023). Exploring the advancements and future research directions of artificial neural networks: a text mining approach. *Applied Sciences*, *13*(5), 3186, https://doi.org/10.3390/app13053186**