MJSAT

Malaysian Journal of Science and Advanced Technology

journal homepage: https://mjsat.com.my/

Utilizing the Artificial Neural Network Approach for the Resolution of First-Order Ordinary Differential Equations

Khadeejah James Audu*[1], Marshal Benjamin[1], Umaru Mohammed[1] and Yusuph Amuda Yahaya[2]

[1]*Department of Mathematics, Federal University of Technology Minna, Nigeria.*
[2]*Department of Mathematics, Pen Resource University, Gombe, Nigeria.*

KEYWORDS

First-Oder ODE
Artificial Neural Network
Computational Efficiency
Numerical echnique
Convergence Analysis

ARTICLE HISTORY

Received 9 February 2024
Received in revised form 28 March 2024
Accepted 20 May 2024
Available online 16 June 2024

ABSTRACT

Ordinary Differential Equations (ODEs) play a crucial role in various scientific and professional domains for modeling dynamic systems and their behaviors. While traditional numerical methods are widely used for approximating ODE solutions, they often face challenges with complex or nonlinear systems, leading to high computational costs. This study aims to address these challenges by proposing an artificial neural network (ANN)-based approach for solving first-order ODEs. Through the introduction of the ANN technique and exploration of its practical applications, we conduct numerical experiments on diverse first-order ODEs to evaluate the convergence rate and computational efficiency of the ANN. Our results from comprehensive numerical tests demonstrate the efficacy of the ANN-generated responses, confirming its reliability and potential for various applications in solving first-order ODEs with improved efficiency and accuracy.

1. INTRODUCTION

Modeling various phenomena in science, engineering, and other fields often involves first-order Ordinary Differential Equations (ODEs) [1]. While traditional numerical methods like Euler's method and the Runge-Kutta method have been useful, they can be computationally demanding for complex or nonlinear ODEs. This study aims to address this challenge by employing Artificial Neural Networks (ANNs) to solve first-order ODEs representing physical processes. Developed by Frank Rosenblatt in 1958, ANNs have proven effective for solving both linear and nonlinear differential equations [2]. Notable applications include a Deep Learning Library for Solving Differential Equations [3], physics-informed neural networks for high-speed flows [4], and feed-forward neural networks with trainable delay [5]. Several studies have explored using ANNs to approximate ODE solutions, such as training neural differential equations on time series [6], optimizing enhanced artificial neural networks for ODEs [7], and applying multilayer neural networks to solve systems of differential equations [8]. These approaches demonstrate the practicality and efficiency of using ANNs for solving ODEs in various domains [9].

In the field of computational mathematics, the creation and use of the ANN for first-order ODE offers a fresh and extremely motivated method. The expanding significance of faithfully modeling real-world processes in domains like science and engineering, where ODE is essential, is the driving force behind this issue. The need for better ODE-solving techniques that can yield more precise and reliable solutions, therefore expanding our knowledge and problem-solving skills across a range of scientific and engineering applications, is the driving force behind this research.

This study addresses the fundamental task of resolving first-order ODEs, crucial for modeling dynamic systems in various scientific and engineering domains. While traditional numerical methods have long been employed for this purpose, they may encounter challenges with stiff equations or high-dimensional systems, motivating the exploration of alternative computational techniques. Inspired by the success of artificial neural networks (ANNs) in diverse fields, this research investigates the application of ANNs to solve first-order ODEs. The novelty lies in adapting ANNs, known for their capacity to learn complex patterns, to tackle numerical solution tasks

*Corresponding author:
E-mail address: Khadeejah James Audu < k.james@futminna.edu.ng >.
https://doi.org/10.56532/mjsat.v4i3.265

typically associated with differential equations. The study's primary contribution is twofold: firstly, it extends the application domain of ANNs into mathematical modeling, potentially offering a promising alternative to traditional numerical methods. Secondly, it provides insights into the effectiveness and limitations of ANNs for resolving ODEs, advancing the understanding and potential utilization of ANN-based approaches in computational mathematics. Through rigorous experimentation and analysis, this research aims to establish the feasibility and practical implications of employing neural networks for ODE resolution, contributing to advancements in computational science and engineering.

2. LITERATURE REVIEW

ODEs are mathematical expressions of the relationships between a function and its derivatives. They are significant in the fields of science and engineering because they are a crucial tool for simulating and evaluating dynamic systems. Numerous phenomena, including economic systems, physical processes, biological dynamics, and many more, are regulated by underlying principles that ODEs capture. ODEs are used in science to describe the behavior and evolution of physical systems, such as the movement of celestial planets, chemical reaction dynamics, and optimal disease propagation. [10] as well as [11]. Scientists can understand complex relationships, predict results, and examine the effects of various factors by using ODEs to express the governing equations.

Control systems, robotics, fluid dynamics, and electrical circuits all use ODEs. Engineers utilize ODEs to predict and optimize system performance, develop efficient algorithms, and make well-informed decisions on the behavior and stability of systems. Control systems can ensure stability and intended results by using ODEs, which provide a mathematical framework for process analysis and design. [12] presented a signal-based method for detecting electrical abnormalities in three-phase AC induction motors (IM) using an automated computer-control system. A dedicated physical prototype was intentionally designed, developed, and constructed for experimental validation.

The significance of ODEs lies in their ability to represent complex systems' dynamic behavior, temporal evolution, and interactions among several variables. In order to gain a better understanding of system dynamics, they give scientists and engineers the capacity to predict, simulate, and mathematically describe genuine phenomena.

The fundamental concepts of the natural world are better understood by scientists and engineers, who can also come up with creative solutions for problems that arise in the real world.

As a bridge between theoretical models and empirical data, ODEs help validate and enhance mathematical models. Researchers can improve ODE-based models' accuracy and dependability, enabling comparing model predictions with real data can lead to more accurate forecasts and better decision-making. One of the key reasons ODEs are important in the scientific and engineering sectors is that they provide a mathematical foundation for understanding and describing dynamic systems. They facilitate the analysis of complex systems, enable modeling and prediction, and offer a framework for developing practical solutions. Because ODEs may be used to recognize and describe the fundamental concepts that underlie the behavior of different systems, they are critical to the advancement of research, technology, and innovation.

Addressing the computational challenge of solving large-scale linear equations is paramount in numerical computation. Authors in [13] introduced an innovative method utilizing deep neural networks for solving such equations. In their study [14] introduced and constructed an artificial neural network model designed to predict a student's likelihood of passing a specific class, while ensuring the confidentiality of personal or sensitive information that could jeopardize student privacy.

Various studies have explored the utilization of neural networks for solving differential problems, employing diverse optimization techniques within a procedural framework.

3. DESCRIPTION OF THE ARTIFICIAL NEURAL NETWORK

An alternative to conventional methods that need discretization and approximation is the artificial neural network (ANN), which shows promise as a method for solving ODEs. Each differential equation is presented as a "optimization problem" in this paradigm, denoting a change in viewpoint. The ANN technique makes use of the network's ability to solve optimization problems, in contrast to approaches that depend on decomposing and approximating differential equations.

This cutting-edge method revolutionizes problem-solving by utilizing neural networks. The ANN can automate the procedure entirely by methodically learning and adapting to ODEs as optimization problems. An approach to ODEs that is dynamic and adaptable is made possible by the network's capacity to solve issues on its own and learn from the data that is sent to it. This change from customary practices suggests a more comprehensive and flexible approach to problem solving.

Artificial neural networks have the potential to solve ODEs because of their innate ability to recognize patterns, adapt, and optimize efficiently. Neural network automation simplifies the process and has the potential to address complicated and non-linear systems that may be difficult to handle with conventional techniques. This method represents a paradigm change in the way differential equations are thought about and handled, creating opportunities for more adaptable and efficient problem-solving in technical and scientific fields.

Consider the following first-order ODE:

$$\frac{d}{dt} = q(t,s), \qquad t \in [a,b] \;\; s(0) = s_0 \tag{1}$$

The defined ANN takes in the input vector $T = (t_1, t_2, t_3, ..., t_n)$, and outputs a scalar $N(t, p)$ where *t* refers to the inputs and *p* denotes the adjustable weights and bias. The ANN is depicted in figure 1.

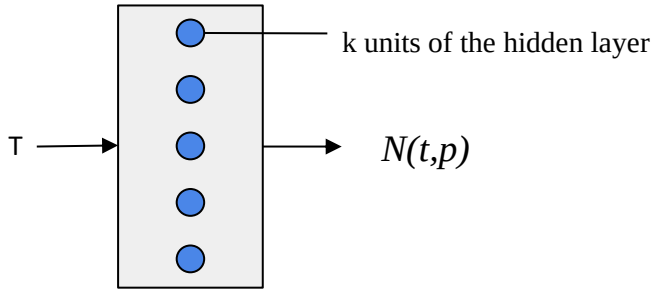

Figure 1: A network of connected block

The neural network defined consists of a hidden layer of *k* units as shown in figure 1. For each input vector, the weighted sum for the hidden layer is calculated.

$$z_i = b_i + \sum_{j=i}^{n} (p_{ij} t_i) \tag{2}$$

where b_i refers to the optional bias and p_{ij} refers to the weights acting on the vector $\underline{T}$. The sum z_i is acted upon by a sigmoid function given as:

$$\sigma(\boldsymbol{z_i}) = \frac{1}{1+e^{-z_i}} \tag{3}$$

Given the hidden weights denoted as the vector $V = (v_1, v_2, v_3, ..., v_n)$, the final scalar output is given as:

$$N(t,p) = \sum_{i=1}^{k} v_i \sigma(z_i) = v_i \sigma\left(b_i + \sum_{j=i}^{n} p_{ij} t_i \right) \tag{4}$$

where *k* refers to the number of units in the hidden node.

According to the universal approximation theorem, utilizing a neural network enables an approximation of the solution of N with arbitrary precision. Therefore, N can be regarded as a neural network of T.

$$\frac{\partial}{\partial t_k} z_i = \frac{\partial}{\partial t_k} (b_i + \textstyle\sum_{j=i}^{n} p_{ij} t_i) \tag{5}$$

$$\frac{\partial}{\partial t_k} \sigma(z_i) = \sigma'(z_i) \frac{\partial}{\partial t_k} z_i = p_{ik} \sigma'(z_i) \tag{6}$$

for first order ODE

$$\frac{dN}{d\underline{t}} = \sigma' v_1 (p_1 \underline{t}) p_1 \tag{7}$$

for second order ODE

$$\frac{d^2 N}{d\underline{t}} = \sigma'' v_1 (p_1 \underline{t}) p_1^2 \tag{8}$$

Hence all the derivatives of N with respect to the input T can be found. Through back propagation, we can always find the derivative of the output with respect to the input cases of multilayer which is called automatic differentiation.

Recall the ODE given in (1). Assuming $NN \cong s(t)$, then the optimization problem is formed as a loss function given as the mean square of the ODE and the boundary conditions:

$$L^2 = \sum_{l} \left| \frac{dNN(t_i)}{dt} - q(t_i, s) \right|^2 + |NN(0) - s_0|^2 \tag{9}$$

Thus, the original ODE is represented as an optimization issue which is shown in equation (9), and is solved using the Adam optimizer algorithm.

3.1 Utilizing Optimization and Backpropagation Techniques to Train Neural Networks

There are three distinct processes in the training process of a neural network, which is constant;

1. Feed the input data into the neural network, the data flows from layer to layer until output is retrieve z* = network (t, w).
2. Examine the neural network's output against the intended output to determine the error.

$$\underline{E} = \frac{(z^* - \underline{z})}{2} \tag{10}$$

3. Reduce inaccuracy by adjusting neural network parameters through gradient descent

$$p \leftarrow p - \infty \frac{\partial \underline{E}}{\partial p} \tag{11}$$

4. Restart from scratch.

3.2 Algorithm for the Artificial Neural Network

1. Library imports necessary for training a neural network to estimate the solution of a first-order ODE and visualize the outcome are as follows:
   - Import torch: PyTorch, a library offering tensor computations and automatic differentiation, is an ideal option for neural network implementation.
   - Import torch.nn as nn: This imports the PyTorch neural network module ('nn'), which encompasses classes and functions for defining and training neural network models.
   - Import torch.optim as optim: This imports the PyTorch optimizer module ('optim'), utilized for updating model parameters during training, such as Adam and SGD.
   - Import matplotlib. pyplot as plt: This imports the Matplotlib library, commonly used for creating visualizations such as plots and graphs. In this case, Matplotlib is utilized to visualize the training loss and compare predicted values with the exact solution of the ODE.
2. NN Definition: This section outlines a fundamental neural network structure, consisting of input, hidden, and output layers.
3. Model and Device Setup: The code determines the computational device, either GPU or CPU, and relocates the neural network accordingly.
4. Question formulation: This section defines the mathematical function along with its initial condition.
5. Loss function/Optimization: This section determines the learning rate of the network and handles optimization.

6. Data preparation: This involves setting up the input data and initial conditions. Then proceed to design the ANN architecture using the values predicted by the model as exact solution.
7. Training loop: The network is trained for a finite number of epochs, computes the loss, backpropagates the gradients, and updates the network parameters.
8. Prediction generation: The code generates predictions by applying the trained network to a range of input values and stores the results.
9. Visualization: This encompasses plotting the loss curve.

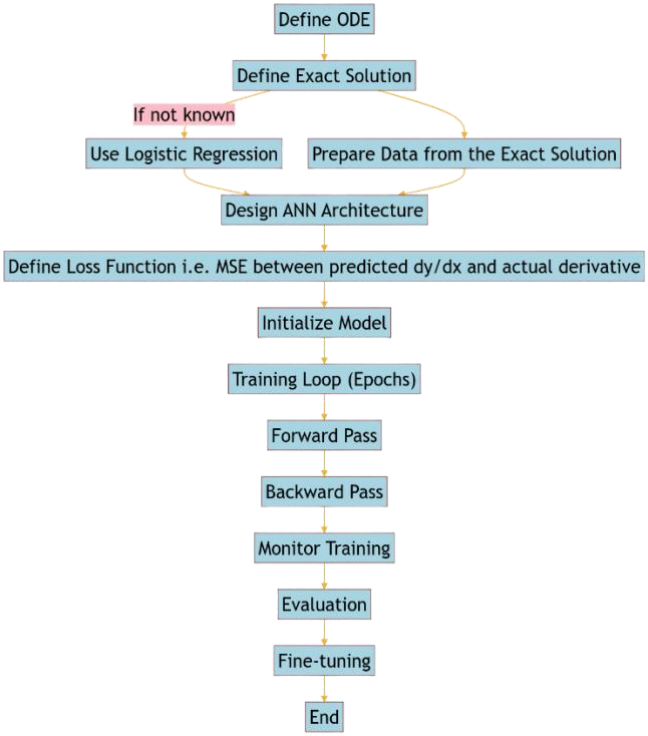

Figure 2: The process flow of the ANN approach.

4. IMPLEMENTATIONS AND RESULTS

This section focuses on utilizing the neural network approach to solve specific first-order ODEs, as illustrated in Figure 3. The numerical computations and experiments are conducted within a Python environment, specifically using Jupyter Notebook, and the results are analyzed and discussed accordingly.

Experiment 1: We employ the ANN approach to resolve:

$\frac{d\underline{z}}{d\underline{t}} = t^2 - 4\underline{z}$ by the conditions $\underline{t_0} = 0, \underline{z_0} = 1$

Solution of Exact: $\underline{z}(t) = \frac{1}{32} + \frac{31e^{-4\underline{t}}}{32} + \frac{\underline{t}^2}{4} - \frac{\underline{t}}{8}$

The generation of the training dataset can be approached through various methodologies. One of such method involves the utilization of the exact solution within a predetermined interval, such as [0,1]. This approach is particularly effective when the exact solution is known and can be accurately represented within the specified interval.

However, in instances where the exact solution is unknown, alternative strategies must be employed. A study conducted by [7] advocates for the use of logistic regression models over a defined interval. This method allows for the generation of a training dataset even in the absence of a known exact solution, thereby expanding the applicability of this approach to a wider range of scenarios.

Table 1. Computational Outcome for Experiment 1

Points {t} of **Training**	**Values** of Exact	**Values** of **ANN**	**ANN Error (Absolute)**
0.0	1.0000000000	0.9975615740	2.4384260×10^{-3}
0.1	0.6706225276	0.6668983698	3.7241578×10^{-3}
0.2	0.4515374005	0.4484636486	3.0737519×10^{-3}
0.3	0.3080318868	0.3070761859	9.557009×10^{-4}
0.4	0.2168372571	0.2169048488	$0.6.75917\times10^{-5}$
0.5	0.1623560488	0.1619767845	3.792643×10^{-4}
0.6	0.1341329962	0.1327966154	1.3364000×10^{-3}
0.7	0.1251597404	0.1234643161	1.6954000×10^{-3}
0.8	0.1307384074	0.1297434270	9.949800×10^{-4}
0.9	0.1477198452	0.1479196250	1.997800×10^{-4}
1.0	0.1739932895	0.1742707193	2.774300×10^{-4}

The neural output converges within the designated accuracy range, as shown in Table 1, which offers a thorough summary of error minimization at 11 training points that are equally spaced. At these precise times throughout the neural network's training rounds, the table probably illustrates the progressive decrease in mistakes. The neural network's predictions approach the desired or true values throughout training, reaching a level of accuracy that satisfies predetermined criteria, as indicated by the convergence of the neural output. In addition to providing insight into the neural network's capacity to gradually improve predictions and reach convergence within the given accuracy constraints, this table is a useful tool for evaluating the neural network's effectiveness in minimizing mistakes across different training phases.

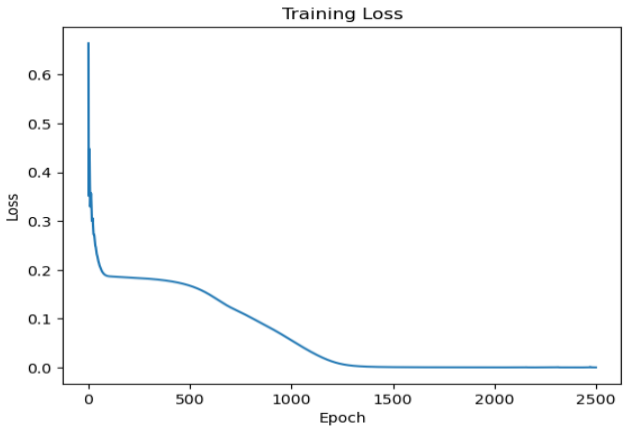

Fig. 4a. Loss Plot concerning Problem 1

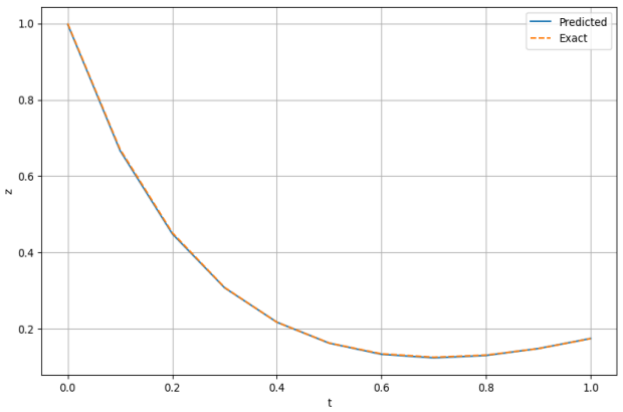

Fig. 4b. Exact and Predicted Values Plot concerning Problem 1

Astounding insights into the neural network's iteration behavior are provided by Figure 4a. During the initial phases, the network exhibited a quick integration of the pattern into the ODE, demonstrating its effective learning abilities. The graphic represented this transitional phase, with the network aiming to stabilize around the 0.2 point on the loss axis. The network reached stability at 0.00 on the loss axis after 1500 iterations, which is noteworthy because it shows a convergence towards the actual solution. Figure 4b provides additional validation of the convergence, indicating that the neural network accurately matched the real solution as stability at 1.0 is reached. The accuracy with which the Artificial Neural Network (ANN) method resolves the first-order ODE is demonstrated by this convergence to the correct solution potential as a strong computational instrument in these kinds of uses. The network's learning and convergence operations in solving the ODE are visually confirmed by the graphical representation in the figures.

Experiment 2: The following linear initial value problem is:

$$\frac{d\underline{z}}{d\underline{t}} = e^{-2\underline{t}} - 5\underline{z}, \text{ at } \underline{t}_0 = 0, \underline{z}_0 = 1$$

Solution of Exact: $z(t) = \frac{(2e^{-5\underline{t}} + e^{-2\underline{t}})}{3}$

Table 2. Computational Outcome for Experiment 2

Points {t} of the **Training**	**Values** of Exact	**Values** of **ANN**	**ANN Error (Absolute)**
0.0	1.0000000000	0.9966232181	3.3767819×10^{-3}
0.1	0.6772640944	0.6730039120	4.2601824×10^{-3}
0.2	0.4686929882	0.4664441943	2.2487939×10^{-3}
0.3	0.3316906691	0.3313499391	3.407300×10^{-4}
0.4	0.2399998605	0.2397259176	2.739429×10^{-4}
0.5	0.1773498058	0.1765649021	7.849037×10^{-4}
0.6	0.1335894465	0.1327142864	8.751601×10^{-4}
0.7	0.1023305878	0.1018679067	4.626811×10^{-4}
0.8	0.0795092732	0.0795983225	8.90493×10^{-5}
0.9	0.0625056326	0.0628914014	3.857688×10^{-4}
1.0	0.0496037267	0.0497684479	1.647212×10^{-4}

A graphic depiction of the convergence of network predictions to the precise solution, at a given degree of precision, may be found in Table 2. The iterative process by which the network improves its predictions at each step and gets closer to the actual or precise solution is probably depicted in the table. For assessing the neural network's accuracy and dependability in approximating answers within the specified accuracy requirements, this convergence is essential. The entries in the table most likely show how, during training, the difference between the network's predictions and the precise answer decreased, highlighting the network's capacity to adjust its outputs until they almost match the required accuracy requirements. Table 2 is an invaluable resource for comprehending the dynamic and iterative characteristics of the neural network's learning process, demonstrating its ability to make precise predictions.

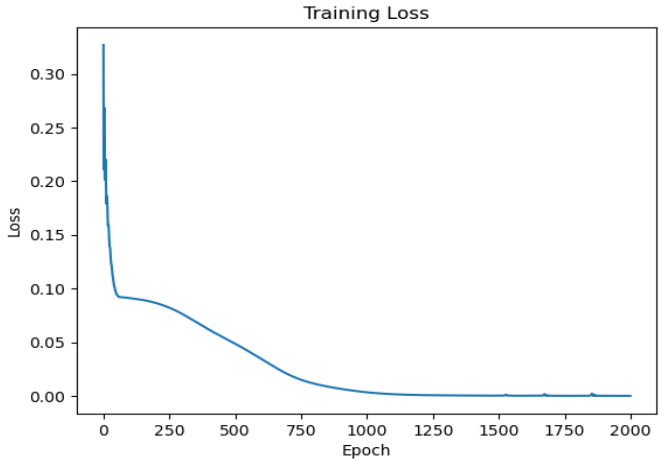

Fig. 5a. Loss Plot concerning Problem 2

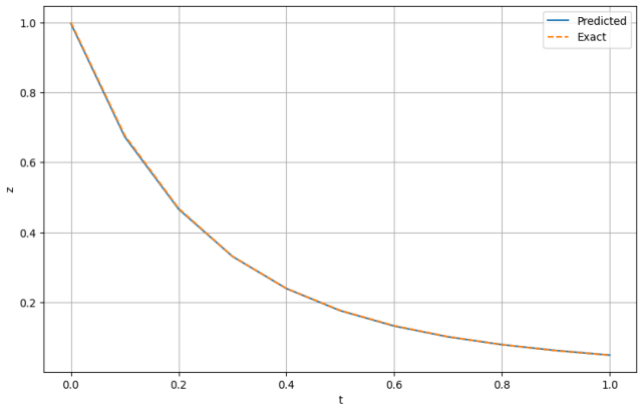

Fig. 5b. Exact and Predicted Values Plot concerning Problem 1

Here, the figure 5a shows that the network is learning and unlearning between 0.20-0.15 and at point 0.00 it perfectly learned the pattern all on the loss axis with 1000 iteration on the epoch axis. Furthermore, Figure 5b shows a plot of convergence of ANN to the exact solution at point 1.0 on the t-axis.

Experiment 3: This experiment engages the Artificial Neural Network approach to compute the predicted values concerning the ODE:

$$\frac{d\underline{z}}{d\underline{t}} = \frac{3}{2} - \frac{1}{2}\underline{z}$$

with respect to the initial condition stated as:

$$z(t_0) = 4$$

Solution of Exact: $z(t) = 3 + e^{-t/2}$

Table 3. Computational Outcome for Experiment 3

Points {t} of the **Training**	**Values** of Exact	**Values** of **ANN**	**ANN Error (Absolute)**
0.0	4.0000000000	4.0002832413	2.832413×10^{-4}
0.1	3.9512295723	3.9523916245	1.1620520×10^{-3}
0.2	3.9048373699	3.9059658051	1.1284350×10^{-3}
0.3	3.8607079983	3.8612987995	5.908012×10^{-4}
0.4	3.8187308311	3.8186030388	1.277923×10^{-4}
0.5	3.7788007259	3.7780096531	7.910728×10^{-4}
0.6	3.7408182621	3.7395720482	1.2462139×10^{-3}
0.7	3.7046880722	3.7032728195	1.4152527×10^{-3}
0.8	3.6703200340	3.6690378189	1.2822151×10^{-3}
0.9	3.6376280785	3.6367480755	8.800030×10^{-4}
1.0	3.6065306664	3.6062524319	2.782345×10^{-4}

Comparing the network predictions to the actual solution, it is impressive how accurate they are given the initial condition of 4. By training using the given initial condition, the neural network appears to create predictions that are quite similar to the actual values, as this remark shows. It is implied by the term "very significant level of accuracy" that the outputs of the neural network show a high degree of precision and consistency, effectively approximating the genuine solution under the given conditions. This finding highlights the efficiency of the Artificial Neural Network (ANN) method in producing precise outcomes, especially when the network's predictions are based on the given initial state. By contrasting the actual solution with the network predictions, it is possible to determine that the performance of the neural network in this specific situation.

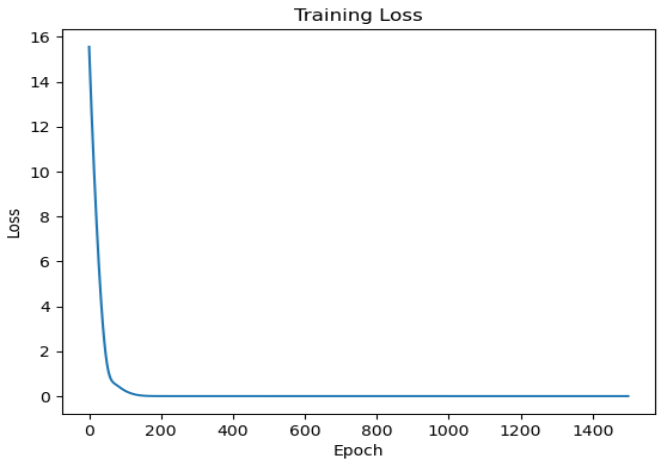

Fig. 6a. Loss Plot concerning Problem 3

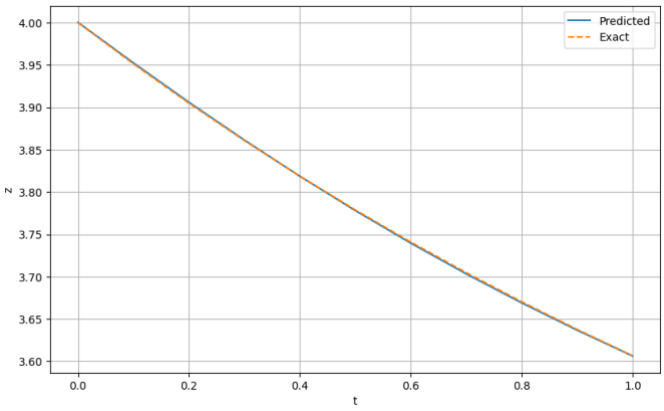

Figure 6b. Exact and Predicted Values Plot concerning Problem 3

Once the network completed 200 iterations on the epoch axis, Figure 6a shows that the pattern was correctly understood across all loss axes at the 0.00 point. The achievement of consistency and accuracy in its forecasts suggests a comprehensive learning process. Upon advancing to Figure 6b, where the time (t) axis is at 1.0, the network generates an output that exhibits surprising convergence to the true solution. With a convergence at 1.0 on the t axis, the neural network demonstrated its ability to produce correct outputs by roughly approximating the genuine answer. Figure 6a shows the network's learning trajectory, while Figure 6b shows the network's convergence with the real solution. The network's learning trajectory is illustrated by the graphs being compared. The analysis's overall findings demonstrate the network's effective learning and correctly estimating the answer to the given challenge.

4. CONCLUSION

The research team has made significant strides in developing an artificial neural network (ANN) architecture specifically designed for the resolution of first-order ODEs. The experimental results demonstrate that the proposed method exhibits a remarkable convergence rate while maintaining minimal errors in predicting outputs. A pivotal breakthrough occurred through the identification and implementation of an optimal learning rate within the ANN architecture. This optimal learning rate plays a crucial role in minimizing the loss function, showcasing a delicate balance that facilitates both accuracy and efficiency in the solution approximation process.

When evaluating the ANN-based model, the quantitative metric used is the 'Root Mean Square Error.' The computed Root Mean Square Error value of 0.006508786788 reflects the average deviation between the true function values and the predictions made by the neural network. Remarkably, the RMSE value is exceptionally small (close to zero), indicating that, on average, the neural network's predictions closely align with the true function values. This high accuracy suggests that the neural network effectively approximates values for any given function.

The method's ability to swiftly converge to solutions for first-order ODEs is noteworthy and holds promise for applications in various scientific and technical fields. The convergence's rapidity signifies the efficacy of the ANN-based

approach, providing an efficient computational tool for addressing complex mathematical problems.

Furthermore, the research underscores the broader implications of utilizing artificial neural networks for solving differential equations. The method's proficiency in minimizing errors and optimizing convergence rates positions it as a valuable asset in scenarios where computational efficiency is paramount. This research contributes to the evolving landscape of neural network methodologies in mathematical problem-solving, offering insights and potential applications in diverse domains.

In summary, the research study has shown that the constructed ANN architecture excels in solving first-order ODEs, exhibiting robust convergence capabilities and an optimized learning mechanism. These findings pave the way for future investigations into the diverse applications of ANNs in mathematical problem-solving. Such explorations hold promise for significant advancements in computational efficiency and precision, offering valuable insights for various fields reliant on numerical simulations and modeling. In future research, the focus will be on exploring how artificial neural networks (ANNs) can effectively solve higher-order ODEs. This involves developing specialized ANN architectures and optimizing training algorithms to address the complexities of higher-order ODEs. By conducting thorough experimentation and analysis, we aim to enhance our understanding of the capabilities of ANNs in resolving higher-order ODEs, contributing to advancements in computational mathematics and scientific modeling.

REFERENCES

[1] Mahata G., Raut D.S., Parida C., Baral S., & Mandangi S. (2022). Application of First-Order Differential Equations. International Journal of Engineering Science Technologies, 6(5), pp. 23-33.

[2] Lu L., Dagger, X. M., Dagger, Z. M., George, S. E. K. (2021). A Deep Learning Library for Solving Differential Equations. Society for Industrial and Applied Mathematics, 63 (1), pp. 208–228.

[3] Mao Z., Jagtap, A. D. & Karniadakis, G. E. Physics-informed neural networks for highspeed flows. Computational Methods in Applied Mechanics and Engineering, 360, pp. 209-225.

[4] Ji, X.A., Moln´ar, T.G., Avedisov, S.S., & Orosz, G. (2020). Feed-forward neural networks with trainable delay. Learning for Dynamics and Control, 120(1), pp. 127–136.

[5] Turan, E.M., & J¨aschke, J. (2021). Multiple shooting for training neural differential equations on time series. Control Systems Letters, 2(6), pp. 1897–1902.

[6] Li, S., Wang, X. (2021). Solving ordinary differential equations using an optimization technique based on training improved artificial neural networks. Soft Computing. 25(5), pp. 3713–3723.

[7] 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, pp. 900-918.

[8] Marchenko, N. A., Sydorenko, G. Y., & Rudenko, R. O. (2021). Using of Multilayer Neural Networks for the Solving Systems of Differential Equations. Bulletin of the National Technical University 'KhPI'. Series: System Analysis, Management, and Information Technologies, 2(6), pp. 125-129.

[9] Arunachalam, S. (2022). Applications of Artificial Neural Networks to Solve Ordinary Differential Equations. International Journal for Research in Applied Science and Engineering Technology, 10, pp. 882-888.

[10] Rehan, Z. (2022). Application of First-Order Differential Equation to Heat Convection in Fluid. Journal of Applied Mathematics and Physics, 8(8), pp. 1456-1462.

[11] Tigist Y., & Teketel K. (2020). Applications of First-Order Ordinary Differential Equation as Mathematical Model. Mathematical Theory and Modelling, 10(3), pp. 1-17.

[12] See, A. K. B. & On, J J.Y. (2023). Revolutionizing Motor Health: IoT-Driven Detection of Electrical Abnormalities in Three-Phase A.C. Induction Motors. Malaysian Journal of Science and Advanced Technology, 3(4), pp. 280-293

[13] Jiang, Z., Jiang, J., Yao, Q. & Yang., G. (2023). A neural network-based PDE solving algorithm with high precision.Scientific Reports, 13:4479.

[14] Chavez, H., Chavez-Arias, B., Contreras-Rosas, S., Alvarez-Rodríguez J. M.& Raymundo, C. (2023) Artificial neural network model to predict student performance using nonpersonal information. Frontier in Education. 8(1106679), pp 1-11.