Solving Ordinary Differential Equations Using Artificial Neural Networks: A Comprehensive Approach

Sourabh Kumar Dubey[1], Raghvendra Singh[2], Hibah Islahi[3]

[1]Research Scholar, Manglayatan University, Aligarh, U.P., India
[2]School of Sciences (Mathematics), Uttar Pradesh Rajarshi Tandon Open University, Prayagraj, U.P., India, Email**:** rsingh@uprtou.ac.in
[3]Institute of Applied Sciences, Manglayatan University, Aligarh, U.P., India

Received: 22.06.2024 Revised : 15.08.2024 Accepted: 16.09.2024

ABSTRACT
Ordinary Differential Equations (ODEs) are fundamental to modelling a wide range of phenomena across various scientific and engineering disciplines. Traditional numerical methods for solving ODEs, while effective, can be computationally intensive and may struggle with complex or high-dimensional problems. Recently, Artificial Neural Networks (ANNs) have emerged as a promising alternative for solving ODEs, leveraging their ability to approximate complex functions and patterns. This paper explores the application of ANNs in solving ODEs, presenting a detailed overview of various neural network architectures and training techniques used to address this problem. We discuss the advantages and limitations of ANNs compared to classical methods, present case studies demonstrating their effectiveness, and propose a framework for integrating ANNs into existing ODE-solving strategies. Our findings suggest that ANNs offer a flexible and efficient approach for certain classes of ODEs, paving the way for further research and practical applications.

Keywords: ANN, differential equations, ODE

1. INTRODUCTION

Ordinary Differential Equations (ODEs) play a fundamental role in mathematical modeling across a wide array of disciplines, including physics, engineering, biology, and finance [1]. These equations describe how a system evolves over time and are essential for predicting system behaviour, whether it's the motion of physical objects, the dynamics of biological processes, or financial market fluctuations. Solving ODEs accurately and efficiently is crucial for understanding and forecasting the behaviour of such systems.

Traditional numerical methods have been the cornerstone of solving ODEs. Techniques such as Euler's method [2], Runge-Kutta methods [3], and finite difference methods [4] have been extensively utilized to obtain approximate solutions. Euler's method offers a straightforward approach by approximating solutions through a series of discrete steps, while the Runge-Kutta methods, including the popular fourth-order variant, provide higher accuracy by considering multiple points within each step. Finite difference methods are widely used for discretizing differential equations and solving them on grids. Although these methods are well-established and widely applicable, they can encounter significant challenges when applied to high-dimensional problems, stiff equations, or complex boundary conditions. High-dimensional problems can lead to increased computational costs and complexities in ensuring numerical stability. Stiff equations, which exhibit rapidly varying solutions, often require specialized techniques to manage their inherent numerical difficulties.

In recent years, the advent of Artificial Intelligence (AI) and machine learning has introduced innovative techniques for addressing the limitations of traditional methods. Among these, ANNs [5] have emerged as a powerful tool for solving ODEs. ANNs, inspired by the neural architecture of the human brain, consist of interconnected nodes (neurons) organized into layers. These networks are capable of learning complex patterns from data and making predictions based on this learned information. This inherent flexibility and adaptability make ANNs a promising alternative for solving differential equations, particularly when traditional methods become cumbersome or are not feasible.

The integration of ANNs into the process of solving ODEs involves training the network to either approximate the solution directly or to learn the underlying patterns from data generated by numerical methods. By feeding the network with inputs related to the differential equation and desired outputs, the ANN learns to predict solutions with a high degree of accuracy. This approach can offer several

advantages, such as reduced computational costs and the ability to handle complex or high-dimensional problems more efficiently than conventional numerical methods [12-14].
The optimization techniques and conditions with fuzzy-based approach provides greater flexibility by using fuzzy sets and fuzzy numbers to represent uncertain data, allowing for more informed decision-making under uncertain conditions [15-22].
This paper provides a detailed overview of how ANNs can be applied to solving ODEs, including an examination of various neural network architectures like feedforward networks, recurrent networks, and deep learning models. Feedforward networks are typically used for straightforward function approximation tasks, while recurrent networks are well-suited for problems involving sequences or time-dependent data. Deep learning models, with their multiple layers and advanced architectures, can capture intricate patterns and improve solution accuracy. We also discuss training methodologies for ANNs, such as backpropagation and optimization techniques, and analyze the trade-offs associated with using ANNs compared to classical numerical methods. Through illustrative case studies, we demonstrate the practical application of ANNs in solving ODEs and propose a framework for future research to explore and expand upon these findings. Our aim is to highlight the potential of ANNs to enhance both the efficiency and accuracy of ODE solutions, paving the way for further advancements in this field.

2. Introduction to ANN

ANNs [6] are sophisticated computational models designed to simulate the information processing capabilities of biological neural networks found in the human brain. These networks are structured to perform complex tasks such as classification, regression, and other machine learning functions by learning from data. The fundamental architecture of an ANN consists of multiple interconnected layers of neurons, each contributing to the network's ability to model and solve various problems [7].
The architecture of an ANN typically includes three main types of layers: the input layer, hidden layers, and the output layer.

1. **Input Layer**: Receiving raw data is the responsibility of this first layer of the network. Information enters the network through each neuron in the input layer, which is matched to a feature of the input data.
2. **Hidden Layers**: One or more hidden layers receive the input data after it is transmitted from the input layer. Neurones in each hidden layer process inputs by applying a weighted sum to them. After that, this total is run via an activation function. The network gains non-linearity from the activation function, which enables it to understand and depict intricate interactions between inputs and outputs. The hidden layers transform the input data in increasingly abstract ways as it progresses through the network [8].
3. **Output Layer**: The final layer of the network is the output layer. It receives the processed information from the last hidden layer and produces the network's final result or prediction. The output layer's structure and activation function depend on the specific task. For example, in classification tasks, the output layer may use a softmax function to provide probabilities for different classes, whereas, in regression tasks, a linear activation function may be used to predict continuous values.

An ANN learns by varying the weights of connections among its neurons in order to reduce the discrepancy between the goal values and the projected output. Usually, this is accomplished by a process known as backpropagation, which iteratively updates the weights depending on the error gradients using gradient descent. By continually adjusting these weights, the network learns to approximate the desired function or pattern from the input data, thereby improving its performance on tasks such as prediction and classification [7].
Overall, the ability of ANNs to process and model complex data through their layered architecture and learning mechanisms makes them powerful tools in various applications, from image and speech recognition to financial forecasting and beyond.

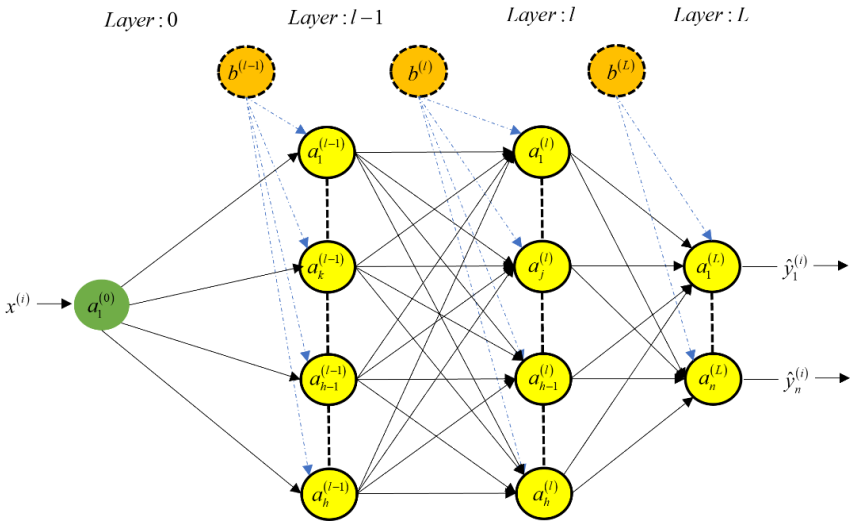

Figure 1: Structure of ANN

One of the most basic and popular network topologies is the feedforward neural network, which is a particular kind of ANN architecture (Figure 1). Without any cycles or feedback loops, information only flows in one direction in a feedforward neural network: from the input layer via the hidden layers to the output layer. Because of their unidirectional nature, feedforward networks are easy to create and comprehend. Because every neuron in a layer is fully connected to every other neuron in the layer above it, the network's many layers enable it to learn intricate mappings of inputs to outputs.The activation function plays a crucial role in determining the output of each neuron. One commonly used activation function is the hyperbolic tangent (tanh) function. The tanh function is defined by the formula [9]:

$$\tanh(x) = \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \right) \tag{1}$$

This function maps input values to a range between -1 and 1, which helps center the data and can improve the training dynamics of the network. The tanh function is symmetric around the origin, which makes it particularly useful in scenarios where inputs are both positive and negative. Additionally, the derivative of the tanh function is:

$$\tanh'(x) = 1 - \tanh^{2}(x) \tag{2}$$

This derivative is essential for the backpropagation algorithm, which is used to update the weights of the network based on the error of the predictions. The tanh function's derivative allows for efficient gradient computation, which in turn facilitates the learning process. However, the tanh function, like other sigmoid-based activation functions, can suffer from the vanishing gradient problem, where gradients become very small for large positive or negative inputs, potentially slowing down learning. Despite this, the tanh function remains a popular choice in feedforward neural networks due to its effective data scaling and symmetry properties, contributing to the overall performance and learning capability of the network.

3. Solution of Differential Equation using ANN

The first order differential equation can be written as

$$\frac{d\psi}{dx} = f(x, \psi) \tag{3}$$

where f is a continuous function, $x \in [0,1]$ and $\psi(0) = \alpha$

In the initial step trial solution of the form

$$\psi(x) = \alpha + xN(x, \vec{\beta}) \tag{4}$$

where $N(x, \vec{\beta}) = \sum_{i=1}^{h} v_i \sigma(z_i)$ (5)

In the above equation σ is activation function. The output of the ANN can be modelled as

$$z_i = \omega_i x + u_i \tag{6}$$

Differentiating equation 4 we get,

$$\frac{d\psi(x)}{dx} = \frac{d}{dx}\left(\alpha + xN(x,\vec{\beta})\right) = N(x,\vec{\beta}) + x\frac{dN(x,\vec{\beta})}{dx} \tag{7}$$

Substituting equation 5 in equation 7 we get,

$$\frac{d\psi(x)}{dx} = \sum_{i=1}^{h} v_i\sigma(z_i) + x\sum_{i=1}^{h} v_i\omega_i\sigma(z_i) \tag{8}$$

Finally, the cost function is modelled as

$$J(\vec{\beta}) = \min\sum_k \left[\frac{d\psi(x_k)}{dx} - f\left(x_k, \psi_k(x_k)\right)\right]^2, \qquad x_k \in [0,1] \tag{9}$$

4. Simulation and Results

To solve a system of two first-order coupled differential equations using an ANN, we start by modelling the problem with a specific neural network architecture [10,11]. The system of equations is defined as $\frac{dy_1}{dt} = f(x, y_1, y_2)$ and $\frac{dy_2}{dt} = f(x, y_1, y_2)$, where x is the input variable and y_1 and y_2 are the dependent variables that we wish to determine.The ANN architecture for solving this system includes an input layer with a single neuron to handle the input variable x. This input is then passed to a hidden layer comprising 10 neurons. In this hidden layer, the tanh (hyperbolic tangent) activation function is used. The tanh function is beneficial here due to its range of [−1,1], which helps in introducing non-linearity and can improve the network's ability to model complex relationships between the input and the outputs.

The network's final layer, known as the output layer, consists of two neurons. These neurons are responsible for producing the output values y_1 and y_2. In the output layer, a linear activation function is often employed to ensure that the outputs can take any real value, which is suitable for continuous outputs.

During the training phase, the network learns to approximate the solution to the differential equations. The process involves comparing the network's output y_1 and y_2 against expected values derived from solving the differential equations. A loss function, which measures the discrepancy between the predicted outputs and the actual values, is minimized through backpropagation and optimization techniques. This iterative process adjusts the weights and biases in the network to improve its performance in solving the differential equations.

$$\frac{dy_1}{dx} = 198y_1 + 199y_2$$

$$\frac{dy_2}{dx} = -398y_1 - 399y_2$$

$$y_1(0) = 1, \quad y_2(0) = -1 \quad \text{and } x \in [0,5]$$

The analytical solution of the equation is

$y_1(x) = e^{-x}$ and $y_2(x) = -e^{-x}$

After completing 120 iterations of training, both the analytical solution and the ANN-based solution for the parameter y_1 have been compared and are illustrated in Figure 2. In this context, the analytical solution refers to the exact solution obtained through mathematical techniques, while the ANN-based solution represents the output of the neural network after its training process.

The results presented in Figure 2 show that the curves corresponding to the analytical solution and the ANN-based solution for y_1 are remarkably similar. In fact, they are so closely aligned that they appear to be superimposed on one another. This indicates a high level of agreement between the results obtained from the traditional analytical method and those predicted by the neural network. The near-identical nature of the two curves suggests that the ANN has effectively learned the underlying dynamics of the system and is capable of approximating the solution with a high degree of accuracy after the specified number of iterations.

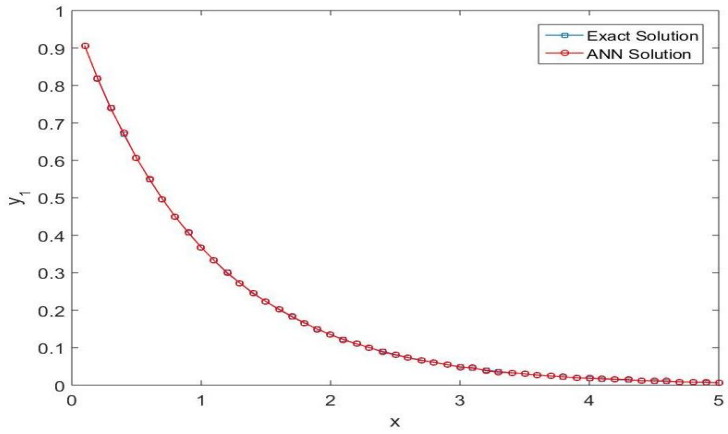

Figure 2: Solution for parameter y_1

Figure 3 presents a detailed view of the error between the exact solution and the ANN-based solution for 50 distinct points across the range of input values. In this analysis, the input values *x* vary from 0 to 5, with increments of 0.1, resulting in a total of 50 equally spaced points.
At each of these 50 points, the difference between the exact analytical solution and the solution provided by the artificial neural network (ANN) is calculated. This difference is referred to as the error. The graph depicted in Figure 3 illustrates how this error varies across the range of input values.

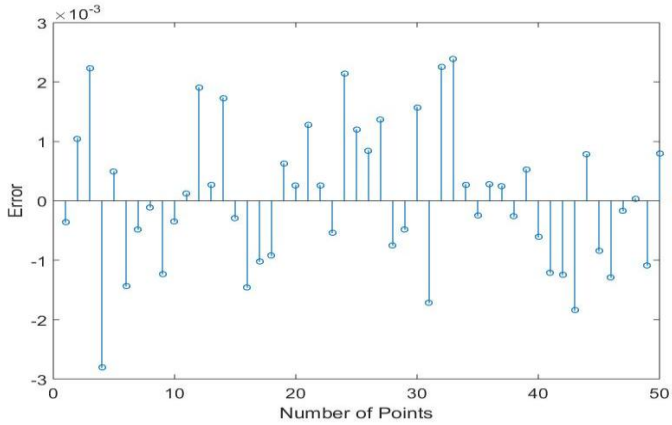

Figure 3: Error for parameter y_1

The results show that the maximum error observed between the exact solution and the ANN solution is 2.88×10^{-3}. This value quantifies the largest discrepancy between the two solutions at any of the 50 points considered. An error of 2.88×10^{-3} (or 0.00288) indicates a very small difference, demonstrating that the ANN has achieved a high level of accuracy in approximating the exact solution over the specified range.

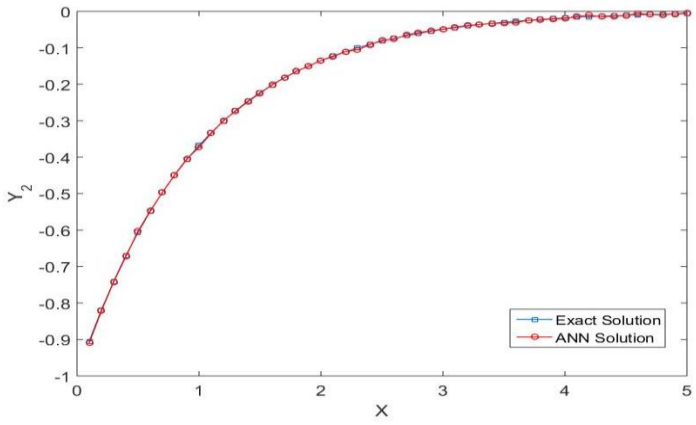

Figure 4: Solution for parameter y_2

Figure 4 showcases a comparison between the exact solution and the ANN-based solution for the parameter y_2. The graph illustrates the results for various values of *x*, spanning from 0 to 5 with increments of 0.1, generating a total of 50 discrete data points. The curve representing the ANN-based solution is plotted alongside the curve for the exact solution. This visual comparison allows for an evaluation of how closely the neural network's predictions match the exact results across the entire range of input values.

Figure 5 provides a detailed representation of the error between the exact and ANN-based solutions for y_2. The error is calculated at the same 50 points within the range of *x*, and the graph displays the variation in error values across these points. The maximum error observed in this analysis is 0.005530. This value represents the largest deviation between the exact solution and the ANN approximation at any of the 50 points considered.

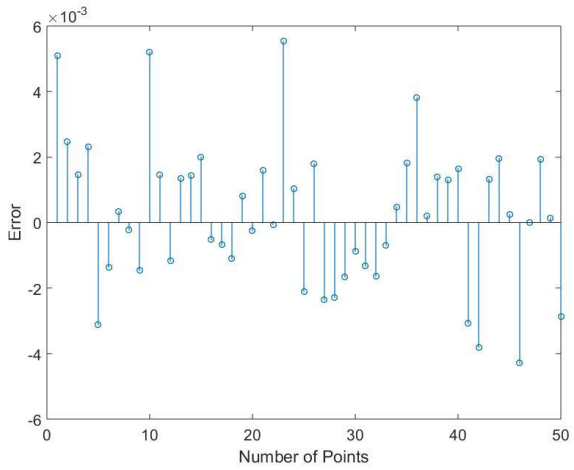

Figure 5: Error for parameter y_2

5. CONCLUSION

In this paper, we investigated the use of ANNs to solve ODEs, focusing on a system of two first-order coupled equations. We designed an ANN with one input neuron, ten hidden units with the tanh activation function, and two output neurons to predict the dependent variables y_1 and y_2.Our results indicate that the ANN effectively approximates the solutions to the ODEs. The predictions for y_1 closely matched the exact analytical solutions, demonstrating the network's ability to learn and model the system dynamics with high accuracy. For y_2, while the maximum error observed was slightly larger, it still remained within an acceptable range, confirming the overall reliability of the ANN approach.The comparison between the ANN-based and analytical solutions, along with the minimal observed errors, highlights the potential of ANNs as a robust tool for solving differential equations. This work not only validates the feasibility of using neural networks for such problems but also suggests opportunities for further refinement and application to more complex systems. Future research could focus on enhancing the network's accuracy and exploring its use in a broader range of differential equation problems.

REFERENCES

[1] Coddington, Earl A. An introduction to ordinary differential equations. Courier Corporation, 2012.
[2] Nurujjaman, Md. "Enhanced Euler's Method to Solve First Order Ordinary Differential Equations with Better Accuracy." Journal of Engineering Mathematics & Statistics 4, no. 1 (2020): 1-13.
[3] Arora, Geeta, Varun Joshi, and Isa Sani Garki. "Developments in Runge–Kutta method to solve ordinary differential equations." In Recent Advances in Mathematics for Engineering, pp. 193-202. CRC Press, 2020.
[4] Ma, Yingbo, Vaibhav Dixit, Michael J. Innes, Xingjian Guo, and Chris Rackauckas. "A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions." In 2021 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1-9. IEEE, 2021.
[5] Shi, Enze, and Chuanju Xu. "A comparative investigation of neural networks in solving differential equations." Journal of Algorithms & Computational Technology 15 (2021): 1748302621998605.

[6] Choi, Rene Y., Aaron S. Coyner, Jayashree Kalpathy-Cramer, Michael F. Chiang, and J. Peter Campbell. "Introduction to machine learning, neural networks, and deep learning." Translational vision science & technology 9, no. 2 (2020): 14-14.
[7] Guillod, Thomas, PanteleimonPapamanolis, and Johann W. Kolar. "Artificial neural network (ANN) based fast and accurate inductor modeling and design." IEEE Open Journal of Power Electronics 1 (2020): 284-299.
[8] Zhang, Jing, and Guanghui Su. "Artificial neural network introductions." In Nuclear Power Plant Design and Analysis Codes, pp. 515-541. Woodhead Publishing, 2021.
[9] Dağlı, Muhammet Cihat, and Feng Qi. "Several recurrence relations and identities on generalized derangement numbers." Results in Nonlinear Analysis 5.2: 185-190.
[10] Dua, Vivek, and Pinky Dua. "A simultaneous approach for parameter estimation of a system of ordinary differential equations, using artificial neural network approximation." Industrial & engineering chemistry research 51, no. 4 (2012): 1809-1814.
[11] Bradley, William, and FaniBoukouvala. "Two-stage approach to parameter estimation of differential equations using neural odes." Industrial & Engineering Chemistry Research 60, no. 45 (2021): 16330-16344.
[12] Singh.R, "A Study of Independent Component Analysis in Neural Networks" International Journal of Science and Engineering, Volume 2, No. 1(2023), pp. 11-20.
[13] Singh.R, "Neural networks based face recognition system for biometric security" Indian journal of Engineering, Volume 20, Issue 53, pp 1-9, January-June,2023,doi: https://doi.org/10.54905/disssi/v20i53/e16ije1640.
[14] Shukla. K. A, Shukla. A, Singh. R "Automatic attendance system based on CNN–LSTM and face recognition", International Journal of Information Technology,Volume 15, Issue 8, (2023), pp 1293–1301,https://doi.org/10.1007/s41870-023-01495-1
[15] Kumar, P., Yadav, V., Naik, P. J., Malik, A. K., & Alaria, S. K. (2023, June). Analysis of fuzzy inventory model with sustainable transportation. In AIP Conference Proceedings (Vol. 2782, No. 1). AIP Publishing.
[16] Malik, A.K., Yadav, S.K. and Yadav, S.R. (2012) Optimization Techniques, I. K International Pub. Pvt. Ltd., New Delhi.
[17] Sadulla, Shaik. "Next-Generation Semiconductor Devices: Breakthroughs in Materials and Applications." Progress in Electronics and Communication Engineering 1.1 (2024): 13-18.
[18] Tyagi, T., Kumar, S., & Malik, A. K. (2023). Fuzzy inventory system: A review on pharmaceutical and cosmetic products. Research Journal of Pharmacy and Technology, 16(7), 3494-3498.
[19] Tyagi, T., Kumar, S., Malik, A. K., & Vashisth, V. (2023). A novel neuro-optimization technique for inventory models in manufacturing sectors. Journal of Computational and Cognitive Engineering, 2(3), 204-209.
[20] Verma, P., Chaturvedi, B. K., & Malik, A. K. Comprehensive Analysis and Review of Particle Swarm Optimization Techniques and Inventory System, International Journal on Future Revolution in Computer Science & Communication Engineering, 2022; 8(3), 111-115.
[21] Yadav, S.R. and Malik, A.K. Operations Research, Oxford University Press, New Delhi, 2014.
[22] Yadav, V., Chaturvedi, B. K., & Malik, A. K. Advantages of fuzzy techniques and applications in inventory control. International Journal on Recent Trends in Life Science and Mathematics, 2022; 9(3), 09-13.