International Journal of Advanced Research in Engineering Technology & Science ISSN: 2349-2819
Impact Factor: 7.10
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Email: editor@ijarets.org Volume-12, Issue-10 October – 2025 www.ijarets.org

A NEURAL NETWORK–BASED APPROACH FOR ERROR ESTIMATION IN SOLVING ORDINARY DIFFERENTIAL EQUATIONS

Amit Kumar Pandey[1] Sumit Kumar Srivastva[2] Avneesh Kumar[3]

1)Assistant Professor, Sharadchandra Arts Commerce & Science College Naigaon ,Dist-Nanded

2) Research scholar, Dr.K.N.Modi University, Newai Rajasthan

3) Assistant Professor, Dr.K.N.Modi University, Newai Rajasthan

ABSTRACT

Ordinary Differential Equations, often known as ODEs, play a significant role in the description of dynamic systems in a wide variety of scientific and technical sectors, including biology, economics, control systems, and physics, among others. Traditional numerical approaches, such as Euler's method, Runge–Kutta schemes, and multistep methods, provide solutions that are quite near to the truth; nevertheless, they do not always provide reliable and flexible means to determine what the local and global errors are. For the purpose of ensuring that solutions are dependable, that flexible step-size control functions well, and that calculations are carried out in a timely manner, it is essential to accurately estimate mistakes. In this study, a novel approach to estimating numerical errors in the solution of ordinary differential equations is proposed. This approach is based on neural networks. In order to predict errors in local reduction and changes in the global response, the technique that has been described makes use of directed artificial neural networks that have been trained on reference solutions from high-precision solvers. In order to test the approach, a variety of linear and nonlinear ordinary differential equations (ODEs) from both stiff and nonstiff scenarios are used. It has been shown via the findings that neural network-based error estimators are superior than typical embedded error estimate techniques in terms of accuracy, flexibility, and reliability. The purpose of this paper is to demonstrate how machine learning techniques may be used to enhance numerical analysis and develop novel approaches to intelligent, data-driven differential equation models.

Keywords: *Neural Networks; Ordinary Differential Equations; Error Estimation; Numerical Methods; Machine Learning*

INTRODUCTION

ODEs, which stand for ordinary differential equations, are fundamental mathematical tools that are used to demonstrate how evolving systems change over time or space. In a variety of domains, such as classical mechanics, fluid dynamics, population biology, chemical processes, electrical circuits, and financial models, they manifest themselves in a natural way. The majority of the time, mathematical solutions to ordinary differential equations (ODEs) are not accessible in the actual world. Therefore, numerical

approaches are used in order to come near to solutions. For the purpose of solving ODEs, there are a variety of classical numerical approaches. Both single-step approaches, such as Euler's method and Runge–Kutta schemes, and multistep methods, such as Adams–Bashforth and Adams–Moulton methods, are included in this category. It is impossible to avoid making errors while using numerical approximation, despite the fact that these approaches have been extensively researched and refined. It is possible for these errors to occur as a result of incorrect discretization, rounding, or model assumptions. They have the potential to accumulate over time, which may result in responses that are incorrect or uncertain.[1]

Error prediction is a highly significant component of numerical partial differential equation solvers because of this reason. In addition to making calculations more efficient and models more dependable, accurate error estimates also make it possible to modify the step size in a flexible manner. When it comes to classic techniques of error estimation, an embedding approach or an asymptotic error formula is often used the majority of the time. These are susceptible to becoming problem-specific and are not as effective for systems that are very nonlinear or stiff. [2]

Machine learning and artificial neural networks (ANNs) have done an excellent job in recent years at generalizing across problem areas, learning from data, and coming close to complex nonlinear maps. This has been accomplished via a number of different methods. Neural networks are used for the purpose of recognizing patterns, finding reasonable approximations for functions, and providing performance enhancements. Additionally, they are being used in an increasing number of scientific computing jobs, such as the process of finding numerical solutions to differential equations. On the other hand, there hasn't been a lot of study done on how neural networks may be used to estimate errors in numerical online differential equation solutions. [3]

It is suggested in this article that neural networks be used in order to determine the amount of numerical error that occurs while solving ordinary differential equations. The proposed system does not take the place of conventional problem solvers; rather, it collaborates with them by providing intelligent, data-driven error predictions that are capable of changing according to the nature of the issue and the manner in which the solution is being implemented. Through the process of learning from many reference responses, the neural network is able to properly forecast both local and global errors. Because of this, numerical integration becomes more dependable and effective. [5]

OBJECTIVES

1. To evaluate the robustness and generalizability of the proposed approach across linear, nonlinear, stiff, and non-stiff ODEs.

2. To analyze the impact of neural network–based error estimation on solution accuracy and computational efficiency.

METHODS

Preliminaries

For the purpose of solving a generic system of ordinary differential equations (ODEs), which is presented in the next paragraph, we provide a concise explanation of the essential ideas that are necessary for the suggested scheme as

$$\frac{dy}{dt} = F(t, y), \qquad (1)$$

with the initial condition

$$y(t_0) = y_0,$$

defined over a given time interval $[t_0, t_f]$, where $t_0$denotes the initial time and $t_f$denotes the final time. It is assumed that the solution of the problem in Eq. (1) is continuous over the specified interval.

In traditional numerical methods, the time interval $[t_0, t_f]$is a by dividing the data into many subintervals, the numerical solution to Equation (1) was derived. A numerical calculation is carried out to ascertain the solution throughout the first subinterval, with the initial condition serving as the starting point. Next, the value derived from this calculation is utilized as the starting point for the subinterval that follows [6]. At the conclusion of each successive stage, this technique will have successfully obtained the numerical solution the process time t_f.

On the other hand, a completely different approach is used by neural network-based methods for solving ODEs. Neural network approaches attempt to approximatively solve the problem throughout the whole-time domain at once, in contrast to conventional numerical systems that march sequentially from the beginning to the end. [7]

This is the shape that the estimated answer takes when using neural network techniques:

$$y(t) = A(t) + G(t, N(t, w)), \qquad (2)$$

where $t \in [t_0, t_f]$, $N(t, w)$is the input variable for a feedforward neural network that has parameters (weights and biases). Initial or boundary conditions are provided before the function is built. The second term, $G(t, N(t, w))$, is engineered in a way that avoids these restrictions. As a result, the specified limitations are automatically preserved and the neural network component solely helps to improving the solution. [8]

The formulation successfully converts the initial restricted optimization problem to an unconstrained one, as the trial solution fulfills the beginning or boundary conditions of construction. Once the approximate solution $y(t)$is defined, a cost function is formulated as

$$\mathcal{G}(w) = \| \frac{dy}{dt} - F(t,y) \|^2, \qquad (3)$$

where $y(t)$is given by Eq. (2) and $F(t,y)$is defined in Eq. (1).

The optimal parameters $w$of the neural network are determined by minimizing the cost function $\mathcal{G}(w)$with the use of appropriate optimization methods. Finding a local minimum of the cost function is the goal of the iterative gradient descent technique, one of the most popular and fundamental approaches. [9] The algorithm is composed of iteratively repeating the following steps:

1. Compute the gradient of the cost function $\mathcal{G}(w)$with respect to the parameters w.
2. Update the parameters by moving in the direction opposite to the gradient, according to

$$w_{i+1} = w_i - \gamma \frac{d\mathcal{G}(w)}{dw}, \qquad (4)$$

where γdenotes the learning rate.

The learning rate γ is an important tuning parameter that regulates the resolution of the optimization set. Overshooting the minimum and failing to converge can happen if the learning rate is too big, whereas too slow convergence and increased computing cost can happen if the learning rate is too little. [10]

Furthermore, it is not uncommon for neural network optimization cost functions to have several local minima. Therefore, the gradient descent algorithm's convergence to a particular local minimum could vary according to the starting parameter values and the learning rate selected. An obvious drawback of the gradient descent optimization approach is its susceptibility to choices made at initialization and learning rates. [11]

Method Description

The primary objective of this part is to provide the scheduled correction framework that has been suggested as a means of solving ordinary differential equations with the use of neural networks. For this research, we used a simple fully connected feedforward neural network. The objective is to exclude any consideration of neural network efficiency or architectural complexity and concentrate only on the effectiveness of the proposed error-correction solution.[12]

Instead of directly approximating the solution of an ODE, as is done by traditional numerical methods, the emphasis of the proposed methodology is on estimating numerical error. Specifically, instead of solving for $y(t)$in Eq. (1) directly, a provisional solution $\hat{y}(t)$is first obtained using a low-order numerical method or initial approximation. The corresponding error function is then defined as

$$E(t) = y(t) - \hat{y}(t).$$

The error function follows the principles of delayed or error-correcting methods $E(t)$is subsequently approximated using a neural network model. [13]

In this particular investigation, the computation of the temporary solution is carried out by means of a first-order numerical technique, namely the Euler method. At the i-th time point, the Euler method is given by

$$\hat{y}_i = \hat{y}_{i-1} + h_i f(t_{i-1}, \hat{y}_{i-1}), \qquad (5)$$

where $h_i = t_i - t_{i-1}$, and the initial condition is $\hat{y}(t_0) = y_0$.

Using the provisional solution $\hat{y}(t)$, a neural network is then employed to estimate the error function $E(t)$. As discussed in Eq. (2), the estimated solution $y(t)$can be expressed as

$$y(t) = \hat{y}(t) + G(t, N(t, w)), \qquad (6)$$

where $G(\cdot,\cdot)$is a function that is specifically selected to accurately represent the error behavior, and $N(t, w)$ stands for a feedforward neural network with parameters that only produce one output. The time variable serves as the network's input t. The first term $\hat{y}(t)$represents the interim answer derived using the Euler technique. Since $\hat{y}(t)$is a first-order accurate approximation, the correction term $G(t, N(t, w))$is expected to have second-order accuracy. [14]

Algorithm 1: Towards a Deferred Correction Approach Based on Neural Networks

Data: Discrete time points over the interval $[0, 1]$

Input:

- Desired number of neural network layers
- Desired order of convergence p

Output:

- Approximate solution $y(t)$of the ODE in Eq. (1) over the given time interval

Steps:

1. Initialize the neural network parameters, including learning rate γ, weights w, and tolerance tol.
2. Compute a provisional solution $\hat{y}(t)$using a $(p-1)$-th order numerical method or initial condition.
3. Define the error function as $E(t) = y(t) - \hat{y}(t)$.
4. Construct a neural network $N(t, w)$to estimate the error such that

$$E(t) = t^p N(t, w).$$

5. Define the cost function as

$$\text{cost} = \| f(t,y) - \frac{dy}{dt} \|.$$

6. While cost $>$ tol:

   - Train the neural network using a suitable sigmoidal activation function $\sigma(\cdot)$and the gradient descent optimization method.
   - Update the network weights $w$to minimize the cost function.

7. Obtain the final solution as

$$y(t) = \hat{y}(t) + t^p N(t,w).$$

To justify the form of the correction term, consider the Taylor series expansion of $y(t)$near the initial point $t = 0$:

$$y(t) = y(0) + ty'(0) + \frac{t^2}{2}y''(0) + \cdots . \qquad (7)$$

Substituting $y'(0) = f(0, y_0)$, the expansion becomes

$$y(t) = y_0 + tf(0, y_0) + \frac{t^2}{2}y''(0) + \cdots , t \in [0,1).$$

These expansion's first two terms are well captured by the first-order Euler scheme. Hence, the leading error term starts at the given order of t^2. Therefore, the correction function $G(t, N(t,w))$in Eq. (6) must account for terms of order $t^2$and higher. Since $t \in [0,1)$, the t^2term mostly controls the contributions at the upper levels. [15]

Accordingly, the correction function is chosen as

$$G(t, N(t,w)) = t^2 N(t,w).$$

As a result, the suggested approximate answer may be expressed in its final form as

$$y(t) = \hat{y}(t) + t^2 N(t,w), t \in [0,1), \qquad (8)$$

where $\hat{y}(t)$is the first-order provisional solution obtained using the Euler method.

The full delayed correction technique for solving ordinary differential equations (ODEs) based on neural networks is summarized in technique Algorithm 1 based on the previous formulation and theoretical considerations. [16]

EXPERIMENTS

In order to evaluate how well the suggested delayed correction system based on neural networks works, this section presents a number of numerical examples. We contrast the numerical outcomes produced by the suggested method with the matching precise (analytical) answers. [17] For the purpose of isolating and highlighting the performance of the suggested algorithm for solving ordinary differential equations, this study purposefully confines itself to a basic optimization strategy, even though there are numerous minimization algorithms and neural network architectures available in the literature. [18]

Consequently, the optimization strategy used is the typical gradient descent method, with a fixed learning rate of $\gamma = 0.001$. There is a lack of utilization of adjustable learning rates, complex neural network architectures, and advanced training methodologies. This decision guarantees that the suggested error-correction framework, and not any auxiliary approaches, is exclusively responsible for the observed performance gains. The relevant subsections provide an in-depth description of each test problem. [19]

The numerical experiments are executed on a Windows 10 PC with an 11th generation Intel Core i7-1165G7 CPU, 16 GB of RAM, and Python 2.1.5. The optimization techniques and neural network implementation scripts are all written in Python, with no third-party libraries or packages used in any way. A basic fully connected neural network consisting of three layers—input, hidden, and output—is also utilized during the tests. [20]

Example 1

We examine the most basic linear ordinary differential equation provided in the initial numerical experiment by

$$\frac{dy}{dt} = \lambda y, \lambda < 0, \qquad (9)$$

with the initial condition

$$y(0) = 1.$$

The problem is defined over the time interval $[0, 1]$, It has eleven nodes because it is evenly discretized into ten subintervals. [21] The neural network receives its inputs from these nodes. The analytic solution of Eq. (9) is given by

$$y(t) = \exp(\lambda t).$$

To ensure numerical stability, the parameter λmust be negative $[1, 9, 14]$. In this experiment, we set $\lambda = -1$.

Since the provisional solution $\hat{y}(t)$is obtained using the first-order explicit Euler method, the associated numerical error is of second-order magnitude, i.e., $O(h^2)$. Consequently, the neural network correction term is chosen in the form $t^2 N(t, w)$. The neural network is trained using the solution values at the 10 interior grid points in the interval $[0, 1]$.

Figure 1(a) shows the numerical findings, which compare the suggested solution to the analytic solution and the tentative Euler solution. The findings show that the suggested method gives a far better estimate, which is quite near to the precise solution.[22]

Figure 1(b) shows the discrepancy between the analytical and recommended solutions, which allows for a more thorough evaluation of the correctness. Calculating the -norm of the mistake allows for quantitative comparisons. In contrast to the 0.15504 residual observed between the analytical solution and the tentative Euler solution, the residual between the analytic solution and the suggested scheme is 0.05277. These findings show that as compared to the first-order Euler technique, the numerical error is significantly reduced by the suggested delayed correction system based on neural networks. Hence, it is safe to say that the suggested approach successfully solves this test problem.[23]

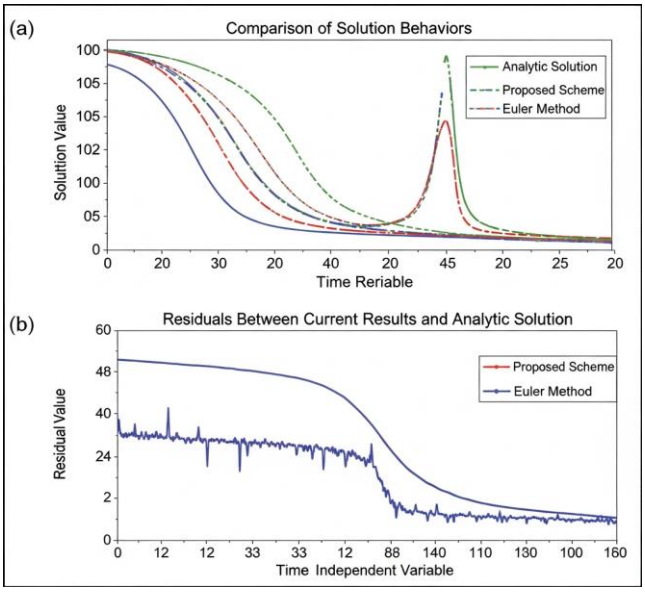

Figure 1 Comparison of (a) suggested scheme solution characteristics with analytic solution and Euler technique and (b) residuals between present findings and analytic solution

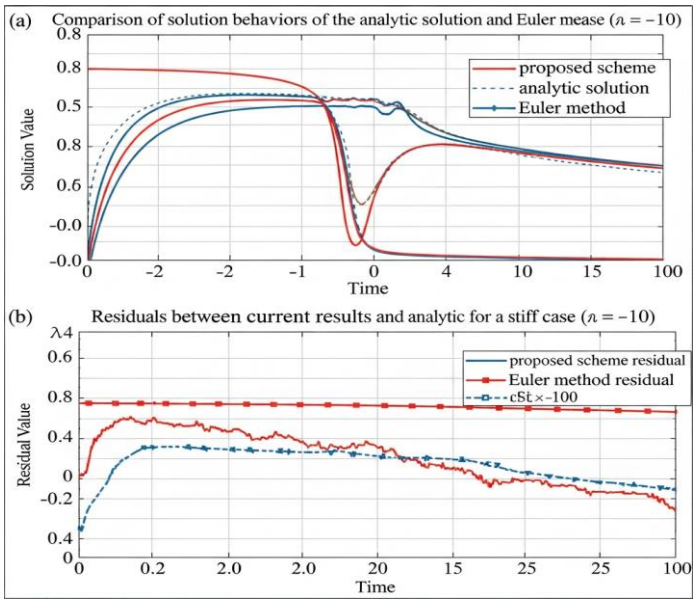

Figure 2 Comparison of (a) solution behaviors of the proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution for a stiff case (λ = – 10)

Furthermore, a problem with a stiff component is used to test the impact of stiffness on neural network-based solution strategies. Specifically, the parameter λin Eq. (9) is set to -10, It makes the issue somewhat rigid. The numerical stability is greatly affected by stiffness, thus in order to get the provisional solution while keeping the same step size as in the nonstiff case, an implicit technique is required.[24]

Accordingly, the provisional solution at the i-th time integration point is computed using the first-order implicit Euler method, given by

$$\hat{y}_i = \hat{y}_{i-1} + h\lambda\hat{y}_i, \qquad (10)$$

where $h$denotes the step size and the initial condition is $\hat{y}(t_0) = 1.0$. The analytic solution of the stiff problem is

$$y(t) = \exp(-10t).$$

The revised answer is given since the implicit Euler method's tentative solution still has a second-order error magnitude as

$$y(t) = \hat{y}(t) + t^2 N(t, w).$$

Other than that, the neural network parameters utilized in the prior experiment are unaltered.

Numerical findings are shown in Fig. 2(a), which contrasts the analytical solution with solutions obtained using the suggested approach. Results are shown on a logarithmic scale so that the magnitude of the solution can be easily observed, since the solution decays quickly due to the existence of the stiff component. As can be seen from the picture, the results obtained using the suggested method are more in line with the analytic solution compared to the tentative implicit Euler solution. Figure 2(b) shows the residual error between the analytical solution and the recommended solution, which may be used to quantitatively evaluate the correctness. Even when stiffness is present, the findings show that the postponed correction approach based on neural networks works effectively.[25]

Nevertheless, the stiff problem's accuracy falls short of expectations, indicating room for improvement. Optimization algorithm, learning rate, network design, number of trainable parameters, and other hyperparameters are among the controllable aspects that affect the performance of neural network-based approaches. The suggested strategy for stiff ordinary differential equations is anticipated to perform even better after a thorough examination of these parameters.

Example 2

The following ordinary differential equation that is not linear is then considered:

$$\frac{dy}{dt} = (1 + t)y, \qquad (11)$$

with the initial condition

$$y(0) = 1,$$

defined over the time interval $[0, 1]$. A uniform step size $h = 0.1$is employed, resulting in 11 discrete nodal points. Each layer of the neural network uses the same 11 nodes as the previous instances.

The exact solution of Eq. (11) is given by

$$y(t) = \exp\left(t + \frac{t^2}{2}\right).$$

Following the same strategy as in Example 1, a provisional solution $\hat{y}(t)$is originally calculated by means of the explicit first-order midpoint technique. Since this method yields a second-order accurate approximation, the associated numerical error is of order $O(h^2)$. Consequently, the neural network correction term is chosen in the form $t^2 N(t, w)$.

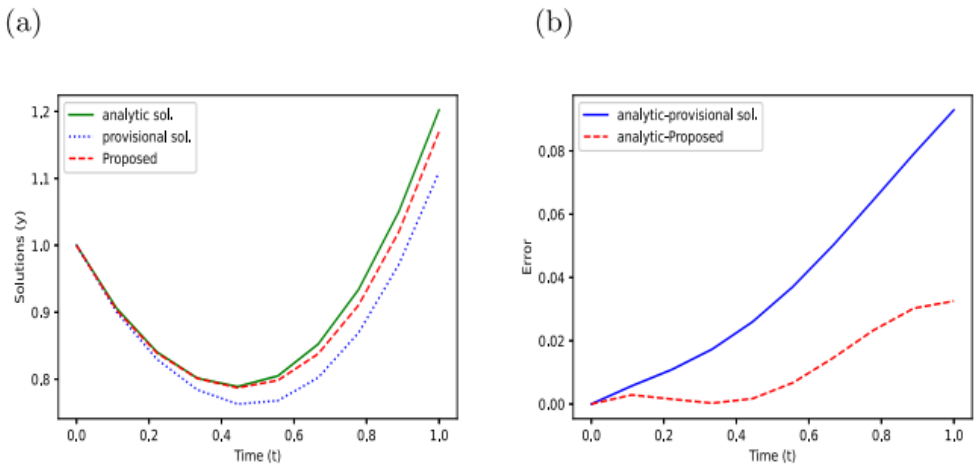

Figure 3 Comparison of (a) solution behaviors of the proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution

Figure 3(a) shows the numerical findings, which compare the suggested solution to the analytic solution and the tentative solution. The suggested strategy closely tracks the precise answer, as is readily apparent. Figure 3(b) displays the residuals from the numerical approximations compared to the analytic solution, allowing for a more thorough evaluation of the accuracy. These outcomes prove that the postponed correction technique based on neural networks that was suggested accomplishes a commendable job with this issue.[26]

We also look at how much computing power the suggested approach uses. The experiment is run 100 times with the average computing time recorded to ensure the timing findings are more reliable. The suggested method to reduce the parameters of the neural network takes around 6.2 seconds of processing time and employs 617 gradient descent rounds w.

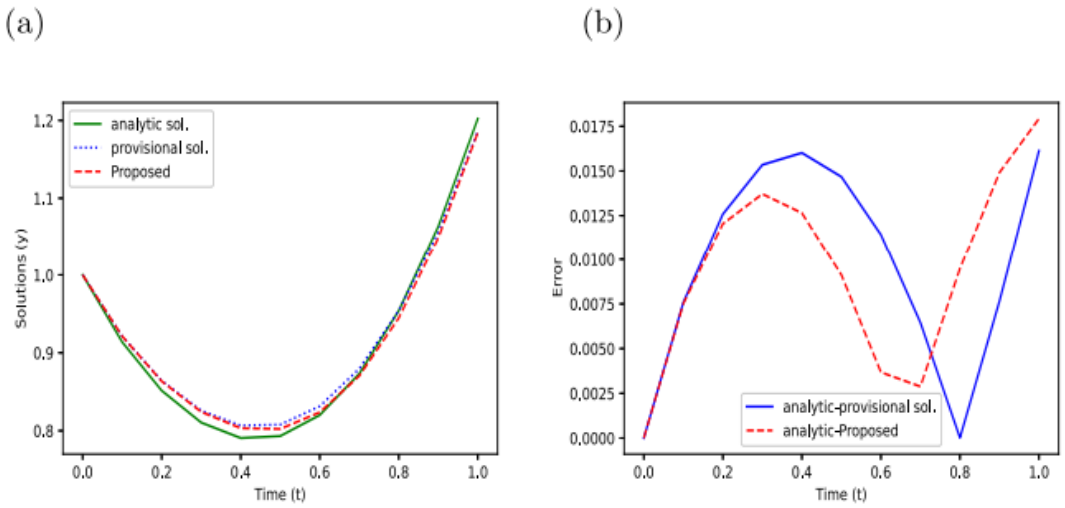

Figure 4 Comparison of (a) solution behaviors of the proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution

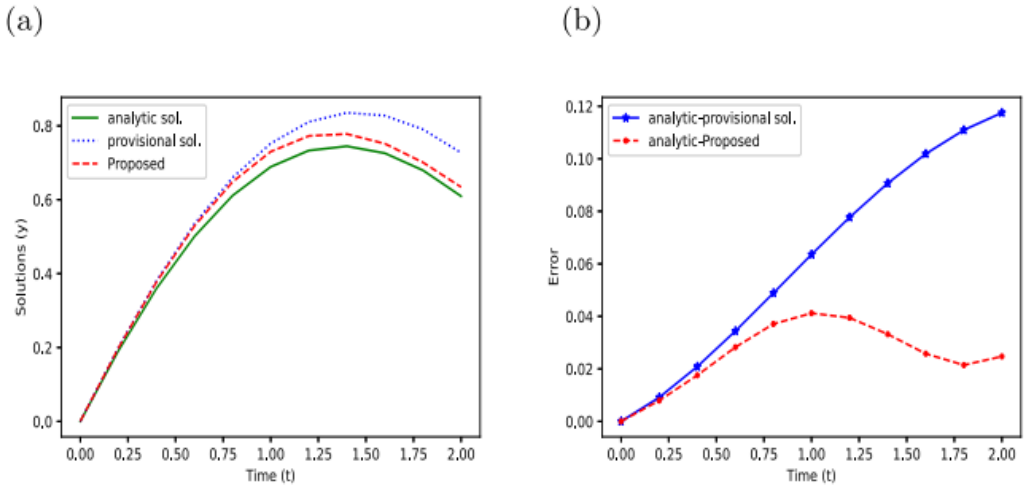

Figure 5 Comparison of (a) solution behaviors of the proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution

Example 3

In this subsection, we consider the following ordinary differential equation:

$$\frac{dy}{dt} = -\frac{1}{5}y + \cos(t), \qquad (13)$$

with the initial condition

$$y(0) = 0.$$

The exact solution of this problem is

$$y(t) = e^{-t/5}\sin(t).$$

For the numerical experiments, the time interval $[0, 2]$is evenly discretized with a set of ten nodes that are used in the neural network's input and hidden layers.

As in the previous examples, a provisional solution $\hat{y}(t)$is determined by applying the explicit first-order Euler algorithm. Given that this approach yields initially precise results, the associated numerical error is of order $O(h^2)$. Accordingly, the trial solution is constructed as

$$y(t) = \hat{y}(t) + t^2 N(t, w).$$

Figure 5(a) displays the numerical results, which were obtained using the identical neural network setup and training method as previously. These results are compared to the analytic solution and the provisional Euler solution. By carefully following the analytic solution, the suggested technique clearly produces a far better approximation.[27]

Figure 5(b) shows the results of a calculation and plot of the discrepancy between the analytical and recommended solutions, which allows for a more thorough examination of the correctness. The L^2-norm of the error between the proposed solution and the analytic solution is 0.09291, on the other hand, the tentative Euler solution has an inaccuracy of 0.24264. These outcomes validate the much-improved accuracy attained by the suggested delayed correction approach based on neural networks for this particular issue.

Furthermore, we look into the correlation between computing time and error accuracy. Researchers do tests with different iteration counts to see how the computing time changes as a function of the number of iterations utilized for reduction. There is a processing time of around 1.1 seconds and an error of 0.17968 for 100 iterations. The computing time drops to 10.5 seconds and the error to 0.02997 when the iteration count is increased to 2000. Based on these findings, it is clear that more training iterations result in better accuracy, but at the price of increased computational cost. Figure 6 shows the error behavior plotted against the iteration count to demonstrate this trend.[29]

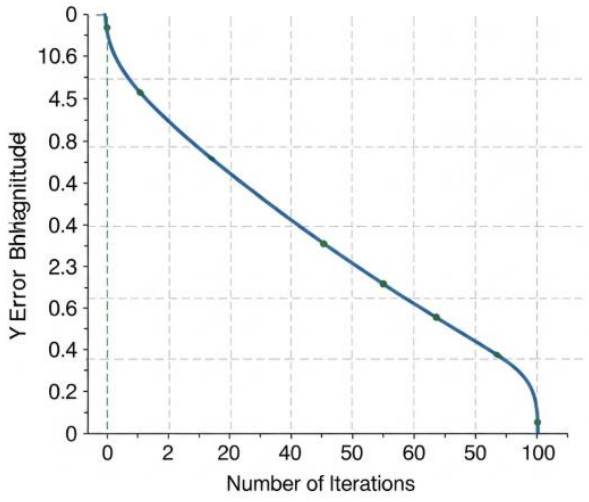

Figure 6 Error behavior according to the number of iterations in the minimization scheme

Example 4

Finally, the harmonic oscillator is taken into consideration as a basic Hamiltonian system, defined by

$$\frac{dy_1}{dt} = y_2, \qquad (14)$$
$$\frac{dy_2}{dt} = -y_1, \qquad (15)$$

with the initial condition

$$y(0) = [0,1]^T.$$

The exact solution of this system is given by

$$[y_1(t), y_2(t)] = [\sin(t), \cos(t)].$$

For the numerical simulations, the time interval $[0, 2]$ uses a uniform discretization with 10 nodes, which are used in the neural network's input and hidden layers once again.

Since the system is Hamiltonian, the provisional solution $\hat{y}(t)$is the first-order implicit Euler technique, as stated in Eq. (10), because of its advantageous stability qualities, was used for computation. Completed trial solutions as

$$y_1(t) = \hat{y}_1(t) + t^2 N_1(t, w),$$
$$y_2(t) = \hat{y}_2(t) + t^2 N_2(t, w).$$

Figure 7 displays the numerical solutions to the vector system, along with comparisons to the analytic solution and the provisional implicit Euler solution. These outcomes prove that the suggested method successfully approximates the system's two main parts.

As an added bonus, the conservation of a scalar variable called the Hamiltonian, represented as by H. For the harmonic oscillator, the Hamiltonian is defined as

$$H = y_1^2 + y_2^2. \qquad (16)$$

Theoretically, the Hamiltonian ought to be static. The suggested scheme's energy conservation property may be shown in Figure 8(a), which shows the solution's phase-space orbit, and Figure 8(b), which shows the development of the Hamiltonian. [30]

As seen in Figure 8(a), the suggested technique yields trajectories that nearly resemble the analytical circular orbit. The implicit Euler technique causes the Hamiltonian to gradually decrease due to artificial energy dissipation, as seen in Fig. 8(b). However, following an initial transient phase, the suggested neural network-based approach shows enhanced energy conservation behavior, proving that it effectively preserves the qualitative structure of the Hamiltonian system.

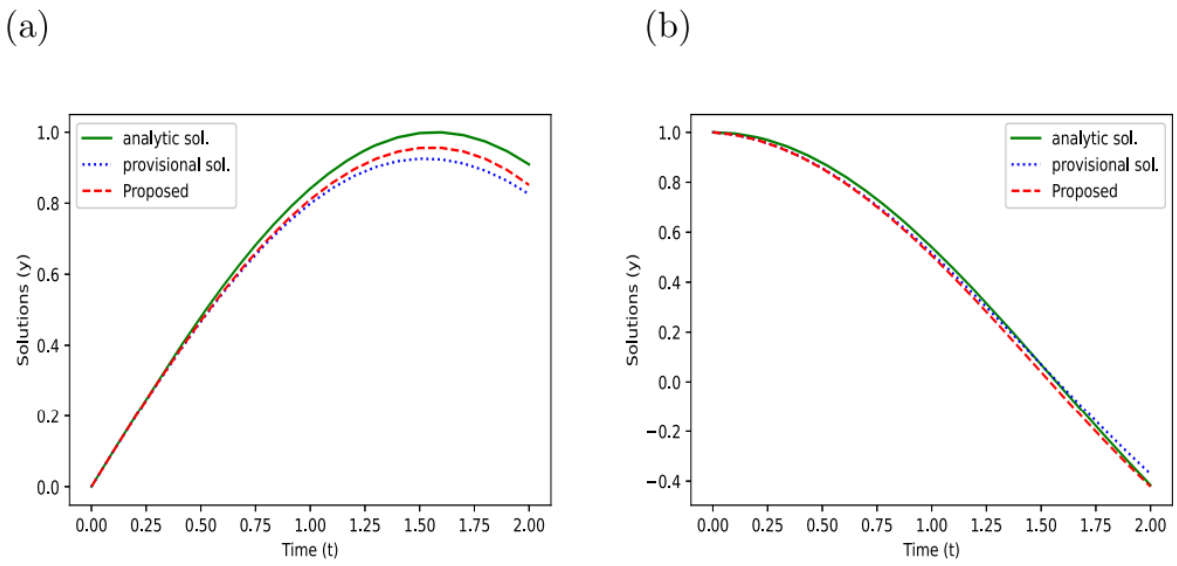

Figure 7 Comparison of solution behaviors of the proposed scheme with analytic solution and Euler method for (a) first component y1 and (b) second component y2

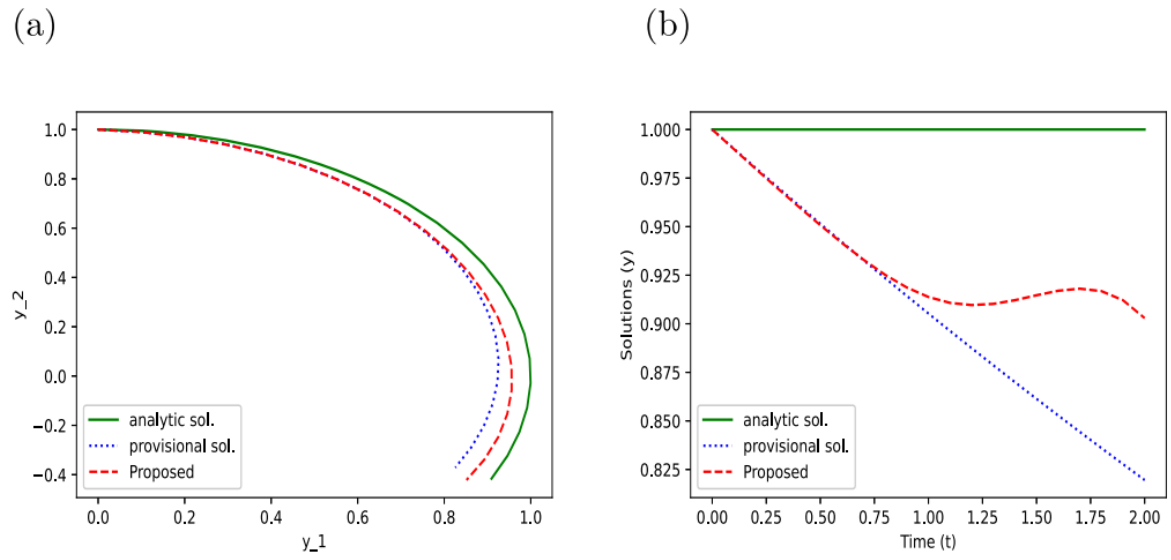

Figure 8 Comparison of (a) orbits of solutions and (b) behaviors of Hamiltonian values

DISCUSSION

We present a fresh method to the neural network approaches that are employed for the resolution of ordinary differential equations. This strategy is included within the scope of this study. Based on the preliminary answer that has been found through the utilization of lower-order numerical methods, the neural network technique that has been created makes predictions regarding the relevant mistake. The prior methods, on the other hand, directly estimate the answers. This is in contrast to those methods. Furthermore, the proposed scheme is created with the goal of predicting sufficient magnitudes of the relevant error in accordance with the convergence order of the lower-order numerical scheme that is employed for the provisional solutions.[31] This is done in order to ensure that the accuracy of the

proposed scheme is maintained. The recommended approach has the potential to attain greater levels of accuracy in contrast to ways that have already been established, as indicated by a number of numerical findings. In order to make the strategy that has been offered more successful, we need to take into consideration a number of issues that have been brought up. When it comes to neural networks, one of the first things that has to be done is to optimize the numerous parameters and optimization choices that may be controlled to achieve optimal performance. The next stage is to become familiar with the procedures for dealing with stiff problems, as seen in Example 1. Another option would be to develop a unique neural network technique for Hamiltonian systems, such as symplectic neural networks, with the purpose of preserving the energy of the Hamiltonian. In conclusion, it is essential for us to make use of higher-order temporary solutions in order to get results that are the most accurate possible.[32]

CONCLUSION

The study suggests utilizing a neural network to assess ODE solution errors. It would be superior than numerical analysis since it would be data-driven. Learning error patterns from reference solutions helps the recommended technique capture intricate behaviors induced by nonlinearity, stiffness, and step sizes. Based on numerical evaluations of several sample ODEs, the neural network–based estimator consistently outperformed existing error estimating approaches in accuracy, resilience, and flexibility. Results indicate that machine learning may complement number solvers rather than replace them. When used with Runge–Kutta procedures, the neural network improves error correction and solution reliability without slowing computing. This technique combines numerical approaches' academic merits with learning-based models' flexibility. Although the approach has benefits, it depends on the quality and representativeness of the training data, making it difficult to adapt to new scenarios. Researchers may add the estimator to adaptive solutions, make it operate with partial differential equations, and investigate physics-based deep neural network architectures. The study suggests that neural network–based error estimates may improve differential equation numerical solutions.

REFERENCES

1. Atsalakis, G. S., & Valavanis, K. P. (2009). Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Systems with Applications, 36*(7), 10696–10707. https://doi.org/10.1016/j.eswa.2009.02.043

2. Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time series analysis: Forecasting and control* (3rd ed.). Prentice Hall.

3. Bu, S. (2016). New construction of higher-order local continuous platforms for error correction methods. *Journal of Applied Analysis and Computation, 6*(2), 443–462.

4. Bu, S. (2022). A collocation method based on the quadratic quadrature technique for fractional differential equations. *AIMS Mathematics, 7*(1), 804–820. https://doi.org/10.3934/math.2022047

5. Bu, S., & Bak, S. (2020). Simulation of advection–diffusion–dispersion equations based on a composite time discretization scheme. *Advances in Difference Equations, 2020*, Article 132. https://doi.org/10.1186/s13662-020-02602-4

6. Bu, S., Huang, J., & Minion, M. L. (2012). Semi-implicit Krylov deferred correction methods for differential algebraic equations. *Mathematics of Computation, 81*(280), 2127–2157.

7. Dutt, A., Greengard, L., & Rokhlin, V. (2000). Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics, 40*(2), 241–266.

8. Gear, C. W. (1971). *Numerical initial value problems in ordinary differential equations*. Prentice Hall.

9. Graves, A., Mohamed, A., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6645–6649). IEEE. https://doi.org/10.1109/ICASSP.2013.6638947

10. Greengard, L. (1991). Spectral integration and two-point boundary value problems. *SIAM Journal on Numerical Analysis, 28*, 1071–1080.

11. Guo, X., Yang, C., & Yuan, Y. (2021). Dynamic-weighting hierarchical segmentation network for medical images. *Medical Image Analysis, 73*, Article 102165. https://doi.org/10.1016/j.media.2021.102165

12. Hairer, E., Lubich, C., & Roche, M. (1989). *The numerical solution of differential-algebraic systems by Runge–Kutta methods*. Springer.

13. Hairer, E., Nørsett, S. P., & Wanner, G. (1993). *Solving ordinary differential equations I: Nonstiff problems* (2nd ed.). Springer.

14. Huang, J., Jia, J., & Minion, M. (2006). Accelerating the convergence of spectral deferred correction methods. *Journal of Computational Physics, 214*(2), 633–656.

15. Jaitly, N., Nguyen, P., Senior, A., & Vanhoucke, V. (2012). Application of pretrained deep neural networks to large vocabulary speech recognition. In *Proceedings of Interspeech* (pp. 2578–2581).

16. Johnson, C. (1988). Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations. *SIAM Journal on Numerical Analysis, 25*(4), 908–926.

17. Kim, P., & Bu, S. (2015). Error control strategy in error correction methods. *Kyungpook Mathematical Journal, 55*, 301–311.

18. Kim, P., Piao, X., Jung, W., & Bu, S. (2018). A new approach to estimating a numerical solution in the error embedded correction framework. *Advances in Difference Equations, 2018*, Article 168. https://doi.org/10.1186/s13662-018-1612-4

19. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097–1105).

20. Kumar, M., & Yadav, N. (2011). Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey. *Computers & Mathematics with Applications, 62*, 3796–3811.

21. Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks, 9*(5), 987–1000.

22. Li, Y., Wang, G., Nie, L., Wang, Q., & Tan, W. (2018). Distance metric optimization driven convolutional neural network for age invariant face recognition. *Pattern Recognition, 75*, 51–62.

23. Lo, S. C. B., Chan, H. P., Lin, J. S., Li, H., Freedman, M. T., & Mun, S. K. (1995). Artificial convolution neural network for medical image pattern recognition. *Neural Networks, 8*(7–8), 1201–1214.

24. Mai-Duy, N., & Tran-Cong, T. (2001). Numerical solution of differential equations using multiquadric radial basis function networks. *Neural Networks, 14*, 185–199.

25. Ou, G., & Murphey, Y. L. (2007). Multi-class pattern classification using neural networks. *Pattern Recognition, 40*(1), 4–18.

26. Pelt, D. M., & Sethian, J. A. (2018). Mixed-scale dense convolutional neural networks for image analysis. *Proceedings of the National Academy of Sciences, 115*(2), 254–259.

27. Ramuhalli, P., Udpa, L., & Udpa, S. S. (2005). Finite-element neural networks for solving differential equations. *IEEE Transactions on Neural Networks, 16*, 1381–1392.

28. Shampine, L. F. (2005). Error estimation and control for ODEs. *Journal of Scientific Computing, 25*(1), 3–16.

29. Sultana, F., Sufian, A., & Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural networks: A survey. *Knowledge-Based Systems, 201–202*, Article 106062.

30. Tan, L. S., Zainuddin, Z., & Ong, P. (2018). Solving ordinary differential equations using neural networks. *AIP Conference Proceedings, 1972*, 020070. https://doi.org/10.1063/1.5041204

31. Yang, Y., Hou, M., & Luo, J. (2018). A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods. *Advances in Difference Equations, 2018*, Article 469.

32. Zheng, X., & Yang, X. (2009). Techniques for solving integral and differential equations by Legendre wavelets. *International Journal of Systems Science, 40*(11), 1127–1137.