RESEARCH

Error estimation using neural network technique for solving ordinary differential equations

Haewon Nam[1], G Bak[2] and Sunyoung Bu[1*]

[*]Correspondence: syboo@hongik.ac.kr
[1]Department of Liberal arts, Hongik University, Sejong, Republic of Korea
Full list of author information is available at the end of the article

Abstract

In this paper, we present a numerical method to solve ordinary differential equations (ODEs) by using neural network techniques in a deferred correction method framework. Similar to the deferred or error correction techniques, a provisional solution of the ODE is preferentially calculated by any lower order scheme to satisfy given initial conditions, and the corresponding error is investigated by fully connected neural networks and structured to obtain sufficient magnitude of the error. Numerical examples are illustrated to demonstrate the efficiency of the proposed scheme.

Keywords: Ordinary differential equations; Neural network; Deferred correction method; Error correction method

1 Introduction

Solving ordinary differential equations (ODEs) has been paid lots of attention of many scientists and mathematicians due to its importance in various fields of sciences and engineering. For this reason, several numerical techniques for solving ODES have been developed during last few decades. Also, the numerical methods can be broadly classified into the following categories: the first is one class of one-step multi-stage techniques such as Runge–Kutta type methods [6, 13, 17], the second is a kind of BDF type multi-step methods and the last is a group of deferred or error correction methods [4, 5, 7, 18] such as spectral deferred correction (SDC) methods [8, 9], etc.

Especially, in [16], Krylov deferred correction method (KDC), one kind of deferred correction methods, has been introduced for getting more accurate and higher order solutions of various differential equations, in which the numerical solution and the corresponding error at each integration step are calculated at the same time, so that the final algorithm can control the error and have good properties such as higher convergence order, better stability and higher accuracy, etc, compared with the existing numerical techniques.

Apart from this way, with the development of artificial intelligence and computer technology, many researchers have recently paid tremendous attentions to develop neural network techniques. Neural networks have been broadly used in many research fields such as pattern recognition [22, 23], speech recognition [10, 15], image processing [12, 26, 29], forecasting [2, 3], and classification [19, 25], etc. For this reason, lots of neural network methods are currently developed and widely used. Based on the advantages of such neural network techniques, there are some attempts to use the neural network techniques for solving various mathematical problems such as differential equations, such as multi-layer perceptron neural network [20, 30], radial basis function neural network [20, 24], finite element neural network [27] and wavelet neural network [32], etc.

Based on these developments of numerical techniques to resolve mathematical problems, in this study, we especially focus on deferred correction schemes to solve ODEs. Usually, the existing numerical schemes are searching for the solutions of given differential systems. Unlike the traditional numerical schemes, on the contrary, the deferred or error correction schemes are investigated for the numerical errors with a provisional solution which is preferentially calculated by any numerical scheme. It is already shown that these schemes can have higher convergence order and higher accuracy without any loss of stability [4, 5, 7, 18, 28].

In addition to the deferred correction schemes, we consider the neural network techniques to solve the ODEs. Recently, several researches have been attempted to solve various types of differential equations by using the neural network techniques. For example, in [21], a trial solution of differential equations with initial and boundary values is written as a sum of two parts -one is represented as a function which can manage a given initial or boundary conditions and the other one consists of a feedforward neural networks which is independent of the initial and boundary conditions. In [31], a Legendre neural networks method for ODEs are presented by representing a trial solution by Legendre network, in which a Legendre polynomial is chosen as a basis function of hidden neurons and a single hidden layer Legendre neural network is used to eliminate the hidden layer by expanding the input pattern using Legendre polynomials. Here, an improved extreme learning machine algorithm was used for training network weights.

The main objective of this paper is to develop a new algorithm to solve ODEs by using neural network techniques to estimate the numerical error with a calculated provisional solution. First of all, we begin with usage of the lower order numerical scheme such as the first order Euler method or the second order midpoint method for the provisional solution. Note that for getting much higher accuracy, we may cast any elaborate higher order numerical scheme but it may cause enormous computational costs for estimating the provisional solutions. In that neural network techniques also require a certain amount of computational costs, the usage of the higher order methods is meaningless in the proposed algorithm. Once the provisional solution is obtained, the corresponding error is estimated by a full connected neural network. In particular, we newly set up the corresponding error according to the convergence order of the numerical schemes for the provisional solutions to obtain sufficient magnitudes of the corresponding error. For an assessment of the effectiveness of the proposed algorithm, several experiments are simulated, and especially, a harmonic oscillator problem is solved to examine the effectiveness of the Hamiltonian property. Throughout these numerical tests, we show that the proposed method works very well and has good properties.

This paper is organized as follows. It starts with some explanations of the basic knowledge that composed of the proposed scheme in Sec. 2.1. In Sec. 2.2, we present the proposed scheme by using neural network systems, in which a provisional solutions of the given system is roughly calculated by lower order numerical scheme, after that the corresponding error is estimated by traditional fully neural network techniques. In Sec. 3, several numerical results are presented to examine the effectiveness and efficiency of the proposed scheme. Finally in Sec. 4, we summarize our results and discuss several possibilities to increase the efficiency of the proposed scheme.

2 Methods

2.1 Preliminaries

In this subsection, we briefly explain the basic backgrounds to require for the proposed scheme to solve a general ODE system described by

$$y'(t) = F(t, y(t)), \tag{1}$$

with the initial conditions $y(0) = y_0$ in a given time interval $[t_0, t_f]$. Here, t_0 is the initial time and t_f is the final time. Also, we assume that the solutions of the given problem Eq. (1) are continuous.

Usually in traditional numerical methods, for getting numerical solutions of the problem Eq. (1), the given time interval is discretized into several sub-intervals. With the initial conditions, the solution in the first sub-interval is numerically calculated and the solution will be an initial condition of the next sub-interval. This process is sequentially continued and the final solutions can eventually be obtained at the final time t_f.

On the other hand, unlike the traditional numerical schemes to solve ODEs, the neural network schemes for solving ODEs have different structures. Most traditional numerical schemes usually have a sequential process to march from an initial time to a final time point, whereas neural network schemes simultaneously seek solutions at all time points. The solution used in neural network techniques can be represented as

$$y(t) = A(t) + G(t, N(t, w)), \quad t \in [t_0, t_f] \tag{2}$$

where t_0 is an initial time point, t_f is the final time point and $N(t, w)$ is a feedforward neural network with parameters w and an input vector t. The first term $A(t)$ usually represents given initial or boundary conditions. The second term $G(t, N(t, w))$ is constructed so as not to contribute to the initial or boundary conditions, since it must satisfy them in the first part. This term employs a neural network whose weights w are to be adjusted in order to deal with the minimization problem. Note that the problem has been reduced from the original constrained optimization problem to an unconstrained one due to the form of the trial solution that satisfies by construction of the initial or boundary conditions. Once the numerical solution y is set up, a cost function $G(w)$ with the weights w of the neural network is defined as

$$G(w) = |y' - F(t, y)|, \tag{3}$$

where y is defined in Eq. (2) and F is defined in Eq. (1).

Based on the cost function defined above, the weights w should be solved by one of various optimization techniques. The most basic technique is the gradient descent algorithm which is an iterative minimization technique for finding a local minimum of the given cost function. The algorithm has the following two steps and processes repeatedly:

- The gradient is calculated by the first order derivative of the cost function $G(w)$ at a point.
- Move backward in the opposite direction of the gradient

$$w_{i+1} = w_i - \gamma \frac{dG(w)}{dw}. \tag{4}$$

Note that to find the local minimum of a function based on gradient descent, we must take steps proportional to the negative of the gradient (move away from the gradient) of the cost function at the current point. Also, the γ is a learning rate which is a tuning parameter in the minimization process and decides the length of the steps. It means that if the learning rate is too high, we might overshoot local minimum with keeping bouncing, without reaching the desired minimum, whereas if the learning rate is too small, the training takes too much time, so the computational cost goes too high.

However, most cost functions usually consist of several local minimum points. The gradient may reach to any one of the minimum points, which depends on both the initial point and the learning rate. Due to this reason, the gradient descent optimization technique may converge to different points whenever executing with different initial points and learning rate, which is a weakness of the gradient descent technique.

2.2 Method Description

The main objective of this subsection is to introduce the propose scheme using the neural network based on deferred correction frameworks. Note that the neural network used in this work is a simple full connected neural network in order to exclude the efficiency or reliability of neural networks and focus only on the effectiveness of the proposed method.

Basically, we focus on the calculation of the numerical error unlike the traditional numerical methods which directly estimate solutions of given equations. That is, instead of solving for $y(t)$ in Eq. (1), a provisional solution $\hat{y}(t)$ is firstly obtained by using any lower order method or initial conditions and then the corresponding error $E(t)$ is defined by $y(t) - \hat{y}(t)$. In a similar way to deferred or error correction techniques, $E(t)$ can be estimated by any neural network algorithm.

Here, we simply try to use the 1st order numerical scheme as a provisional solution, such as Euler method. Remind that at i-th time point, Euler method can be described as

$$\hat{y}(t_i) = \hat{y}(t_{i-1}) + h_i f(t_{i-1}, \hat{y}(t_{i-1})), \tag{5}$$

where $h_i = t_i - t_{i-1}$ and $\hat{y}(t_0) = y_0$. Based on the provisional solution $\hat{y}$ calculated above, we cast a neural network technique to estimate the error function $E(t) = y(t) - \hat{y}(t)$. As explained in Eq. (2), the estimated solution $y(t)$ can be represented as

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

where $G(.)$ is an appropriate function with respect to t and $N(t, w)$ for estimating the corresponding error term and $N(t, w)$ is a single-output feedforward neural network with parameters w and n-input units fed with the input time vector t. The first term $\hat{y}(t)$ is a provisional solution calculated in Eq. (5). Since the $\hat{y}(t)$ is the first order solution, the error term $G(t, N(t, w))$ should have the second order magnitude.

On the other hand, near the initial point $t = 0$, the Taylor expansion of the $y(t)$ can be represented as follows:

$$\begin{aligned} y(t) &= y(0) + ty'(0) + \frac{t^2}{2} y''(0) + \cdots && (7) \\ &= y_0 + tf(0, y_0) + \frac{t^2}{2} y''(0) + \cdots, && (8) \end{aligned}$$

where $t \in [0, 1)$. By the first order Euler scheme, the first two terms in Eq. (8) can be estimated, so error should contain from the t^2 term. Therefore, the error function $G(t, N(t, w))$ in Eq. (6) composes of t^2 term and beyond. Due to $t \in [0, 1)$, t^2 term is dominant, so we can define $G(t, N(t, w))$ as $t^2 N(t, w)$. Eventually, the final form of the desired solution in Eq. (6) can be summarized as

$$y(t) = \hat{y}(t) + t^2 N(t, w), \quad t \in [0, 1), \tag{9}$$

where $\hat{y}(t)$ is the first order estimation.

Based on the whole discussion above and backgrounds, we get the following neural network algorithm to solve ODE :

Data: Discrete time points in a given time interval $[0, 1)$
Input: the desired number of layers, the desired order of convergence (p)
Result: Solutions of the given ODE (Eq.(1)) over desired time points or time intervals
Initialize basic parameters for neural network (learning rate γ, weight w's, tolerance tol)
Perform a lower order scheme to calculate a provisional solution $\hat{y}(t)$ with the $(p-1)$th order method or an initial condition
Define the error $E(t) = y(t) - \hat{y}(t)$
Go to neural network $N(t, w)$ to estimate $E(t)$ such that $E(t) = t^p N(t, w)$ with setting up $cost = ||f(t, y) - y'(t)||$
while $cost > tol$ **do**
    Perform neural network $N(t, w)$ with an appropriate sigmoidal function $\sigma(t)$ and basic gradient descent scheme
    Get new w's to be local minimum of $cost$
    Reset $cost$ with the new w's
end
Get the finalized solution $y(t) = \hat{y}(t) + t^p N(t, w)$

Algorithm 1: Proposed Algorithms

3 Experiments

In this section, we test several examples to examine the effectiveness of the proposed scheme and compare the results with own exact solutions. Note that there are several minimization algorithms used in the neural network techniques. As mentioned above, we concentrate only on the efficiency of the proposed algorithm for solving ODE systems, without any aid of effects caused by other techniques such as the choice of neural networks, or minimization schemes, etc. Therefore, for these experiments, we simply use the basic gradient descent method as a minimization tool with the fixed learning rate $\gamma = 0.001$. Details of each problem will be explained in each subsection.

All numerical results are obtained using Python 2.1.5, on a computer with 11th Gen INTEL core I7-1165G7 CPU, 16 GB memory, and WIN10 operating system. All computational codes including neural networks and minimization schemes are implemented in Python by ourselves without any usage of libraries or packages. Also, since a simple neural network is used for this work, only 3 layers are used.

3.1 Example 1

For the first example, we test the simplest form of the ODEs described by

$$y' = \lambda y, \qquad \lambda < 0, \tag{10}$$

with the initial condition $y(0) = 1$ and the time interval is $[0, 1]$ and uniformly discretized into 10 sub-intervals. That is, 11 node points are used in the input and middle layers of the neural network. The analytic solution is $y(t) = \exp(\lambda t)$. Note that to sustain the numerical stability, λ of the example (10) should be negative $[1, 11, 14]$. For this test, we simply set up $\lambda = -1$.

Since a provisional solution $\hat{y}$ is estimated by the first order explicit Euler method, the corresponding error has the second order $O(h^2)$ magnitude, so the neural network term can be set up to $t^2N(t, p)$. The network was trained on the 10 sub-interval points in $[0, 1]$. All numerical results are plotted in Fig. 1(a) and compared with the analytic solution and the provisional solution. It can be seen that the proposed scheme is closer to the analytic solution so we can conclude that the proposed scheme produces quite reasonable results.

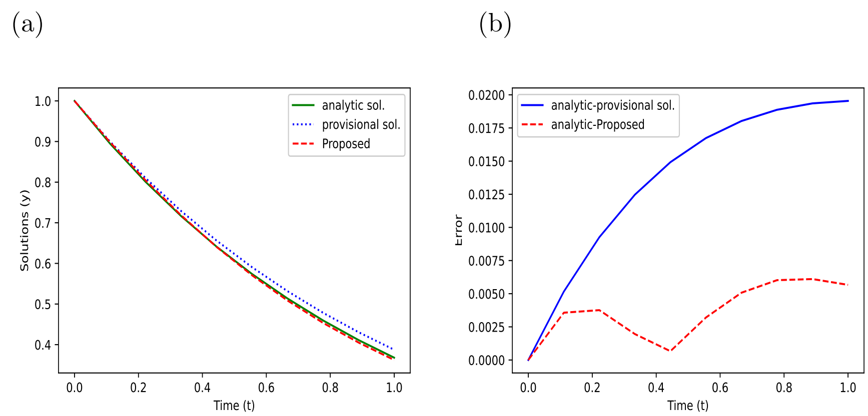

Figure 1 Comparing (a)solution behaviors of proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution

For further examination of the accuracy, we check the difference between the analytic solution and the proposed one and plot it in Fig. 1(b). To precisely compare the difference with the provisional solution, $L2$-norm is measured. The residual between the analytic solution and proposed scheme and between the analytic one and the provisional solution are 0.05277 and 0.15504, respectively. Summing up these results, we can easily see that the proposed scheme works well and has a good accuracy for this problem.

Additionally, to investigate the effect of the stiffness in neural network techniques, we apply the propose scheme to a problem with having stiff component, so λ is set up -10 in problem (10) to be mildly stiff. Since it becomes stiff, the corresponding provisional solution should be obtained from an implicit method as long as the same step size is persisted to be used in the non-stiff case. Hence, the provisional solution at i-th time integration point can be obtained by the first order implicit Euler method described by

$$\hat{y}(t_i) = \hat{y}(t_{i-1}) + hf(t_i, \hat{y}(t_i)), \tag{11}$$

where h is a step size and $\hat{y}(t_0) = 100$. The analytic solution is represented as $y(t) = \exp(-10t)$. With the provisional solution having the second order magnitude, the solution is taken as $y(t) = \hat{y}(t) + t^2N(t, p)$. Other conditions for the neural network are the same as above. Fig. 2(a) displays the solution behaviors of the proposed schemes and comparisons with the analytic solution. Note that due to stiff component, the solution behavior is changed rapidly, so the figure is plotted in log-scale to observe the magnitude of the solution. It can show that the results from the proposed scheme is closer to the analytic solution. To precisely check the residual, we plot the error between the results from the proposed scheme and the analytic solution in Fig. 2(b). It shows the proposed scheme works well even for the stiff problem. However, the results are not perfectly satisfied and we need to consider other possibilities to improve the results for stiff problems. Actually, there are lots of components to control in the neural network techniques - such as several choices of the minimization technique, free parameters in each minimization, and the number of the free parameters, etc.

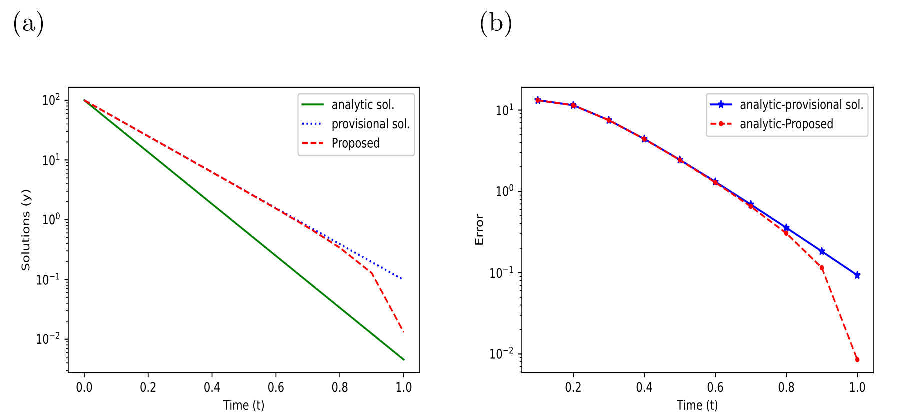

Figure 2 Comparing (a) solution behaviors of proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution for stiff case ($\lambda = -10$)

3.2 Example 2

Next, we consider the following ODE described by

$$y' + \left(t + \frac{1+3t^2}{1+t+t^3}\right) y = t^3 + 2t + \frac{t^2(1+3t^2)}{1+t+t^3}, \tag{12}$$

with the initial condition $y(0) = 1$ and a time interval is $[0, 1]$ with an uniform step size $h = 0.1$. Similarly to the previous example, 11 node point are used in each layer. The exact solutions is $y(t) = \dfrac{\exp(-t^2/2)}{1+t+t^3} + t^2$.

Similarly above, a provisional solution $\hat{y}$ is estimated by the first order explicit midpoint method, so the corresponding error has the second order $O(h^2)$ magnitude and the error is set up to $t^2N(t,p)$. We plot the results in Fig. 3(a) and compare them with the analytic solution and the provisional solution. We easily check that the proposed scheme is quite closer to the analytic solution. To magnify the residual between the analytic solution and the results above, we plot each residual in Fig. 3(b). It can be concluded that the proposed scheme works well and have a good accuracy for this problem.

Next, to investigate the effect of the convergence magnitude in the proposed neural network technique, we try to use higher order provisional solutions. Firstly, a provisional solution $\hat{y}$ is estimated by the second order explicit midpoint method as

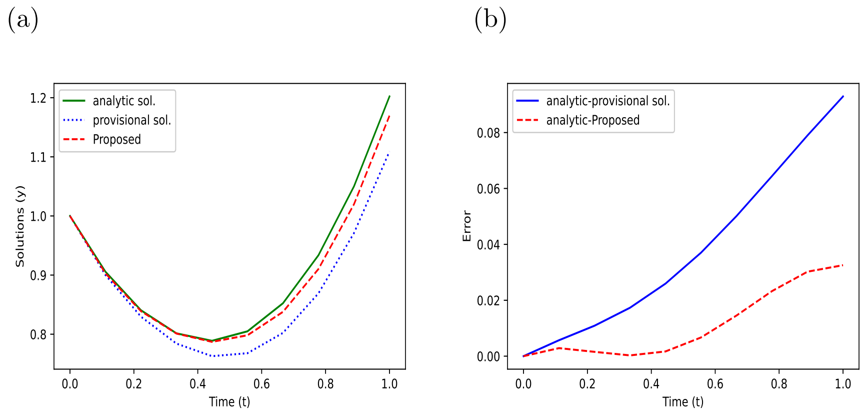

Figure 3 Comparing (a) solution behaviors of proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution

described by

$$\hat{y}(t_i) = \hat{y}(t_{i-1}) + hf(t_{i-1} + \frac{h}{2}, \hat{y}(t_{i-1}) + \frac{h}{2} f(t_{i-1}, \hat{y}(t_{i-1})), \tag{13}$$

where h is a step size and $\hat{y}(t_0) = y_0$. Therefore, the corresponding error is $O(h^3)$ so the neural solution can be defined as $t^3 N(t, p)$. The results concerning the accuracy at grid points are presented in Fig. 4(a) with comparisons of the analytic solution and the provisional solution. It can be seen that the proposed scheme is closer to the analytic solution.

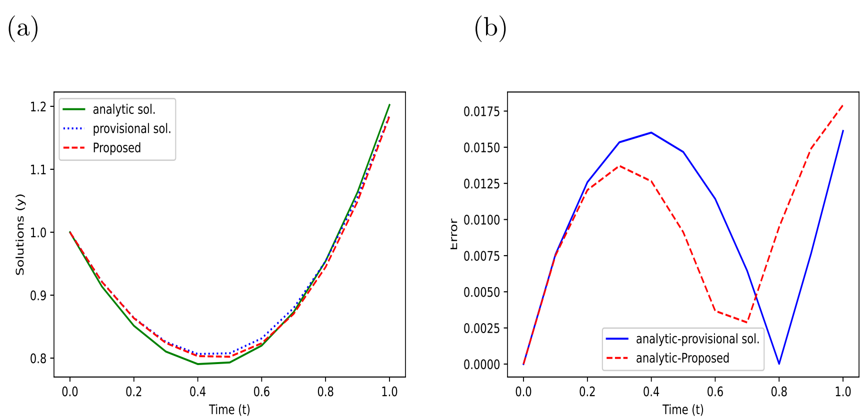

Figure 4 Comparing (a) solution behaviors of proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution

Since the higher order solution is more identical to the analytic solution, we plot the residual between the analytic solution and the proposed one in Fig. 4(b) to inspect the detailed accuracy of the proposed scheme. One can see that the proposed scheme has broadly quite smaller error compared with the second convergence order method, although there is few parts to have larger error.

3.3 Example 3

In this subsection, the following differential equation is considered :

$$y' + \frac{1}{5}y = \exp\left(-\frac{1}{5}t\right)\cos(t), \tag{14}$$

with the initial condition $y(0) = 0$. The exact solutions is $y(t) = \exp(-t/5)\sin(t)$. For the experiments, we uniformly use 10 node points in $[0, 2]$ in input and middle layers.

As done above, we estimate a provisional solution $\hat{y}$ by the first order explicit Euler method, so the corresponding error can be $O(h^2)$. The trial solution is formed to $y(t) = \hat{y} + t^2 N(t, p)$. With the same setting for the neural network scheme, we generate all numerical results and plot them in Fig. 5(a). Also the results are compared with the analytic solution and the provisional solution. It can be seen that the proposed scheme is closer to the analytic solution. To take a closer look

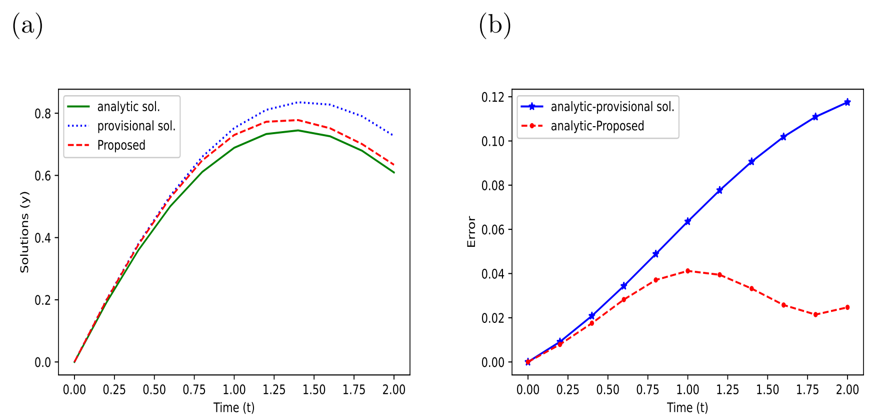

Figure 5 Comparing (a) solution behaviors of proposed scheme with analytic solution and Euler method and (b) residuals between current results and analytic solution

at the accuracy, we calculate the difference between the analytic solution and the proposed one and plot it in Fig. 5(b). The $L2$-norm of the difference between the results from the proposed scheme and the analytic solution is 0.09291, whereas the $L2$-norm of the comparison result is 0.24264. Therefore, as seen in the previous examples, we can easily conclude that the proposed scheme have a good accuracy for this problem.

3.4 Example 4

As the last example, we consider the simple Hamiltonian system known as a harmonic oscillator:

$$y_1' = y_2 \tag{15}$$

$$y_2' = -y_1 \tag{16}$$

with the initial condition $y(0) = [0, 1]$. The exact solutions is $[y_1(t), y_2(t)] = [\sin t, \cos t]$. For the experiments, we uniformly use 10 node points in $[0, 2]$ in input and middle layers.

Since the given system is Hamiltonian, a provisional solution $\hat{y}$ is estimated by the first order implicit Euler method described in Eq. (11), due to its stability. The trial solutions are set to $y_1(t) = \hat{y_1} + t^2 N_1(t,p)$ and $y_2(t) = \hat{y_2} + t^2 N_2(t,p)$. We simulate this vector system and generate numerical result as seen in Fig. 6. The results are compared with the analytic solution and the provisional solution. It can be seen that both numerical solutions of the system have a quite good accuracy.

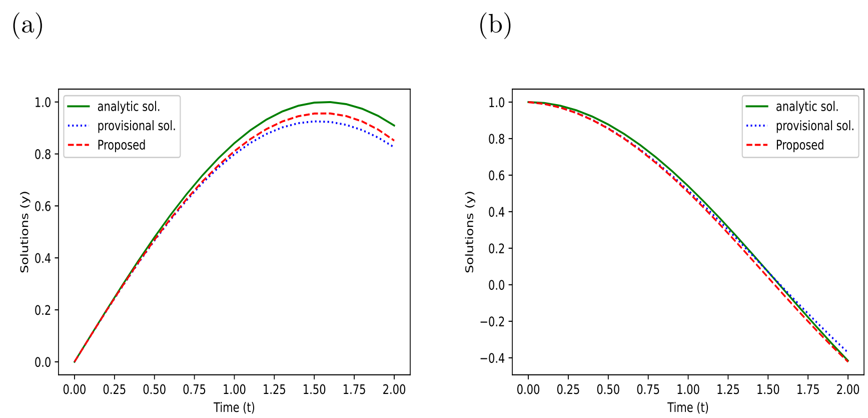

Figure 6 Comparing solution behaviors of proposed scheme with analytic solution and Euler method for (a) first component y_1 and (b) second component y_2

Additionally, a Hamiltonian system is a dynamical system described by the scalar function H, called the Hamiltonian. In this problem, the Hamiltonian H can be defined as

$$H = y_1^2 + y_2^2, \tag{17}$$

and the value of H should be conserved. To examine the conservation property, we plot an orbit of solutions in Fig. 7 (a) and the Hamiltonian H in Fig. 7 (b). Fig. 7 (a) shows that the proposed scheme is closer to the analytic solution. Also, one can verify that in Fig. 7 (b), the implicit Euler method reduces the total energy of H, whereas results obtained from the proposed scheme try to conserve the energy after a certain moment.

4 Discussion

In this paper, we introduce a new variation of the neural network techniques to solve ordinary differential equations. Unlike the traditional techniques which directly estimate solutions, the proposed neural network scheme estimates the corresponding error based on the calculated provisional solution by lower order numerical schemes. Also, the proposed scheme is designed with consideration to estimate sufficient magnitudes of the corresponding error according to the convergence order of lower order numerical scheme used for the provisional solutions. Several numerical results show that the proposed scheme can get better accuracy, compared with existing techniques.

In order to improve the efficiency of the proposed scheme, we should consider several issues. The first is to optimize the several parameters and choices of optimization which can be controlled in Neural networks. The second is to investigate

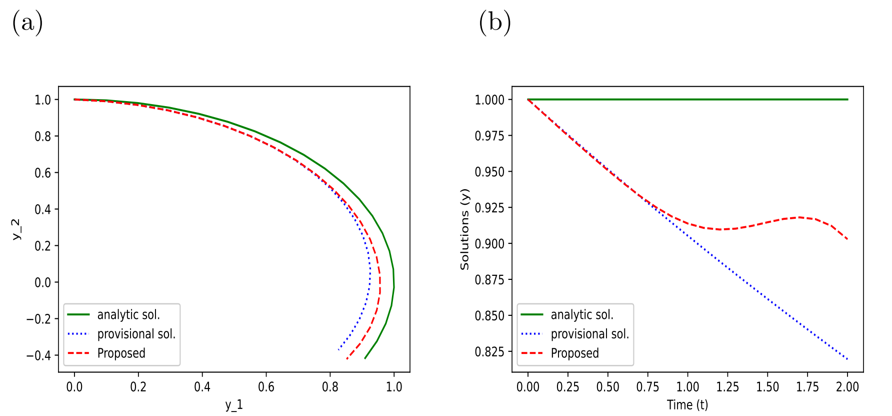

Figure 7 Comparing (a) orbits of solutions and (b) behaviors of Hamiltonian values

strategies for stiff problems as seen in Example 1 or to design a new neural network algorithm for Hamiltonian systems, such as symplectic neural networks to keep the energy of Hamiltonian. Lastly, for more accurate results, we need to employ higher order solutions as provisional solutions. Results along these directions will be reported soon.

Funding

The first author Nam was supported by basic science research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number NRF-2019R1I1A3A01059010) and the second author Baek and the corresponding author Bu were supported by basic science research program through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT) (grant number NRF-2022R1A2C1004588)

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Author's contributions

The author Bu provided several ideas to accomplish this manuscript and simulated several numerical experiments. and the author Nam provided the basic idea of this work and wrote the manuscripts. All authors read and approved the final manuscript.

Acknowledgements

Author details

[1]Department of Liberal arts, Hongik University, Sejong, Republic of Korea. [2]Department of Liberal arts, Hongik University, Sejong, Republic of Korea.

References

1. K. E. Atkinson, An introduction to numerical analysis, John Wiley & Sons, Inc., 1989.
2. G. S.Atsalakis and K P.Valavanis, Forecasting stock market short-term trends using a neuro-fuzzy based methodology,Expert Systems with Applications. 36(7), 10696–10707(2009).
3. G. Box, G. Jenkins, and G. Reinsel, Time series analysis: forecasting and control. Prentice Hall, 1994.
4. S. Bu, J. Huang and M. L. Minion, Semi-implicit Krylov deferred correction methods for differential algebraic equations, Math. Comput., 81(280), 2127–2157(2012).
5. S. Bu, New construction of higher-order local continuous platforms for Error Correction Methods, J. Appl. Anal. Compu., 6(2), 443–462 (2016).
6. S. Bu, W. Jung, and P. Kim, An error embedded Runge-Kutta method for initial value problems, Kyungpook Math. J., 56(2), 311–327(2016).
7. P. Kim, X. Piao, W. Jung and S. Bu, A new approach to estimating a numerical solution in the error embedded correction framework, Adv. Diff. Eqn., 2018:168, 1687–1847(2018).
8. A. Dutt, L. Greengard, and V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, BIT 40(2) 241-266 (2000).
9. L. Greengard, Spectral Integration and Two-Point Boundary Value Problems, SIAM J. Num. Anal., 28, 1071-1080 (1991).

10. A. Graves, A. Mohamed, and G. Hinton, Speech recognition with deep recurrent neural networks, Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pp. 6645–6649. IEEE, 2013.
11. C. W. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, 1971.
12. X. Guo, C. Yang, Y. Yuan, Dynamic-weighting hierarchical segmentation network for medical images, Medical Image Analysis, 73, (2021).
13. E. Hairer, C. Lubich, and M. Roche, The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods, Springer-Verlag, 1989.
14. E. Hairer, S. P. Norsett and G. Wanner, Solving ordinary differential equations. I nonstiff, Springer Series in Computational Mathematics, Springer, 1993.
15. N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke ,Application of pretrained deep neural networks to large vocabulary speech recognition. Interspeech, 2012.
16. J. Huang, J. Jia, and M. Minion, Accelerating the Convergence of Spectral Deferred Correction Methods, J. of Comp. Physics, 214(2), 633–656(2006).
17. C. Johnson, Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations, SIAM J. Numer. Anal., 25(4), 908–926(1988).
18. P. Kim and S. Bu, Error Control Strategy in Error Correction Methods, Kyungpook Math. J., 55, 301–311(2015).
19. A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, In Advances in neural information processing systems, 1097-–1105 (2012).
20. M. Kumar, N. Yadav, Multilayer perceptrons and radial bsis function neural network methods for the solution of differential equations:A survey, Computers and Math with Appl., 62, 3796–3811(2011).
21. I.E. Lagaris; A. Likas; D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, 9(5) , 987 – 1000(1998).
22. Y. Li, G. Wang, L. Nie, Q. Wang, and W. Tan, Distance metric optimization driven convolutional neural network for age invariant face recognition, Pattern Recognition, 75, 51–62(2018).
23. S. C. B. Lo, H. P. Chan, J. S. Lin, H. Li, M. T. Freedman, and S. K. Mun, Artificial convolution neural network for medical image pattern recognition, Neural Networks, 8(7-8), 1201–1214(1995).
24. N. Mai-Duy, T. Tran-Cong, Numerical solution of differential equations using multiquadric radial basis function networks, Neural Networks, 14, 185-–199(2001).
25. G. Ou, and Y. L. Murphey, Multi-class pattern classification using neural networks, Pattern Recognition, 40(1), 4–18 (2007)
26. D. M. Pelt, J. A. Sethian, Mixed-scale dense network for image analysis, PNAS, 115(2), 254–259 (2018).
27. P. Ramuhalli, L. Udpa and S. S. Udpa, Finite-element neural networks for solving differential equations, in IEEE Transactions on Neural Networks, 16, 1381–1392 (2005).
28. L. F. Shampine, Error estimation and control for ODEs, J. Sci. Comput., 25(1), 3–16 (2005).
29. F. Sultana, A. Sufian, and P. Dutta, Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey, Knowledge-Based Systems, 201–202, (2020).
30. L. S. Tan, Z. Zainuddin, and P. Ong, Solving ordinary differential equations using neural networks, AIP Conference Proc., 1972, 020070(2018).
31. Y. Yang, M. Hou and J. Luo, A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods, Adv. Diff. Eqn., 2018:469, (2018).
32. X. Zheng, X. Yang, Techniques for solving integral and differential equations by Legendre wavelets, International Journal of Systems Science, 40(11), 1127–1137 (2009).