Advances in Continuous and Discrete Models

RESEARCH Open Access

Error estimation using neural network technique for solving ordinary differential equations

Haewon Nam[1], Kyung Ryeol Baek[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 by 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 class consists of one-step multistage techniques such as Runge–Kutta-type methods [5, 13, 17], the second includes BDF-type multistep methods [6], and the last is a group of deferred or error correction methods [4, 7, 18, 19] such as spectral deferred correction (SDC) methods [8, 11], etc.

In particular, in [15], Krylov deferred correction method (KDC), one 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 attention to develop neural network techniques. Neural networks have been broadly used in many research fields such as pattern recognition [23, 24], speech recognition [10, 16], image processing [12, 27, 30],

forecasting [2, 3], classification [20, 26], 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. Researchers use multilayer perceptron neural network [21, 31], radial basis function neural network [21, 25], finite element neural network [28], wavelet neural network [33], 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, 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, 7, 18, 19, 29].

In addition to the deferred correction schemes, we consider the neural network techniques to solve the ODEs. Recently, several researchers have attempted to solve various differential equations by using the neural network techniques. For example, in [22], 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 consists of a feedforward neural network which is independent of the initial and boundary conditions. In [32], a Legendre neural network method for ODEs is 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 by using a 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 employ any elaborate higher-order numerical scheme but it may cause enormous computational costs for estimating the provisional solutions. Since neural network techniques also require a certain amount of computational costs, the usage of 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 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 lead to the proposed scheme in Sect. 2.1. In Sect. 2.2, we present our scheme by using neural network systems, in which a provisional solution of the given system is roughly calculated by a lower-order numerical scheme, after that the corresponding error is estimated by traditional fully neural network techniques. In Sect. 3, several numerical

results are presented to examine the effectiveness and efficiency of the proposed scheme. Finally, in Sect. 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 basics needed for the proposed scheme to solve a general ODE system described by

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

with the initial condition $y(0) = y_0$ 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 in Eq. (1) are continuous.

Usually, in traditional numerical methods, for getting numerical solutions of the problem in Eq. (1), the given time interval is discretized into several subintervals. With the initial conditions, the solution in the first subinterval is numerically calculated and that solution will be an initial condition of the next subinterval. This process is sequentially continued, and the final solution 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 structure. 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

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

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

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

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 found 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.

- The algorithm moves in the opposite direction of the gradient

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

Note that to find a local minimum of a function based on the gradient descent, we must take steps proportional to the negative of the gradient (move away from the previous point) of the cost function at the current point. Also, 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 a local minimum and keep bouncing, without reaching the desired minimum, whereas if the learning rate is too small, the training takes too much time, so the computational cost gets too large.

However, most cost functions usually contain several local minima. The gradient may reach any such minimum, which depends on both the initial point and 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 framework. Note that the neural network used in this work is a simple fully 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 first 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 first-order numerical scheme as a provisional solution, such as Euler method. Recall that at the ith time point, Euler method can be described as

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

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

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

where $G(\cdot, \cdot)$ is an appropriate function of 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 $\hat{y}(t)$ is the first-order solution, the error term $G(t, N(t, w))$ should have second-order magnitude.

Algorithm 1: Proposed algorithm

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 the neural network (learning rate γ, weights w, 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 $E(t) = t^p N(t, w)$ setting $cost = \|f(t, y) - y'(t)\|$
while $cost > tol$ **do**
  Utilize neural network $N(t, w)$ with an appropriate sigmoidal function $\sigma(t)$ and basic gradient descent scheme
  Get new ws giving a local minimum of $cost$
  Reset $cost$ with the new ws
end
Get the final solution $y(t) = \hat{y}(t) + t^p N(t, w)$

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

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

where $t \in [0, 1)$. By the first-order Euler scheme, the first two terms in Eq. (7) can be estimated, so the error should contain terms from the t^2 term. Therefore, the error function $G(t, N(t, w))$ in Eq. (6) comprises the t^2 term and beyond. As $t \in [0, 1)$, the 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

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

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

Based on the whole discussion above and background, we get neural network algorithm to solve ODEs (see Algorithm 1).

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 $\gamma = 0.001$. The 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 ODE described by

$$y' = \lambda y, \quad \lambda < 0, \tag{9}$$

with the initial condition $y(0) = 1$. The time interval is $[0, 1]$, which is uniformly discretized into 10 subintervals. 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 in (9) should be negative [1, 9, 14]. For this test, we simply set $\lambda = -1$.

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 be $t^2 N(t, w)$. The network was trained on the 10 subinterval points in $[0, 1]$. All numerical results are plotted in Fig. 1(a) and compared with the analytic and provisional solutions. 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.

For further examination of the accuracy, we check the difference between the analytic and proposed solutions and plot it in Fig. 1(b). To precisely compare the difference with the provisional solution, the L_2-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.

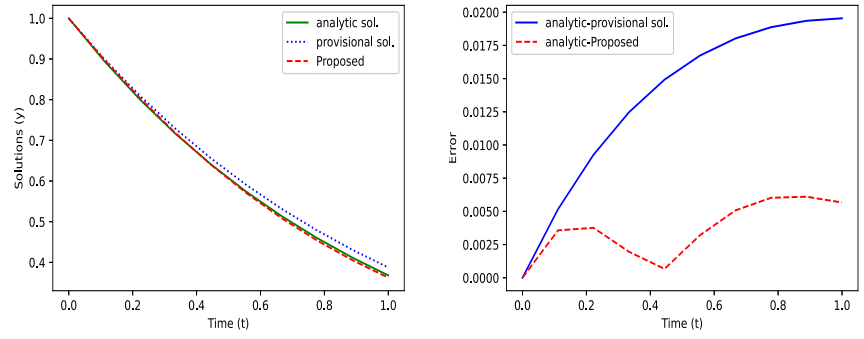

Figure 1 Comparison of (**a**) solution behaviors of the proposed scheme with analytic solution and Euler method and (**b**) residuals between current results and analytic solution

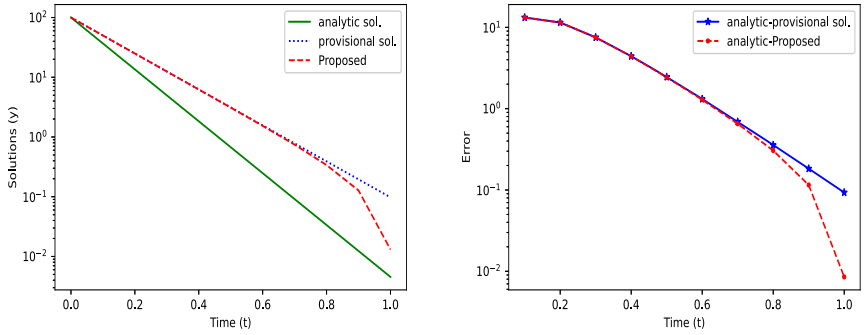

Figure 2 Comparison of (**a**) solution behaviors of the proposed scheme with analytic solution and Euler method and (**b**) residuals between current results and analytic solution for a stiff case ($\lambda = -10$)

Additionally, to investigate the effect of the stiffness in neural network techniques, we apply the propose scheme to a problem with a stiff component, so λ is set to -10 in problem (9) and is required 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 persistently used in the nonstiff case. Hence, the provisional solution at the ith time integration point can be obtained by the first-order implicit Euler method described by

$$\hat{y}(t_i) = \hat{y}(t_{i-1}) + hf\big(t_i, \hat{y}(t_i)\big), \tag{10}$$

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 second-order magnitude, the solution is taken as $y(t) = \hat{y}(t) + t^2 N(t, w)$. Other conditions for the neural network are the same as above. Figure 2(a) displays the solution behaviors of proposed schemes and comparisons with the analytic solution. Note that due to the stiff component, the solution improved rapidly. Here, the figure is plotted in the log-scale to observe the magnitude of the solution. It can show that the results from the proposed scheme are 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 that the proposed scheme works well even for the stiff problem. However, the results are not perfectly satisfactory 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, the number of the free parameters, etc.

3.2 Example 2

Next, we consider the following ODE:

$$y' + \left(t + \frac{1 + 3t^2}{1 + t + t^3}\right)y = t^3 + 2t + \frac{t^2(1 + 3t^2)}{1 + t + t^3}, \tag{11}$$

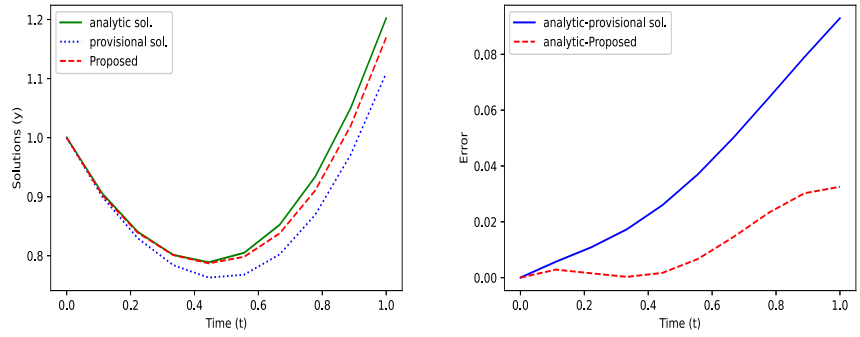

Figure 3 Comparison of (**a**) solution behaviors of the proposed scheme with analytic solution and Euler method and (**b**) residuals between current results and analytic solution

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

Similarly as above, a provisional solution $\hat{y}$ is estimated by the first-order explicit midpoint method, so the corresponding error has second-order $O(h^2)$ magnitude and the error is set to $t^2N(t, w)$. We plot the results in Fig. 3(a) and compare them with the analytic and provisional solutions. We easily check that the proposed scheme is quite close to the analytic solution. To examine 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 has a good accuracy for this problem. Also, we check the computational time for the proposed scheme. Notice that for increasing the reliability of computational time, we take the average by executing this code 100 times. It needs 6.2 seconds CPU time and 617 iterations for minimization of the parameters w.

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

$$\hat{y}(t_i) = \hat{y}(t_{i-1}) + hf\left(t_{i-1} + \frac{h}{2}, \hat{y}(t_{i-1}) + \frac{h}{2} f\left(t_{i-1}, \hat{y}(t_{i-1})\right)\right), \tag{12}$$

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 $t^3N(t, w)$. The results concerning the accuracy at grid points are presented in Fig. 4(a) with comparisons of the analytic and provisional solutions. It can be seen that the proposed scheme is closer to the analytic solution.

Since a higher-order solution is much closer to the analytic solution, we plot the residual between the analytic solution and the proposed one in Fig. 4(b) to inspect in details the accuracy of the proposed scheme. One can see that the proposed higher-order scheme overall has a quite smaller error compared with the second-order convergence method, although there are a few parts which have a larger error.

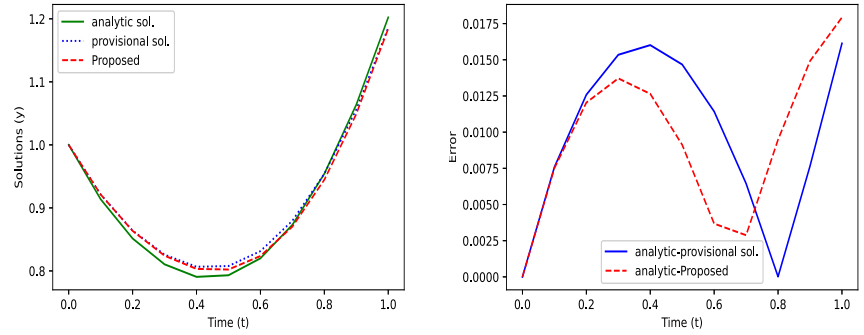

Figure 4 Comparison of (**a**) solution behaviors of the proposed scheme with analytic solution and Euler method and (**b**) residuals between current results and analytic solution

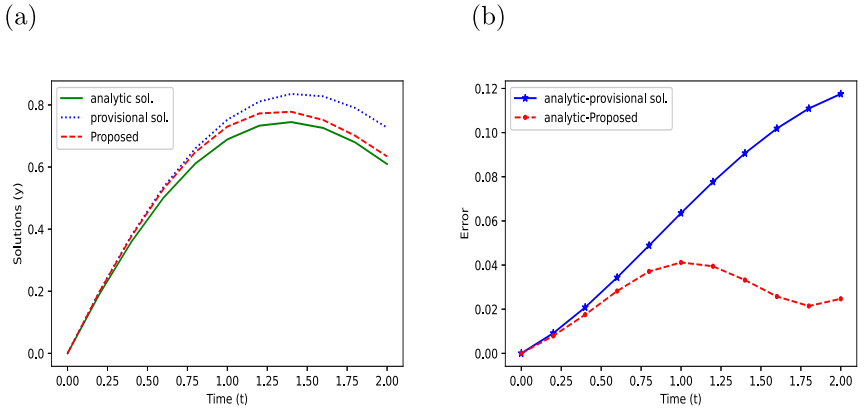

Figure 5 Comparison of (**a**) solution behaviors of the proposed scheme with analytic solution and Euler method and (**b**) residuals between current results and analytic solution

3.3 Example 3

In this subsection, the following differential equation is considered:

$$y' + \frac{1}{5}y = \exp\left(-\frac{1}{5}t\right)\cos(t), \tag{13}$$

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 from $[0, 2]$ in the 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 be $y(t) = \hat{y} + t^2 N(t, w)$. 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 and provisional solutions. It can be seen that the proposed scheme is closer to the analytic solution. To take a closer look at the accuracy, we calculate the difference between the analytic solution and the proposed one and plot it in Fig. 5(b). The L_2-norm of the difference between the results from the proposed scheme and the analytic solution

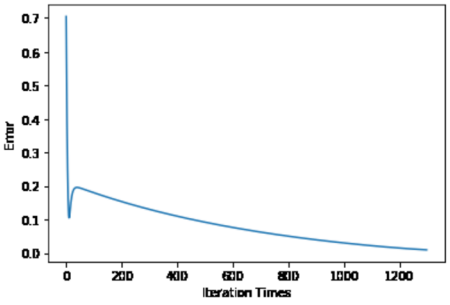

Figure 6 Error behavior according to the number of iterations in the minimization scheme

is 0.09291, whereas the L_2-norm of the comparison result is 0.24264. Therefore, as seen in the previous examples, we can easily conclude that the proposed scheme has a good accuracy for this problem.

Also, we check the error accuracy versus computational time of the proposed scheme. Since the computational time depends on the iteration times in the minimization scheme, we measure computational times and error accuracy by varying the iteration times. For the 100 iteration times, the accuracy is 0.17968 and it needs 1.1 seconds, and for the 2000 iterations, the accuracy is 0.02997 and the computational time is 10.5 seconds. Therefore, one can easily conclude that the more iterations we use, the more accurate results are generated. To support this argument, we plot error behavior according to the iterations in Fig. 6.

3.4 Example 4

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

$$y_1' = y_2, \tag{14}$$

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

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 from $[0, 2]$ in the 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. (10), due to its stability. The trial solutions are set to $y_1(t) = \hat{y_1} + t^2 N_1(t, w)$ and $y_2(t) = \hat{y_2} + t^2 N_2(t, w)$. We simulate this vector system and generate the numerical result as seen in Fig. 7. The results are compared with the analytic and provisional solutions. It can be seen that both numerical solutions of the system have a quite good accuracy.

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

$$H = y_1^2 + y_2^2, \tag{16}$$

and the value of H should be conserved. To examine the conservation property, we plot an orbit of solutions in Fig. 8(a) and the Hamiltonian H in Fig. 8(b). Figure 8(a) shows that

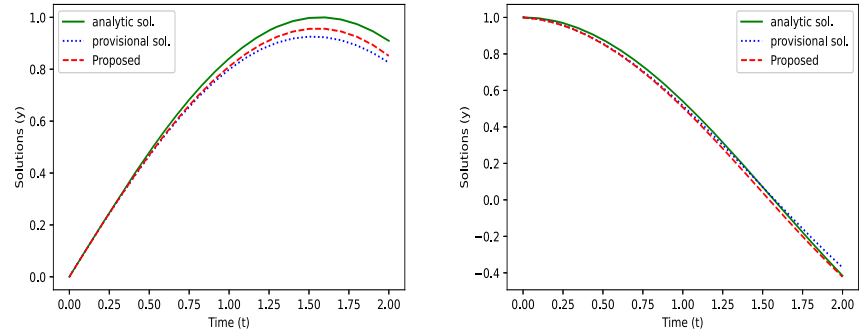

Figure 7 Comparison of solution behaviors of the proposed scheme with analytic solution and Euler method for (**a**) first component y_1 and (**b**) second component y_2

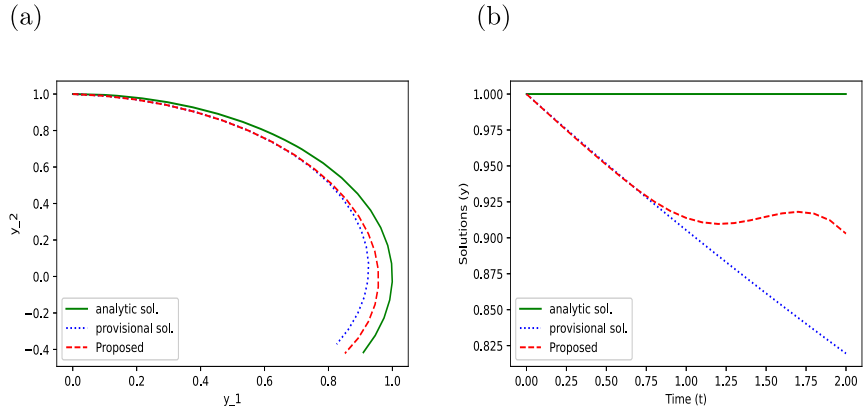

Figure 8 Comparison of (**a**) orbits of solutions and (**b**) behaviors of Hamiltonian values

the proposed scheme is closer to the analytic solution. Also, one can verify in Fig. 8(b) that 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 the 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 the 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 the Hamiltonian. Lastly, for more accurate results, we need to employ higher-order provisional solutions. Results along these directions will be reported soon.

Funding
The first author Nam was supported by the 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 the basic science research program through the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT) (grant number NRF-2022R1A2C1004588).

Availability of data and materials
Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
The author HN provided the basic idea of this work and wrote the manuscript. The author KRB implemented and debugged related codes. The author SB also provided several ideas to improve this manuscript and simulated several numerical experiments. All authors read and approved the final manuscript.

Author details
[1]Department of Liberal arts, Hongik University, Sejong, Republic of Korea. [2]Industry-Academia Cooperation Group, Hongik University, Sejong, Republic of Korea.

Publisher's Note

Received: 17 August 2021 Accepted: 6 June 2022 Published online: 15 June 2022

References

1. Atkinson, K.E.: An Introduction to Numerical Analysis. Wiley, New York (1989)
2. Atsalakis, G.S., Valavanis, K.P.: Forecasting stock market short-term trends using a neuro-fuzzy based methodology. Expert Syst. Appl. **36**(7), 10696–10707 (2009)
3. Box, G., Jenkins, G., Reinsel, G.: Time Series Analysis: Forecasting and Control. Prentice Hall, New York (1994)
4. Bu, S.: New construction of higher-order local continuous platforms for error correction methods. J. Appl. Anal. Comput. **6**(2), 443–462 (2016)
5. Bu, S.: A collocation methods based on the quadratic quadrature technique for fractional differential equations. AIMS Math. **7**(1), 804–820 (2022)
6. Bu, S., Bak, S.: Simulation of advection–diffusion–dispersion equations based on a composite time discretization scheme. Adv. Differ. Equ. **2020**, 132 (2020)
7. Bu, S., Huang, J., Minion, M.L.: Semi-implicit Krylov deferred correction methods for differential algebraic equations. Math. Comput. **81**(280), 2127–2157 (2012)
8. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT Numer. Math. **40**(2), 241–266 (2000)
9. Gear, C.W.: Numerical Initial Value Problems in Ordinary Differential Equations. Prentice Hall, New York (1971)
10. Graves, A., Mohamed, A., Hinton, G.: Speech recognition with deep recurrent neural networks. In: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pp. 6645–6649. IEEE (2013)
11. Greengard, L.: Spectral integration and two-point boundary value problems. SIAM J. Numer. Anal. **28**, 1071–1080 (1991)
12. Guo, X., Yang, C., Yuan, Y.: Dynamic-weighting hierarchical segmentation network for medical images. Med. Image Anal. **73** (2021)
13. Hairer, E., Lubich, C., Roche, M.: The Numerical Solution of Differential-Algebraic Systems by Runge–Kutta Methods. Springer, Berlin (1989)
14. Hairer, E., Norsett, S.P., Wanner, G.: Solving Ordinary Differential Equations. I: Nonstiff Problems. Springer Series in Computational Mathematics. Springer, Berlin (1993)
15. Huang, J., Jia, J., Minion, M.: Accelerating the convergence of spectral deferred correction methods. J. Comput. Phys. **214**(2), 633–656 (2006)
16. Jaitly, N., Nguyen, P., Senior, A., Vanhoucke, V.: Application of pretrained deep neural networks to large vocabulary speech recognition. Interspeech (2012)
17. Johnson, C.: 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. Kim, P., Bu, S.: Error control strategy in error correction methods. Kyungpook Math. J. **55**, 301–311 (2015)
19. Kim, P., Piao, X., Jung, W., Bu, S.: A new approach to estimating a numerical solution in the error embedded correction framework. Adv. Differ. Equ. **2018**, 168 (2018)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
21. Kumar, M., Yadav, N.: Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. Comput. Math. Appl. **62**, 3796–3811 (2011)
22. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**(5), 987–1000 (1998)
23. Li, Y., Wang, G., Nie, L., Wang, Q., Tan, W.: Distance metric optimization driven convolutional neural network for age invariant face recognition. Pattern Recognit. **75**, 51–62 (2018)
24. Lo, S.C.B., Chan, H.P., Lin, J.S., Li, H., Freedman, M.T., Mun, S.K.: Artificial convolution neural network for medical image pattern recognition. Neural Netw. **8**(7–8), 1201–1214 (1995)
25. Mai-Duy, N., Tran-Cong, T.: Numerical solution of differential equations using multiquadric radial basis function networks. Neural Netw. **14**, 185–199 (2001)
26. Ou, G., Murphey, Y.L.: Multi-class pattern classification using neural networks. Pattern Recognit. **40**(1), 4–18 (2007)
27. Pelt, D.M., Sethian, J.A.: Mixed-scale dense network for image analysis. Proc. Natl. Acad. Sci. **115**(2), 254–259 (2018)
28. Ramuhalli, P., Udpa, L., Udpa, S.S.: Finite-element neural networks for solving differential equations. In: IEEE Transactions on Neural Networks, vol. 16, pp. 1381–1392 (2005)
29. Shampine, L.F.: Error estimation and control for ODEs. J. Sci. Comput. **25**(1), 3–16 (2005)
30. Sultana, F., Sufian, A., Dutta, P.: Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey. Knowl.-Based Syst., 201–202 (2020)
31. Tan, L.S., Zainuddin, Z., Ong, P.: Solving ordinary differential equations using neural networks. AIP Conf. Proc. **1972**, 020070 (2018)
32. Yang, Y., Hou, M., Luo, J.: A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods. Adv. Differ. Equ. **2018**, 469 (2018)
33. Zheng, X., Yang, X.: Techniques for solving integral and differential equations by Legendre wavelets. Int. J. Syst. Sci. **40**(11), 1127–1137 (2009)