Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **13**, No. 1, pp. 101-118

DOI: 10.4208/aamm.OA-2019-0281
February 2021

Legendre Neural Network for Solving Linear Variable Coefficients Delay Differential-Algebraic Equations with Weak Discontinuities

Hongliang Liu[1], Jingwen Song[1], Huini Liu[1], Jie Xu[1] and Lijuan Li[2,*]

[1] *School of Mathematics and Computational Science & Hunan Key Laboratory for Computation and Simulation in Science and Engineering, Xiangtan University, Xiangtan, Hunan 411105, China*
[2] *School of Automation and Electronic Information, Xiangtan University, Xiangtan, Hunan 411105, China*

Received 29 September 2019; Accepted (in revised version) 30 June 2020

Abstract. In this paper, we propose a novel Legendre neural network combined with the extreme learning machine algorithm to solve variable coefficients linear delay differential-algebraic equations with weak discontinuities. First, the solution interval is divided into multiple subintervals by weak discontinuity points. Then, Legendre neural network is used to eliminate the hidden layer by expanding the input pattern using Legendre polynomials on each subinterval. Finally, the parameters of the neural network are obtained by training with the extreme learning machine. The numerical examples show that the proposed method can effectively deal with the difficulty of numerical simulation caused by the discontinuities.

AMS subject classifications: 65L80, 68T07, 68W25

Key words: Convergence, delay differential-algebraic equations, Legendre activation function, neural network.

1 Introduction

Delay differential-algebraic equations (DDAEs) arise in many areas of mathematical model, such as circuit design, mechanical system, power system, control theory [1], multi-body control system, bioeconomic system [2], fluid mechanics [3], chemical engineering. The reference [4] has indicated that differential-algebraic equations (DAEs) are neither differential equations nor algebraic equations. Differential equations only involve differentiation, while DAEs contain differentiation and integration, which changes the behavior of the

*Corresponding author.
Email: lilj@xtu.edu.cn (L. J. Li)

solution [5], so the numerical methods for solving differential equations can not be directly applied to solve DAEs. Furthermore, DDAEs are not only restricted by algebraic conditions but also affected by time delays. Bellen and Zennaro [6] have verified that delays can cause discontinuities and affect stability of the solution. Thus, algebraic conditions and delays cause some difficulties in numerical simulation of DDAEs. In recent years, many documents on its characteristics and theories of DDAEs have been discussed in [8–13], stability results of numerical methods for DDAEs have been presented in [14–21], and convergence analysis of numerical methods for DDAEs have been described in [22–26]. Furthermore, Ascher and Petzold [27] developed a numerical approach of high index DDAEs. The above works are under the basis of the solutions being smooth. Few studies have been carried out on the effects of DDAEs with weak discontinuities [28].

Neural networks have been widely used in the solution of mathematical physics problems. Now popular neural networks include feedforward neural network, radial basis function neural network, cosine basis function neural network, diagonal recurrent neural network, cellular neural network, finite-element neural network, etc. The neural network can obtain the weights and structure of the network by training and learning, showing strong self-learning and adaptive ability. That is, the network structure, node weights and step sizes can be automatically adjusted according to environmental requirements. An advantage of neural networks to solve differential equations is that the solution of the differential equations can be expressed as a differential function. Earlier in 1992, Shelton et al. [29] applied neural network to deal with the coupled nonlinear ordinary differential equations, and it was also used to solve (non) linear ordinary differential equations [30–36], partial differential equations [37–44], delay differential equations [45,46], stiff differential equations [47, 48], stochastic differential equations [49] and fractional differential equations [50–52]. Kozlov and Tiumentsev [53] introduced a neural network based on semi-empirical models for solving DAEs of index 2. Yang et al. [54] solved DAEs based on the artificial neural network. Few scholars manage to solve DDAEs with weak discontinuities at present.

Compared with the traditional methods, the advantages of the Legendre neural network (LNN) to solve DDAEs are as follows [55, 56]. (1) LNN can overcome the iterative process commonly used in traditional numerical methods. (2) The computational complexity of LNN does not increase quickly with the increase of sample points. (3) LNN has fault tolerance and tolerance capabilities. The contribution of each neuron and each connection to the overall network function is small, so the failure of a few neurons and connections has little effect on the network function. Based on the above advantages, in this paper, LNN [58] combined with the extreme learning machine (ELM) algorithm [59,60] is applied to solve the variable coefficients linear DDAEs with weak discontinuities. Firstly, the interval is divided equally into multiple subintervals according to the weak discontinuity points. Secondly, we eliminate the hidden layer by using the Legendre basis function that is used to extend the input pattern. Thirdly, the weights of LNN are obtained by ELM training neural network on each subinterval. Finally, we get the approximate solutions of the DDAEs on the whole interval.

2 Legendre orthogonal polynomial and its properties

Definition 2.1 ([7]). Let $f(t), g(t) \in C[a,b]$, $\rho(t)$ is the weight function in $[a,b]$. If

$$(f(t),g(t)) = \int_a^b \rho(t) f(t) g(t) dt = 0,$$

$f(t)$ and $g(t)$ are orthogonal with respect to $\rho(t)$ in $[a,b]$.

Definition 2.2 ([7]). Orthogonal polynomial with weight of $\rho(t) = 1$ in $[-1,1]$ is called Legendre polynomial $P_n(t)$, and

$$P_0(t) = 1, \quad P_1(t) = t, \cdots, \quad P_n(t) = \frac{1}{2^n n!} \frac{d^n}{dt^n}(t^2-1)^n, \quad t \in [-1,1], \quad n = 1, \cdots.$$

Next we will introduce properties of the Legendre polynomial.

Property 1. $P_n(t)$ and $P_m(t)$ are the Legendre polynomials and satisfy

$$(P_n(t),P_m(t)) = \int_{-1}^{1} P_n(t) P_m(t) dt = \begin{cases} 0, & m \neq n, \\ \dfrac{2}{2n+1}, & m = n. \end{cases}$$

Property 2. $P_{2n}(t)$ only contains even power and $P_{2n+1}(t)$ only contains odd power, and we have

$$P_n(-t) = (-1)^n P_n(t).$$

Property 3. We have recursion formula

$$(n+1)P_{n+1}(t) = (2n+1)tP_n(t) - nP_{n-1}(t), \quad n = 1, \cdots.$$

Property 4. $P_n(t)$ has n different zeros in $(-1,1)$.

Lemma 2.1 ([58]). *Suppose that the vector $P(t)$ is defined as $P(t) = [P_0(t), P_1(t), \cdots, P_N(t)]^T$, where $P_n(t)$ $(n = 0, \cdots, N)$ is the nth order Legendre polynomial on the interval $[-1,1]$, and let $P'(t)$ be defined as $P'(t) = [P_0'(t), P_1'(t), \cdots, P_N'(t)]^T$, where $P_n'(t)$, $(n = 0, \cdots, N)$ is the derivative of $P_n(t)$. Then $P'(t) = DP(t)$, where D is the Legendre operational matrix given by*

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 5 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 3 & 0 & 7 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 5 & 0 & 9 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 3 & 0 & 7 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 5 & 0 & 9 & \cdots & 2n-1 & 0 \end{bmatrix}_{(N+1)\times(N+1)}.$$

3 Legendre neural network for solving linear variable coefficients DDAEs with weak discontinuities

3.1 Legendre neural network and approximation

In this paper, Legendre polynomial is used as activation function to construct a single layer neural network, which consists of three parts: input layer, output layer and hidden layer. β is the output weight connecting the hidden node with the output node, where $\beta=[\beta_0,\beta_1,\cdots,\beta_N]^T$. The structure of neural network is shown in Fig. 1.

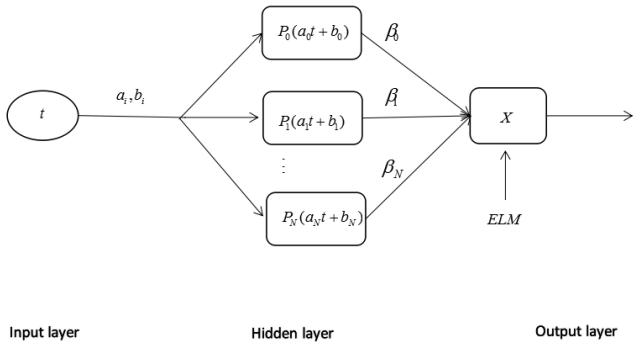

Figure 1: Legendre neural network.

Lemma 3.1 ([58]). *For any continuous function $x(t):[a,b]\to\mathbb{R}$, there are a natural number N, constants a_i, b_i, β_i $(i=0,\cdots,N)$, and Legendre polynomial $P_n(t)$ $(n=0,\cdots,N)$, such that the Legendre neural network with $N+1$ neurons is given by*

$$x_{LNN}(t)=\sum_{i=0}^{N}\beta_iP_i(a_it+b_i), \tag{3.1}$$

$x_{LNN}(t)$ is an approximation of $x(t)$, and

$$\|x(t)-x_{LNN}(t)\|=\left\|x(t)-\sum_{i=0}^{N}\beta_iP_i(a_it+b_i)\right\|<\varepsilon,$$

where ε is a small positive constant.

We consider linear variable coefficients DDAEs with weak discontinuities as follows

$$\begin{aligned}E(t)x'(t)=&A_1(t)x(t)+B_1(t)x(t-\tau)+C_1(t)y(t)\\&+D_1(t)y(t-\tau)+f_1(t),\qquad t\in[0,T],\end{aligned} \tag{3.2a}$$

$$\begin{aligned}0=&A_2(t)x(t)+B_2(t)x(t-\tau)+C_2(t)y(t)\\&+D_2(t)y(t-\tau)+f_2(t),\qquad t\in[0,T],\end{aligned} \tag{3.2b}$$

$$x(t)=\psi_1(t), \qquad -\tau\le t\le 0, \tag{3.2c}$$
$$y(t)=\psi_2(t), \qquad -\tau\le t\le 0, \tag{3.2d}$$

when $t=0$, the algebraic condition (3.2b) holds, τ is a positive constant, $x(t)=[x_1(t),x_2(t),\cdots,x_n(t)]^T\in\mathbb{R}^n$. If $t=k\tau$, $(k\in N)$, $x(t)$ is the weak discontinuity. If $t\neq k\tau$, $x(t)$ is differentiable. $y(t)=[y_1(t),y_2(t),\cdots,y_m(t)]^T\in\mathbb{R}^m$ and $y(t)$ is continuous, $E(t)\in\mathbb{R}^{n\times n}$ is a nonsingular continuous matrix function, $A_1(t)$, $B_1(t)\in\mathbb{R}^{n\times n}$, $C_1(t)$, $D_1(t)\in\mathbb{R}^{n\times m}$, $A_2(t)$, $B_2(t)\in\mathbb{R}^{m\times n}$ and $C_2(t)$, $D_2(t)\in\mathbb{R}^{m\times m}$ are all continuous matrix functions, $f_1(t)\in\mathbb{R}^{n\times 1}$, $f_2(t)\in\mathbb{R}^{m\times 1}$, $\psi_1(t)\in\mathbb{R}^{n\times 1}$ and $\psi_2(t)\in\mathbb{R}^{m\times 1}$ are all continuous vector functions.

We take $a_i=1$, $b_i=0$ in the formula (3.1). Next we apply LNN to solve Eqs. (3.2) and choose Legendre polynomial $P_n(t)$ as activation function of neural network, the approximation of $x(t)$ can be denoted by

$$x_{iLNN}(t)=\sum_{j=0}^{N}\beta_{ij}P_j(t), \quad i=1,\cdots,n,$$

let $P(t)=[P_0(t),P_1(t),\cdots,P_N(t)]^T$, $\beta_i=[\beta_{i0},\beta_{i1},\cdots,\beta_{iN}]^T$, $i=1,\cdots,n$, then

$$x_{iLNN}(t)=P(t)^T\beta_i, \quad i=1,\cdots,n, \tag{3.3}$$

similarly,

$$y_{jLNN}(t)=\sum_{i=0}^{N}\hat{\beta}_{ji}P_i(t), \quad j=1,\cdots,m,$$

where $\hat{\beta}_j=[\hat{\beta}_{j0},\hat{\beta}_{j1},\cdots,\hat{\beta}_{jN}]^T$, $j=1,\cdots,m$, and $y_{jLNN}(t)$ is approximate to $y_j(t)$ and satifies

$$y_{jLNN}(t)=P(t)^T\hat{\beta}_j, \quad j=1,\cdots,m. \tag{3.4}$$

Since $x(t)$, $y(t)$ are exact solutions of Eqs. (3.2) and they are continuous functions, according to Lemma 3.1, the approximate solutions and true solutions satisfy the following inequality

$$\|x(t)-x_{LNN}(t)\|<\varepsilon, \quad \|y(t)-y_{LNN}(t)\|<\varepsilon.$$

3.2 ELM algorithm for training the Legendre neural network

Given that the solutions of Eqs. (3.2) are weakly discontinuous, the $[0,T]$ interval is divided into $[0,\tau]$, $[\tau,2\tau],\cdots,[(k-1)\tau,k\tau],\cdots$, where $k\tau$ are weak discontinuity points.

(1) When $t\in[0,\tau]$, according to Lemma 2.1 and substitute Eqs. (3.3), (3.4) into Eqs. (3.2),

then

$$\begin{cases} [E(t)(I_{n\times n}\bigotimes P^T(t)D^T)-A_1(t)(I_{n\times n}\bigotimes P^T(t))]U-C_1(t)(I_{m\times m}\bigotimes P^T(t))V \\ \quad =f_1(t)+B_1(t)\psi_1(t-\tau)+D_1(t)\psi_2(t-\tau), \\ -A_2(t)(I_{n\times n}\bigotimes P^T(t))U-C_2(t)(I_{m\times m}\bigotimes P^T(t))V \\ \quad =f_2(t)+B_2(t)\psi_1(t-\tau)+D_2(t)\psi_2(t-\tau), \\ I_{n\times n}\bigotimes P^T(0)=\psi_1(0), \\ I_{m\times m}\bigotimes P^T(0)=\psi_2(0), \end{cases} \tag{3.5}$$

where $\bigotimes$ is Kronecker product, and

$$U=\begin{bmatrix}\beta_1\\ \beta_2\\ \vdots\\ \beta_n\end{bmatrix},\quad V=\begin{bmatrix}\hat{\beta}_1\\ \hat{\beta}_2\\ \vdots\\ \hat{\beta}_m\end{bmatrix},\quad I_{n\times n}\bigotimes P^T(t)=\begin{bmatrix}P^T(t) & 0 & \cdots & 0\\ 0 & P^T(t) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & P^T(t)\end{bmatrix}.$$

Eqs. (3.5) are written as

$$\begin{pmatrix} a_{11}(t) & a_{12}(t)\\ a_{21}(t) & a_{22}(t)\\ a_{31}(0) & a_{32}(0)\\ a_{41}(0) & a_{42}(0)\end{pmatrix}\begin{pmatrix} U\\ V\end{pmatrix}=\begin{pmatrix} g_1(t)\\ g_2(t)\\ \psi_1(0)\\ \psi_2(0)\end{pmatrix}, \tag{3.6}$$

where

$$\begin{aligned} &a_{11}(t)=E(t)(I_{n\times n}\bigotimes(P^T(t)D^T))-A_1(t)(I_{n\times n}\bigotimes P^T(t)),\\ &a_{12}(t)=-C_1(t)(I_{m\times m}\bigotimes P^T(t)),\\ &a_{21}(t)=-A_2(t)(I_{n\times n}\bigotimes P^T(t)),\\ &a_{22}(t)=-C_2(t)(I_{m\times m}\bigotimes P^T(t)),\\ &a_{31}(t)=I_{n\times n}\bigotimes P^T(t),\quad a_{32}(t)=\mathbf{0}_{n\times(m(N+1))},\\ &a_{41}(t)=\mathbf{0}_{m\times(n(N+1))},\quad a_{42}=I_{m\times m}\bigotimes P^T(t),\\ &g_1(t)=f_1(t)+B_1(t)\psi_1(t-\tau)+D_1(t)\psi_2(t-\tau),\\ &g_2(t)=f_2(t)+B_2(t)\psi_2(t-\tau)+D_2(t)\psi_2(t-\tau). \end{aligned}$$

We divide arbitrarily $[0,\tau]$ into M segments and set $t_{0,l}$, $(l=0,\cdots,M)$ are the interval nodes. We take $t_{0,0}=0$, where M satisfies $N\leq M$, and we define

$$H^{(1)}=\begin{bmatrix} a_{11}(t_{0,1}) & a_{12}(t_{0,1})\\ a_{21}(t_{0,1}) & a_{22}(t_{0,1})\\ \vdots & \vdots\\ a_{11}(t_{0,M}) & a_{12}(t_{0,M})\\ a_{21}(t_{0,M}) & a_{22}(t_{0,M})\\ a_{31}(t_{0,0}) & a_{32}(t_{0,0})\\ a_{41}(t_{0,0}) & a_{42}(t_{0,0})\end{bmatrix},\quad \Lambda^{(1)}=\begin{bmatrix} U^{(1)}\\ V^{(1)}\end{bmatrix},\quad I^{(1)}=\begin{bmatrix} g_1(t_{0,1})\\ g_2(t_{0,1})\\ \vdots\\ g_1(t_{0,M})\\ g_2(t_{0,M})\\ \psi_1(t_{0,0})\\ \psi_2(t_{0,0})\end{bmatrix},$$

the dimensions of $H^{(1)}$, $\Lambda^{(1)}$ and $I^{(1)}$ are $(n+m)(M+1)\times(n+m)(N+1)$, $(n+m)(N+1)\times 1$ and $(n+m)(M+1)\times 1$ respectively. At each discrete point $t_{0,l}$, $(l=0,\cdots,M)$, Eqs. (3.6) is equivalent to the matrix form

$$H^{(1)}\Lambda^{(1)}=I^{(1)}.$$

Applying the ELM algorithm, we can obtain

$$(\tilde{U}^{(1)},\tilde{V}^{(1)})^T=\arg\min_{U^{(1)},V^{(1)}}\|H^{(1)}(U^{(1)},V^{(1)})^T-I^{(1)}\|$$

in $[0,\tau]$, the approximate solutions of Eqs. (3.2) can be denoted as

$$x_{LNN}^{(1)}(t)=(I_{n\times n}\otimes P^T(t))\tilde{U}^{(1)},\quad y_{LNN}^{(1)}(t)=(I_{m\times m}\otimes P^T(t))\tilde{V}^{(1)}.$$

(2) Assume the numerical solutions of Eqs. (3.2) in $[(k-2)\tau,(k-1)\tau]$ $(k=2,\cdots)$ have been obtained and written as

$$x_{LNN}^{(k-1)}(t)=(I_{n\times n}\otimes P^T(t))\tilde{U}^{(k-1)},\quad y_{LNN}^{(k-1)}(t)=(I_{m\times m}\otimes P^T(t))\tilde{V}^{(k-1)},$$

in order to find the numerical solutions of the equations in $[(k-1)\tau,k\tau]$, we take the numerical solutions $x_{LNN}^{(k-1)}(t)$ and $y_{LNN}^{(k-1)}(t)$ in $[(k-2)\tau,(k-1)\tau]$ as the initial conditions for Eqs. (3.2) on next subinterval. Next we only need to replace the $\psi_1(t)$, $\psi_2(t)$ in Eqs. (3.5) with $x_{LNN}^{(k-1)}(t)$ and $y_{LNN}^{(k-1)}(t)$, and repeat the solving process for $t\in[0,\tau]$. Similarly, $[(k-1)\tau,k\tau]$ is divided into M parts, we can get

$$H^{(k)}\Lambda^{(k)}=I^{(k)}.$$

Applying the ELM algorithm, we obtain

$$(\tilde{U}^{(k)},\tilde{V}^{(k)})^T=\arg\min_{U^{(k)},V^{(k)}}\|H^{(k)}(U^{(k)},V^{(k)})^T-I^{(k)}\|$$

in $[(k-1)\tau,k\tau]$, the approximate solutions of Eqs. (3.2) can be expressed as

$$x_{LNN}^{(k)}(t)=(I_{n\times n}\otimes P^T(t))\tilde{U}^{(k)},\quad y_{LNN}^{(k)}(t)=(I_{m\times m}\otimes P^T(t))\tilde{V}^{(k)}.$$

Substituting the weights into the approximate solutions on each subinterval, we can obtain the numerical solutions of Eqs. (3.2) in $[(k-1)\tau,k\tau]$. Other intervals are analogized according to the above solving process, and we can get the numerical solutions of Eqs. (3.2) in $[0,T]$.

4 Steps for solving DDAEs by using Legendre neural network and ELM algorithm

Step 1 The interval $[0,T]$ is divided into $[(k-1)\tau,k\tau]$, $k=1,\cdots$, where $k\tau$ are weak discontinuities, then we divide $[(k-1)\tau,k\tau]$ into M segments and $t_{k,l}$ $(l=0,\cdots,M)$ are the interval nodes.

Step 2 Construct approximate solutions by Legendre polynomials on kth subinterval, $x_{LNN}^{(k)}=(I_{n\times n}\otimes P^T(t))U^{(k)}$, $y_{LNN}^{(k)}=(I_{m\times m}\otimes P^T(t))V^{(k)}$.

Step 3 The approximate solutions $x_{LNN}^{(k)}$, $y_{LNN}^{(k)}$ and their derivatives at kth discrete point are brought into the original equations (3.2) to obtain the equation $H^{(k)}\Lambda^{(k)}=I^{(k)}$.

Step 4 Apply ELM optimization algorithm to minimize $\|H^{(k)}\Lambda^{(k)}-I^{(k)}\|$ and calculate the neural network output weights $\tilde{\Lambda}^{(k)}=H^{(k)\dagger}I^{(k)}$, where $H^{(k)\dagger}$ is the Moore-Penrose generalized inverse of matrix $H^{(k)}$.

Step 5 Substitute the $\tilde{\Lambda}^{(k)}$ into the numerical solutions of the original equations (3.2) on the interval $[(k-1)\tau,k\tau]$, we can get $x_{LNN}^{(k)}$, $y_{LNN}^{(k)}$.

Step 6 According to the above steps, similarly solve the original equations (3.2) on *(k+1)*th subinterval and then obtain the numerical solutions in $[0,T]$.

5 Connections between the proposed method and the hp-spectral collocation method

The similarity between the two methods is that polynomials are used as a set of basis functions and their linear combinations are used to give the form of an approximate solution. At the same time, there are some differences between the two methods. The spectral collocation method uses the orthogonality of the Legendre polynomial, and the solution interval must be transformed to $[-1,1]$ interval [57]. The proposed method does not use the orthogonality of the polynomials, while Legendre polynomials are only used as a set of activation functions. The theory of spectral collocation method is better but it is limited in taking points, while our method can do arbitrary taking points and implement easily.

6 Convergence discussion

Known by reference [59], given an arbitrary distinct set of samples (x_i,t_i), $x_i\in\mathbb{R}^n$, $t_i\in\mathbb{R}^m$, standard single hidden layer feedforward network with $\tilde{N}$ hidden nodes is modeled as

follows

$$\sum_{i=1}^{\widetilde{N}}\beta_i g(\omega_i x_j+b_i),\quad j=1,\cdots,N, \tag{6.1}$$

where g is the activation function, ω_i is the input weight, b_i is the threshod, and β_i is the output weight. When the error between the output and the samples is zero, that is the neural network completely approximates the samples, and the following equations hold

$$\sum_{i=1}^{\widetilde{N}}\beta_i g(\omega_i x_j+b_i)=t_j,\quad j=1,\cdots,N. \tag{6.2}$$

Eqs. (6.2) are written in matrix form

$$H_{N\times\tilde{N}}\beta_{\tilde{N}\times m}=T_{N\times m}. \tag{6.3}$$

Next, we introduce an important lemma.

Lemma 6.1 ([59]). *Given any small positive value $\varepsilon>0$ and activation function $g:\mathbb{R}\to\mathbb{R}$ which is infinitely differentiable in any interval, there exists $\tilde{N}\leq N$ such that for N arbitrary distinct samples (x_i,t_i) where $x_i\in\mathbb{R}^n$ and $t_i\in\mathbb{R}^m$ for any w_i and b_i randomly chosen from any intervals of $\mathbb{R}^n$ and $\mathbb{R}$, respectively, according to any continuous probability distribution, then with probability one, $\|H_{N\times\tilde{N}}\beta_{\tilde{N}\times m}-T_{N\times m}\|<\varepsilon$.*

By Lemma 6.1, we induce the following Theorem 6.1.

Theorem 6.1. *Given any small positive value $\varepsilon>0$, we select $M+1$ sample points $(t_i,x(t_i))$, $(t_i,y(t_i))$, where $t_i\in\mathbb{R}$, $x(t_i)\in\mathbb{R}^n$, $y(t_i)\in\mathbb{R}^m$, and Legendre activation function $P(t):\mathbb{R}\to\mathbb{R}$. On each solution subinterval, we select $N+1$ hidden nodes which satisfy the condition $N\leq M$, using the proposed method to solve DDAEs Eqs.* (3.2)*, then*

$$\|H^{(k)}\Lambda^{(k)}-I^{(k)}\|<\varepsilon\quad in\quad [(k-1)\tau,k\tau],\quad k=1,\cdots.$$

Proof. Since the Legendre activation function $P(t)$ is infinitely differentiable and the number of hidden nodes is less than or equal to the number of sample points, the conditions of Lemma 6.1 are satisfied. Therefore,

$$\|H^{(k)}\Lambda^{(k)}-I^{(k)}\|<\varepsilon\quad \text{in}\ [(k-1)\tau,k\tau].$$

Thus, we complete the proof. □

7 Numerical results

Example 7.1. We consider the linear DDAEs with weak discontinuties [9]

$$Ex'(t)=Ax(t)+Dx(t-\tau)+f(t),\quad t\in[0,3], \tag{7.1}$$

where $\tau=1$, $f(t)=\mathbf{0}$, and

$$E=\begin{bmatrix}1&0\\0&0\end{bmatrix},\quad A=\begin{bmatrix}0&1\\1&0\end{bmatrix},\quad D=\begin{bmatrix}0&0\\0&-1\end{bmatrix},$$

the initial conditions

$$\begin{cases}x_1(t)=\dfrac{1}{3}(t-1)^3+(t-1)^2-1, & t\in[-\tau,0],\\ x_2(t)=\dfrac{1}{3}t^3+t^2-1, & t\in[-\tau,0],\end{cases}$$

the exact solutions

$$x_1(t)=\begin{cases}\dfrac{1}{3}(t-1)^3+(t-1)^2-1, & t\in[0,1],\\ (t-1)^2-1, & t\in[1,2],\\ 2t-4, & t\in[2,3],\end{cases}\quad x_2(t)=\begin{cases}t^2-1, & t\in[0,1],\\ 2t-2, & t\in[1,2],\\ 2, & t\in[2,3].\end{cases}$$

When $t=2$, $x_2(t)$ is weakly discontinuous. Firstly, we take $M=30,N=9$ and use overall simulation method to solve Eqs. (7.1) in $[0,T]$. Secondly, we apply the proposed piecewise simulation method with $M=10$, $N=9$. The global error of the ith component in t_j is denoted as hx_i^j

$$hx_i^j=\frac{|x_i^j-x_i(t_j)|}{\max\{1,x_i(t_j)\}},$$

where x_i^j is the numerical solution of the ith component in t_j. Similarly, the piecewise error of the ith component in t_j is denoted as Dx_i^j. We use the same mark below.

Fig. 2 shows the effect of global approximation of the true solutions with $M=30$ and $N=9$. The errors of $x_1(t)$ and $x_2(t)$ in $[0,3]$ are recorded in Table 1. Since the solution $x_2(t)$ of Eqs. (7.1) is weakly discontinuious, the global approximation approach is not suitable for solving Eqs. (7.1).

According to Table 1, in the case of global simulation, the maximum error of $x_1(t)$ is 5.73e-02 and the maximum error of $x_2(t)$ is 3.94e-01. In the case of piecewise simulation, the maximum errors of $x_1(t)$ and $x_2(t)$ are 7.19e-07 and 1.69e-06 respectively, which proves the superiority and feasibility of the developed method.

Now, in order to verify that our proposed method is not restricted by taking points, we select uniformly distributed random points on each subinterval to solve Eqs. (7.1).

Fig. 3, Fig. 4 and Fig. 5 show the simulation effect of the true solutions and the numerical solutions by using developed piecewise approximation method with $M=10$, $N=9$. According to the results in Table 2, random partition method is also applicable and well implemented. For convenience, we will use equidistant points to solve.

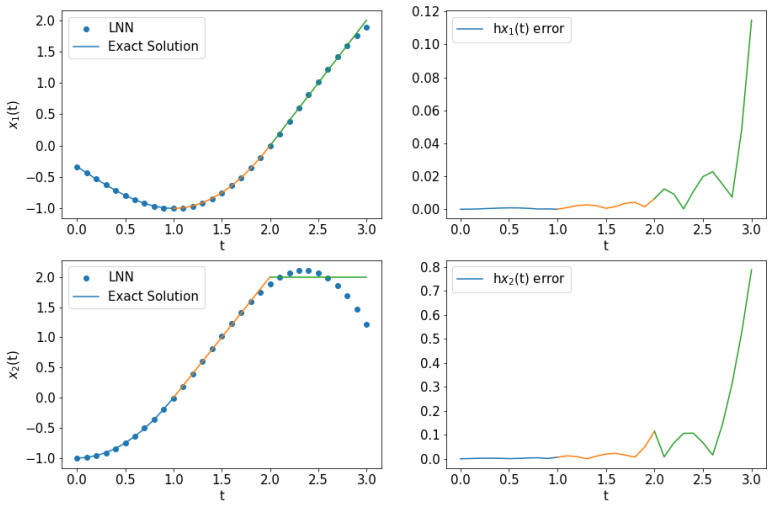

Figure 2: The global approximation in $[0,3]$ of Example 7.1.

Table 1: Error comparison of Example 7.1.

t	$Dx_1(t)$	$hx_1(t)$	$Dx_2(t)$	$hx_2(t)$
0	2.30e-11	8.20e-11	1.04e-10	6.64e-10
0.2	1.40e-11	2.09e-04	7.90e-11	2.17e-03
0.4	1.00e-11	7.09e-04	2.00e-11	2.15e-03
0.6	1.50e-11	8.04e-04	5.40e-11	1.62e-03
0.8	2.40e-11	1.27e-04	6.20e-11	4.16e-03
1	3.40e-11	5.40e-07	1.99e-10	6.46e-03
1	7.25e-09	5.40e-07	2.53e-08	6.46e-03
1.2	1.32e-08	2.17e-03	6.63e-08	9.15e-03
1.4	2.46e-08	2.14e-03	1.51e-07	1.08e-02
1.6	4.49e-08	1.62e-03	2.58e-07	1.90e-02
1.8	7.90e-08	4.16e-03	3.66e-07	4.56e-03
2	1.34e-07	6.46e-03	5.19e-07	5.73e-02
2	8.76e-09	6.46e-03	5.26e-07	5.73e-02
2.2	1.48e-07	9.15e-03	1.23e-07	3.33e-02
2.4	2.71e-07	1.08e-02	3.35e-07	5.35e-02
2.6	4.14e-07	1.90e-02	5.37e-07	7.92e-03
2.8	5.41e-07	4.56e-03	9.53e-07	1.57e-01
3	7.19e-07	5.73e-02	1.69e-06	3.94e-01

Example 7.2. Given the following linear DDAEs with constant coefficients

$$Ex'(t)=Ax(t)+Bx(t-\tau)+f(t),\quad t\in[0,3], \tag{7.2}$$

Table 2: The random points errors of Example 7.1.

t	0.00	0.08	0.44	0.54	0.78	1.00
$Dx_1(t)$	1.98e-13	5.54e-14	1.36e-11	2.07e-11	3.91e-11	5.73e-11
$Dx_2(t)$	4.00e-11	1.30e-11	1.45e-10	1.84e-10	2.52e-10	3.92e-10
t	1.00	1.08	1.44	1.54	1.78	2.00
$Dx_1(t)$	6.31e-09	6.45e-09	5.03e-09	1.46e-08	5.91e-08	1.42e-07
$Dx_2(t)$	4.73e-08	5.61e-08	1.06e-07	1.15e-07	1.15e-07	1.22e-07
t	2.00	2.08	2.44	2.54	2.78	3.00
$Dx_1(t)$	8.06e-09	1.04e-07	2.23e-07	2.70e-07	2.91e-07	3.27e-07
$Dx_2(t)$	1.04e-07	7.15e-07	1.58e-07	4.33e-08	2.10e-07	2.93e-07

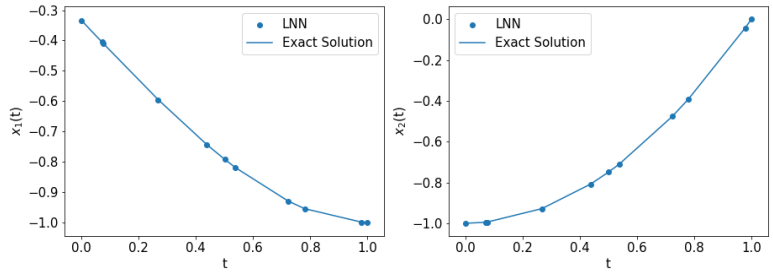

Figure 3: Simulation of random points in $[0,1]$ of Example 7.1.

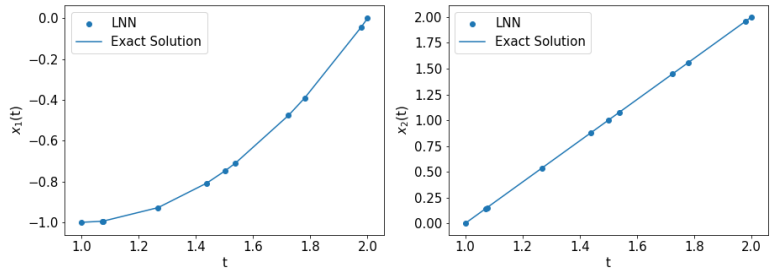

Figure 4: Simulation of random points in $[1,2]$ of Example 7.1.

where $\tau=1$, and

$$E=\begin{bmatrix}1&0\\0&0\end{bmatrix},\quad A=\begin{bmatrix}1&1\\2&1\end{bmatrix},\quad B=\begin{bmatrix}-2&-1\\0&0\end{bmatrix},\quad f(t)=\mathbf{0},$$

the initial conditions

$$\begin{cases}x_1(t)=e^{-t}, & t\in[-\tau,0],\\ x_2(t)=-2e^{-t}, & t\in[-\tau,0],\end{cases}$$

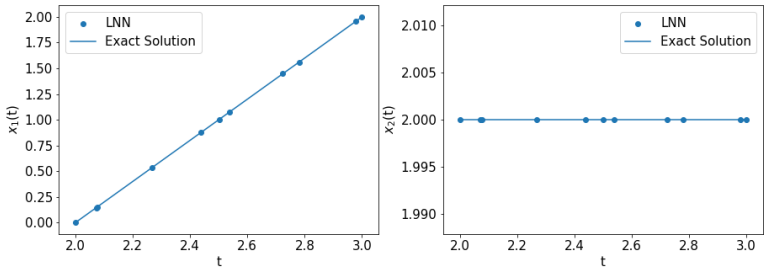

Figure 5: Simulation of random points in [2,3] of Example 7.1.

Table 3: Errors of different methods for Example 7.2.

t	$Dx_1(t)$	$hx_1(t)$	$Vx_1(t)$	$Dx_2(t)$	$hx_2(t)$	$Vx_2(t)$
0.0	1.83e-14	1.16e-08	0.00e+00	2.80e-13	5.67e-09	0.00e+00
0.2	1.38e-12	5.09e-05	1.38e-12	4.76e-12	1.02e-04	2.77e-12
0.4	1.49e-12	6.00e-05	6.95e-10	5.95e-12	1.20e-04	1.39e-09
0.6	1.24e-12	5.46e-05	2.62e-08	2.36e-13	1.09e-04	5.24e-08
0.8	5.98e-13	4.60e-05	3.42e-07	4.97e-14	9.19e-05	6.85e-07
1.0	1.19e-12	3.78e-05	2.50e-06	2.29e-12	7.55e-05	5.01e-06
1.0	3.00e-12	3.78e-05	2.50e-06	3.00e-12	7.55e-05	5.01e-06
1.2	6.00e-12	3.09e-05	1.27e-05	1.70e-11	6.19e-05	2.54e-05
1.4	1.10e-11	2.53e-05	4.99e-05	6.40e-11	5.06e-05	9.97e-05
1.6	1.90e-11	2.07e-05	1.63e-04	1.52e-10	4.14e-05	3.26e-04
1.8	3.20e-11	1.70e-05	4.62e-04	3.20e-10	3.39e-05	9.24e-04
2.0	5.40e-11	1.39e-05	1.17e-03	6.07e-10	2.78e-05	2.35e-03
2.0	1.50e-10	1.39e-05	1.17e-03	5.92e-10	2.78e-05	2.35e-03
2.2	8.10e-11	1.14e-05	2.72e-03	5.95e-10	2.28e-05	5.44e-03
2.4	7.60e-11	9.31e-06	5.85e-03	7.16e-10	1.86e-05	1.17e-02
2.6	1.01e-10	7.63e-06	1.18e-02	3.30e-10	1.52e-05	2.37e-02
2.8	1.41e-10	6.24e-06	2.27e-02	7.27e-10	1.25e-05	4.53e-02
3.0	2.16e-10	5.10e-06	4.15e-02	1.21e-09	1.02e-05	8.30e-02

and the exact solutions

$$\begin{cases} x_1(t)=e^{-t}, & t\in[0,3], \\ x_2(t)=-2e^{-t}, & t\in[0,3]. \end{cases}$$

In order to compare our proposed method and the traditional approximate analytical solution method, we use the variational iteration method [61] to solve Eqs. (7.2), and the errors of $x_1(t)$ and $x_2(t)$ are respectively denoted as $Vx_1(t)$ and $Vx_2(t)$.

In Table 3, we use piecewise simulation with $M=10$, $N=9$, overall simulation with $M=30$, $N=9$ and variational iteration method to solve Eqs. (7.2). In case of iterating step 8 using variational iteration method, the maximum errors of $x_1(t)$ of the three methods are 2.16e-10, 6.00e-05 and 4.15e-02 respectively. the maximum errors of $x_2(t)$ of the three

Table 4: The piecewise error of Example 7.3.

interval	M,N	$x_1(t)$	$x_2(t)$
[0,1]	10,9	1.39e-11	2.36e-11
[1,2]	70,9	2.27e-09	1.76e-09
[2,3]	460,9	2.20e-09	1.68e-08
[3,4]	380,9	1.98e-08	9.11e-09
[4,5]	333,9	2.08e-07	4.77e-08

methods are 1.21e-09, 1.20e-04 and 8.30e-02 respectively. It is obvious that the maximum error is the lowest by using piecewise simulation, which verifies the superiority of our proposed method.

Example 7.3. Given the following linear DDAEs with weak discontinuties

$$Ex'(t)=A(t)x(t)+B(t)x(t-\tau)+f(t),\quad t\in[0,5], \tag{7.3}$$

where $\tau=1$, and

$$E=\begin{bmatrix}1&1\\0&0\end{bmatrix},\quad A(t)=\begin{bmatrix}t&0\\0&2\end{bmatrix},\quad B(t)=\begin{bmatrix}1&0\\0&t\end{bmatrix},$$

$$f_1(t)=\begin{cases}(1-t)e^t, & t\in[0,1],\\ (1-t)e^t-e^{t-1}+1, & t\in[1,5],\end{cases}\quad f_2(t)=\begin{cases}-2t, & t\in[0,1],\\ -t-t^2, & t\in[1,5],\end{cases}$$

the initial conditions

$$\begin{cases}x_1(t)=1, & t\in[-\tau,0],\\ x_2(t)=0, & t\in[-\tau,0],\end{cases}$$

and the exact solutions

$$\begin{cases}x_1(t)=e^t, & t\in[0,5],\\ x_2(t)=t, & t\in[0,5].\end{cases}$$

On the [0,1], [1,2] and [1,3] intervals, we use the proposed piecewise simulation to solve Eqs. (7.3) with $M=10$, $N=9$, $M=70$, $N=9$ and $M=460$, $N=9$. On the [3,4] and [4,5] intervals, we take $M=380$, $N=9$ and $M=333$, $N=9$ respectively for piecewise simulation. In order to compare the errors more easily in each subinterval, the mean relative error of Eqs. (7.3) in each subinterval is given in Table 4.

Table 4 shows that as the number of discontinuities increases, we can change the number of sample points, and our proposed method can still be effectively implemented.

8 Conclusions

In this paper, we propose a single hidden layer Ledendre neural network combined with ELM algorithm to solve linear variable coefficients DDAEs with weak discontinuities.

The convergence results of the proposed method are given and numerical experiments demonstrate the effectiveness and superiority of the piecewise approximation method to deal with DDAEs with weak discontinuities, while the global approximation is not suitable for solving the equations with weak discontinuities. If the solutions of DDAEs are smooth, the piecewise approximation method and the global approximation both can be applied to solve such kind of equations. In the future, the issue of error accumulation will be the focus of our subsequent research.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No. 11971412), the Natural Science Foundation of Hunan Province of China (No. 2018JJ2378) and Scientific Research Fund of Hunan Provincial Science and Technology Department (No. 2018WK4006).

References

[1] L. F. SHAMPINE, AND P. GAHINET, *Delay differential-algebraic equations in control theory*, Appl. Numer. Math., 56(3) (2006), pp. 574–588.
[2] G. ZHANG, L. ZHU, AND B. CHEN, *Hopf bifurcation in a delayed differential-algebraic biological economic system*, Nonlinear Anal. Real World Appl., 3 (2011), pp. 1708–1719.
[3] P. HA, *Analysis and numerical solutions of delay differential-algebraic equations*, Technische Universität Berlin, Fakultät II–Mathematik und Naturwissenschaften, 2015.
[4] L. R. PETZOLD, *Differential-algebraic equations are not ODE's*, SIAM J. Sci. Statist. Comput., 3(3) (1982), pp. 367–384.
[5] U. M. ASCHER, AND L. R. PETZOLD, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equation, Philadelphia: Society for Industrial and Applied Mathematics, 1998.
[6] A. BELLEN, AND M. ZENNARO, Numerical Methods for Delay Differential Equations, United Kingdom: Oxford University Press, 2013.
[7] T. S. CHIHARA, An Introduction to Orthogonal Polynomials, New York: Gordon and Breach, 1978.
[8] N. H. DU, V. H. LINH, AND V. MEHRMANN ET AL., *Stability and robust stability of linear time-invariant delay differential-algebraic equations*, SIAM J. Matrix Anal. Appl., 34(4) (2013), pp. 1631–1654.
[9] B. UNGER, *Discontinuity propagation in delay differential-algebraic equations*, Electron. J. Linear Al., 34(1) (2018), pp. 582–601.
[10] P. HA, AND V. MEHRMANN, *Analysis and numerical solution of linear delay differential-algebraic equations*, BIT Numer. Math., 56(2)(2016), pp. 633–657.
[11] N. H. SAU, V. N. PHAT, AND P. NIAMSUP ET AL., *On finite-time stability of linear positive differential-algebraic delay equations*, IEEE T. Circuits-II, 65(12) (2018), pp. 1984–1987.
[12] W. ZHU, AND L. R. PETZOLD, *Asymptotic stability of hessenberg delay differential-algebraic equations of retarded or neutral type*, Appl. Numer. Math., 27(3) (1998), pp. 309–325.

[13] P. HA, *Spectral characterizations of solvability and stability for delay differential-algebraic equations*, Acta Math. Vietnam., 43(4) (2018), pp. 715–735.
[14] X. HU, Y. CONG, AND G. HU, *Delay-dependent stability of Runge-Kutta methods for linear delay differential-algebraic equations*, J. Comput. Appl. Math., 363 (2019), pp. 300–311.
[15] H. TIAN, Q. YU, AND J. KUANG, *Asymptotic stability of linear neutral delay differential-algebraic equations and Runge-Kutta methods*, SIAM J. Numer. Anal., 52(1) (2014), pp. 68–82.
[16] Q. YU, H. TIAN, AND J. KUANG, *Asymptotic stability of general linear methods for systems of linear neutral delay differential-algebraic equations*, Int. J. Comput. Math., 92(4) (2015), pp. 816–835.
[17] C. ZHANG, AND H. CHEN, *Asymptotic stability of block boundary value methods for delay differential-algebraic equations*, Math. Comput. Simulat., 81(1) (2010), pp. 100–108.
[18] X. HU, Y. CONG, AND G. HU, *Delay-dependent stability of linear multistep methods for DAEs with multiple delays*, Numer. Algorithms, 79(3) (2018), pp. 719–739.
[19] H. CHEN, AND C. ZHANG, *Stability analysis of linear multistep and Runge-Kutta methods for neutral multidelay differential-algebraic systems*, Math. Comput. Model., 55(3-4) (2012), pp. 530–537.
[20] V. DESHMUKH, *Approximate stability analysis and computation of solutions of nonlinear delay differential-algebraic equations with time periodic coefficients*, J. Vib. Control, 16(7-8) (2010), pp. 1235–1260.
[21] Y. LI, L. SUN, AND Q. YU, *Stability of two-step Runge-Kutta methods for neutral delay differential-algebraic equations*, Int. J. Comput. Math., 88(2) (2011), pp. 375–383.
[22] T. QIN, AND C. ZHANG, *A general class of one-step approximation for index-1 stochastic delay differential-algebraic equations*, J. Comput. Math., 37(2) (2019), pp. 151–169.
[23] R. HAUBER, *Numerical treatment of retarded differential-algebraic equations by collocation methods*, Adv. Comput. Math., 7(4) (1997), pp. 573–592.
[24] H. LIU, AND A. XIAO, *Convergence of linear multistep methods and one-leg methods for index-2 differential-algebraic equations with a variable delay*, Adv. Appl. Math. Mech., 4(5) (2012), pp. 636–646.
[25] T. LUZYANINA, AND D. ROOSE, *Periodic solutions of differential-algebraic equations with time delays: computation and stability analysis*, Int. J. Bifurcat. Chaos, 16(01) (2006), pp. 67–84.
[26] L. SUN, Y. CONG, AND J. KUANG, *Asymptotic behavior of nonlinear delay differential-algebraic equations and implicit Euler methods*, Appl. Math. Comput., 228 (2014), pp. 395–403.
[27] L. R. PETZOLD, *The numerical solution of delay differential-algebraic equations of retarded and neutral type*, SIAM J. Numer. Anal., 32(5) (1995), pp. 1635–1657.
[28] V. H. LINH, N. D. TRUONG, AND M. V. BULATOV, *Convergence analysis of linear multistep methods for a class of delay differential-algebraic equations*, Vestnik YuUrGU. Ser. Mat. Model. Progr., 11(4) (2018), pp. 78–93.
[29] R. O. SHELTON, J. A. DARSEY, AND B. G. SUMPTER ET AL., *Neural network error correction for solving coupled ordinary differential equations*, IEEE, International Joint Conference on Neural Networks, 1992.
[30] V. DUA, *An artificial neural network approximation based decomposition approach for parameter estimation of system of ordinary differential equations*, Comput. Chem. Eng., 35(3) (2011), pp. 545–553.
[31] S. MALL, AND S. CHAKRAVERTY, *Application of legendre neural network for solving ordinary differential equations*, Appl. Soft Comput., 43 (2016), pp. 347–356.
[32] L. XU, H. WEN, AND Z. ZENG, *The algorithm of neural networks on the initial value problems in ordinary differential equations*, IEEE, Conference on Industrial Electronics and Applications, 00

(2007), pp. 813–816.
[33] J. LI, S. LUO, AND Y. QI ET AL., *Numerical solution of differential equations by radial basisfunction neural networks*, IEEE, International Joint Conference on Neural Networks, 2002.
[34] F. BOTELHO, AND J. E. JAMISON, *Qualitative behavior of differential equations associated with artificial neural networks*, J. Dyn. Differ. Equ., 16(1) (2004), pp. 179–204.
[35] A. J. J. MEADE, AND A. A. FERNANDEZ, *Solution of nonlinear ordinary differential equations by feedforward neural networks*, Math. Comput. Model., 20(9) (1994), pp. 19–44.
[36] L. ZHONG, C. GUO, AND H. SONG, *Neural network based parameters whiten method study of gray differential equations*, IEEE, ISECS International Colloquium on Computing, Communication, Control, and Management, 1 (2008), pp. 42–46.
[37] P. RAMUHALLI, L. UDPA, AND S. S. UDPA ET AL., *Finite-element neural networks for solving differential equations*, IEEE T. Neural Networ., 16(6) (2005), pp. 1381–1392.
[38] L. ZJAVKA, AND W. PEDRYCZ, *Constructing general partial differential equations using polynomial and neural networks*, Neural Netw., 00 (2016), pp. 58–69.
[39] D. GOBOVIC, AND M. E. ZAGHLOUL, *Analog cellular neural network with application to partial differential equations with variable mesh-size*, IEEE, International Symposium on Circuits and Systems-ISCAS, 6 (1994), pp. 359–362.
[40] K. RUDD, AND S. FERRARI, *A constrained integration (CINT) approach to solving partial differential equations using artificial neural networks*, Neurocomputing, 155 (2015), pp. 277–285.
[41] M. J. AEIN, AND H. A. TALEBI, *Introducing a training methodology for cellular neural networks solving partial differential equations*, IEEE, International Joint Conference on Neural Networks, 00 (2009), pp. 71–75.
[42] K. S. MCFALL, *Automated design parameter selection for neural networks solving coupled partial differential equations with discontinuities*, J. Franklin I., 350(2) (2013), pp. 300–317.
[43] J. BERG, AND K. NYSTRÖM, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, 317 (2018), pp. 28–41.
[44] I. E. LAGARIS, A. LIKAS, AND D. I. FOTIADIS ET AL., *Artificial neural networks for solving ordinary and partial differential equations*, IEEE T. Neural Networ., 9(5) (1998), pp. 987–1000.
[45] J. FANG, C. LIU AND T. E. SIMOS ET AL., *Neural network solution of single-delay differential equations,* Mediterr. J. Math., 17(30) (2020), `https://doi.org/10.1007/s00009-019-1452-5`.
[46] B. KRASZNAI, I. GYŐ RI, AND M. PITUK, *The modified chain method for a class of delay differential equations arising in neural networks*, Math. Comput. Model., 51(5-6) (2010), pp. 452–460.
[47] T. V. LAZOVSKAYA, AND D. A TARKHOV, *Fresh approaches to the construction of parameterized neural network solutions of a stiff differential equation*, St Petersburg Polytechnical University Journal Physics & Mathematics, 1(2) (2015), pp. 192–198.
[48] J. C. CHEDJOU, K. KYAMAKYA, AND M. A. LATIF ET AL., *Solving stiff ordinary differential equations and partial differential equations using analog computing based on cellular neural networks*, IEEE, International Workshop on Nonlinear Dynamics and Synchronization, 00 (2009), pp. 213–220.
[49] Z. XIE, D. KULASIRI, AND S. SAMARASINGHE ET AL., *The estimation of parameters for stochastic differential equations using neural networks*, Inverse Probl. Sci. Eng., 15(6) (2007), pp. 629–641.
[50] F. ROSTAMI, AND A. JAFARIAN, *A new artificial neural network structure for solving high-order linear fractional differential equations*, Int. J. Comput. Math., 95(3) (2018), pp. 528–539.
[51] M. PAKDAMAN, A. AHMADIAN, AND S. EFFATI ET AL., *Solving differential equations of fractional order using an optimization technique based on training artificial neural network*, Appl. Math. Comput., 293 (2017), pp. 81–95.
[52] C. J. ZÚÑIGAAGUILAR, A. CORONELESCAMILLA, AND J. F. GÓMEZAGUILAR ET AL., *New*

numerical approximation for solving fractional delay differential equations of variable order using artificial neural networks, Eur. Phys. J. Plus, 133(2) (2018), pp. 75–91.

[53] D. S. KOZLOV, AND Y. V. TIUMENTSEV, *Neural network based semi-empirical models for dynamical systems represented by differential-algebraic equations of index 2*, Procedia Computer Science, 123 (2018), pp. 252–257.

[54] Z. YANG, J. LAN, AND Y. J. WU, *On solutions to several classes of differential-algebraic equations based on artificial neural networks*, Appl. Math. Mech., 40(02) (2019), pp. 5–16 (in Chinese).

[55] S. CHAKRAVERTY, AND S. MALL, Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations, Boca Raton: CRC Press, 2017.

[56] N. YADAV, A. YADAV, AND M. KUMAR, An Introduction to Neural Network Methods for Differential Equations , Netherlands: Springer, 2015.

[57] C. WANG, Z. WANG, AND H. JIA, *An hp-version spectral collocation method for nonlinear volterra integro-differential equation with weakly singular kernels*, J. Sci. Comput., 72(2) (2017), pp. 647–678.

[58] 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. Differential Equations, 1 (2018), Article Number: 469.

[59] G. HUANG, Q. ZHU, AND C. K. SIEW, *Extreme learning machine: Theory and applications*, Neurocomputing, 70(1-3) (2006), pp. 489–501.

[60] H. SUN, M. HOU, AND Y. YANG ET AL., *Solving partial differential equation based on bernstein neural network and extreme learning machine algorithm*, Neural Process. Lett., 00 (2018), pp. 1–20.

[61] H. LIU, A. XIAO, AND Y. ZHAO, *Variational iteration method for delay differential-algebraic equations*, Math. Comput. Appl., 15(5) (2010), pp. 834–839.