

## Scientific Reports

Detail	Information
Title	Scientific Reports
Publisher	Nature Portfolio
Scope	Broad range of scientific disciplines (biology, chemistry, physics, mathematics, earth sciences, social sciences)
Frequency	Continuous publication (articles published online as soon as they are ready)
Website	<a href="https://www.nature.com/srep/">https://www.nature.com/srep/</a>
Print ISSN	2045-2322
Electronic ISSN	2045-2322
Impact Factor	4.375 (as of 2022 Journal Citation Reports)
Ranking	Middle range among multidisciplinary science journals
Peer Review Process	Single-blind (reviewers know the authors' identities)
Open Access Policy	Open access (articles are freely available; APCs required)
Article Types	Research Articles, Technical Reports, Data Descriptors
Editor-in-Chief	Various leading researchers (specific names may change)
Submission Process	Online submission via the journal's submission system
Publication Process	Articles published online with DOI
Citation and Metrics	Citations tracked via Google Scholar, Web of Science; Altmetric Score available
Ethics Statement	Adheres to strict ethical guidelines (disclosure of conflicts of interest, ethical approval for research, integrity standards)

# OPEN Numerical solution of neutral delay differential equations using orthogonal neural network

Chavda Divyesh Vinodbhai & Shruti Dubey<sup>1,2</sup>

In this paper, an efficient orthogonal neural network (ONN) approach is introduced to solve the higher-order neutral delay differential equations (NDDEs) with variable coefficients and multiple delays. The method is implemented by replacing the hidden layer of the feed-forward neural network with the orthogonal polynomial-based functional expansion block, and the corresponding weights of the network are obtained using an extreme learning machine (ELM) approach. Starting with simple delay differential equations (DDEs), an interest has been shown in solving NDDEs and system of NDDEs. Interest is given to consistency and convergence analysis, and it is seen that the method can produce a uniform closed-form solution with an error of order  $2^{-n}$ , where  $n$  is the number of neurons. The developed neural network method is validated over various types of example problems (DDEs, NDDEs, and system of NDDEs) with four different types of special orthogonal polynomials.

Delay differential equation (DDE) plays a crucial role in epidemiology, population growth, and many mathematical modeling problems. In DDEs, the dependent variable depends not only on its current state but also on a specific past state. One type of DDE in which time delays are included in the state derivative is called the neutral delay differential equation (NDDE). Delay terms are classified into three types: discrete, continuous, and proportional delay. In this paper, we are focusing on proportional DDEs and NDDEs. One famous example of proportional delay differential equations is the pantograph differential equation which was first introduced in<sup>1</sup>.

Generally, the exact solution of delay differential equations is complicated to find, and due to the model's complexity, many DDEs do not have an exact solution. Various numerical schemes have been developed over the years to find the approximate solution of delay differential equations. There are several articles<sup>2–9</sup> that illustrate some exact and numerical methods for approximate solutions of DDEs and NDDEs.

Artificial neural networks (ANNs) have been utilised to produce an approximate solution of differential equations for the past 22 years. A neural network approach for several ordinary and partial differential equations was first proposed by Lagaris et al. in<sup>10</sup>. The approximate solution delivered by the artificial neural networks has a variety of advantages: (i) The derived approximation of the solution is in closed analytic form. (ii) The generalization ability of an approximation is excellent. (iii) Discretization of derivatives is not required. Many articles on approximation artificial neural network solutions to different differential equations are available in the literature<sup>11–20</sup>. As far as we know, the studies for obtaining an approximate solution to delay differential equations using artificial neural networks are limited. There is very little literature available for solving delay differential equations using ANNs. J. Fang et al. solved first-order delay differential equations with single delay using ANN<sup>21</sup>. In<sup>22</sup>, Chih-Chun Houe et al. obtained approximate solutions of proportional delay differential equation using ANN. All these artificial neural network approaches suffer from common problems: (1) All the algorithms are time-consuming and therefore they are computationally expensive numerical optimization algorithms, (2) They completely depend on the trial solution, which is difficult to construct for higher dimensional problems. Recently in<sup>23</sup>, Manoj and Shagun obtained an approximate solution of differential equations using an optimization-free neural network approach in which they trained the network weights using ELM algorithm<sup>24</sup>. In<sup>25</sup>, authors solved the first-order pantograph equation using the optimization-free ANN approach. Linear first-order delay differential-algebraic equations have been solved using Legendre neural network in<sup>26</sup>.

This work presents an orthogonal neural network with an extreme learning machine algorithm (ONN-ELM) to obtain an approximate solution for higher-order delay differential equations, neutral delay differential equations, and a system with multiple delays and variable coefficients. The ONN model is a particular functional link neural network (FLNN)<sup>12,27–29</sup> case. It has the advantage of fast and very accurate learning. The entire procedure becomes much quicker than a traditional neural network because it removes the high-cost iteration procedure

Department of Mathematics, Indian Institute of Technology Madras, Chennai, Tamil Nadu 600036, India. <sup>2</sup>email: sdubey@iitm.ac.in



and trains the network weights using the Moore-Penrose generalized inverse of the proposed approach:

- It is a single hidden layer neural network, we only need to train the output layer weights by randomly selecting the input layer weights.
- We use an unsupervised extreme learning machine algorithm to train the output weights; no optimization technique is used in this procedure.
- It is simple to implement, accurate compared to other numerical schemes mentioned in the literature, and runs quickly.

This work considers four different orthogonal polynomials-based neural networks: (i) Legendre neural network, (ii) Hermite neural network, (iii) Laguerre neural network, and (iv) Chebyshev neural network with ELM for solving DDEs, NDDEs, and systems of NDDEs with multiple delays and variable coefficients. The interest is to find the orthogonal neural network among these four that can produce more accurate solution.

The layout of this paper is as follows. In "Preliminaries" section, we present some definitions and properties of orthogonal polynomials and a description of the considered problems. In "Orthogonal neural network" section, we describe the architecture of the orthogonal neural network (ONN) with an extreme learning algorithm (ELM). "Error analysis" section discusses the convergence analysis and error analysis. The methodology of the proposed method is presented in "Methodology" section. Various numerical illustrations are presented in "Numerical illustrations" section and a comparative study is given in "Comparative analysis" section.

## Preliminaries

In this section, first, we introduce basic definitions and some properties of the orthogonal polynomials. Throughout the paper, we will use  $P_n(x)$  to represent the orthogonal polynomial of order  $n$ .

### Orthogonal polynomial.

**Definition 1** The orthogonal polynomials are special class of polynomials  $P_n(x)$  defined on  $[a, b]$  that follow an orthogonality relation as,

$$\int_a^b g(x) P_m(x) P_n(x) dx = \delta_{m,n} k_n,$$

where  $n, m \in \mathbb{N}$ ,  $\delta_{m,n}$  is Kronecker delta,  $g(x)$  is a weight function and  $k_n = \int_a^b g(x) [P_n(x)]^2 dx$ .

### Remark

1. If a weight function  $g(x) = 1$ , then the orthogonal polynomial  $P_n(x)$  is called Legendre polynomial.
2. If a weight function  $g(x) = (1 - x^2)^{-1/2}$ , then the orthogonal polynomial  $P_n(x)$  is called Chebyshev polynomial of first kind.
3. If a weight function  $g(x) = e^{-x^2}$ , then the orthogonal polynomial  $P_n(x)$  is called Hermite polynomial.
4. If a weight function  $g(x) = e^{-x}$ , then the orthogonal polynomial  $P_n(x)$  is called Laguerre polynomial.

**Properties of orthogonal polynomials.** The following are some of the remarkable properties of a set of orthogonal polynomials:

- Each polynomial  $P_n(t)$  is orthogonal to any other polynomial of degree  $< n$  in a set of orthogonal polynomials  $\{P_0(t), \dots, P_n(t), \dots\}$ .
- Any set of orthogonal polynomials has a recurrence formula that connects any three consecutive polynomials in the sequence, i.e., the relation  $P_{n+1}(t) = (a_n t + b_n) P_n(t) - c_n P_{n-1}(t)$  exists, with constants  $a_n, b_n, c_n$  depending on  $n$ .
- The zeroes of orthogonal polynomials are real numbers.
- There is always a zero of orthogonal polynomial  $P_{n+1}(t)$  between two zeroes of  $P_n(t)$ .

**Moore-Penrose generalized inverse.** In this section, the Moore-Penrose generalized inverse is introduced.

There can be problems in obtaining the solution of a general linear system  $Ax = y$ , where  $A$  may be a singular matrix or may even not be square. The Moore-Penrose generalized inverse can be used to solve such difficulties. The term generalized inverse is sometimes referred to as a synonym of pseudoinverse. More precisely, we define the Moore-Penrose generalized inverse as follows:

**Definition 2** <sup>30</sup> A matrix  $B$  of order  $n \times m$  is the Moore-Penrose generalized inverse of matrix  $A$  of order  $m \times n$ , if the following hold

$$ABA = A, BAB = B, (AB)^T = AB, (BA)^T = BA,$$

where  $A^T$  denotes the transpose of matrix  $A$ . The Moore-Penrose generalized inverse of matrix  $A$  is denoted by  $A^+$ .

**Definition 3**  $x_0 \in \mathbb{R}^n$  is said to be a minimum norm least-squares solution of a general linear system  $Ax = y$  if for any  $y \in \mathbb{R}^m$

$$\|x_0\| \leq \|x\|, \forall x \in \{x : \|Ax - y\| \leq \|Az - y\|, \forall z \in \mathbb{R}^n\}$$

where  $\|\cdot\|$  is the Euclidean norm.

In other words, if a solution  $x_0$  has the smallest norm among all the least-squares solutions, it is considered to be a minimum norm least-squares solution of the general linear system  $Ax = y$ .

**Theorem 1** <sup>30</sup> Let  $B$  be a matrix with a minimum norm least-squares solution to the linear equation  $Ax = y$ . Then  $B = A^+$ , the Moore-Penrose generalized inverse of matrix  $A$ , is both required and sufficient.

**Problem definition.** In this subsection, we present the general form of the pantograph equation, higher order delay differential equation, higher order neutral delay differential equation, and the system of higher order delay differential equation with variable coefficients and multiple delays.

**The generalized Pantograph equation.** Pantograph type equation arises as a mathematical model in the study of the wave motion of the overhead supply line to an electric locomotive. The following equation gives the generalized form of a pantograph type equation with multiple delays:

$$z'(t) = a(t)z(t) + \sum_{i=1}^k b_i(t)z(q_i t) + \sum_{j=1}^l c_j(t)z'(q_j t) + g(t), \quad (1)$$

with initial conditions

$$z(t_0) = z_0, \quad (2)$$

where  $g(t)$ ,  $a(t)$ ,  $b_i(t)$  and  $c_j(t)$  is continuous function,  $0 < q_i, q_j < 1$  for some  $k, l \in \mathbb{N}$  and  $t \in [t_0, t_1]$  for some,  $t_0, t_1 \in \mathbb{R}$ .

**Higher order DDEs and NDDEs.**

- Consider the general form of Higher-order DDEs with multiple delay

$$z^k(t) = f(t, z(t), \dots, z^{k-1}(t), z(q_1 t), \dots, z(q_n t)), \quad (3)$$

with initial conditions

$$z(t_0) = z_0, z'(t_0) = z_1, \dots, z^{k-1}(t_0) = z_{k-1}, \quad (4)$$

where  $q_i \in (0, 1)$  for  $i = 1, \dots, n$  and  $z^k$  denotes the  $k$ th derivative of  $z(t)$ .

- Consider the general form of Higher-order NDDEs with multiple delay

$$z^k(t) = f(t, z(t), \dots, z^{k-1}(t), z(q_1^1 t), \dots, z(q_{n_1}^1 t), z'(q_1^2 t), \dots, z'(q_{n_2}^2 t), \dots, z^k(q_1^{k+1} t), \dots, z^k(q_{n_{k+1}}^{k+1} t)), \quad (5)$$

with initial condition

$$z(t_0) = z_0, z'(t_0) = z_1, \dots, z^{k-1}(t_0) = z_{k-1}, \quad (6)$$

where all  $q_i^j \in (0, 1)$  for  $j = 1, \dots, k+1, i = 1, \dots, n_j, n_j, k \in \mathbb{N}$  and  $z^k$  denotes the  $k$ th derivative of  $z(t)$ .

**Higher order system of DDE.** Consider the general form of higher order coupled neutral delay differential equation with multiple delays as:

$$\begin{aligned} z_1^k(t) &= f(t, z_1(t), \dots, z_1^{k-1}(t), z_2(t), \dots, z_2^k(t), z_1(q_1^1 t), \dots, z_1(q_{n_1}^1 t), z_2(p_1^1 t), \dots, \\ & z_2(p_{m_1}^1 t), z_1'(q_1^2 t), \dots, z_1'(q_{n_2}^2 t), z_2'(p_1^2 t), \dots, z_2'(p_{m_2}^2 t), \dots, z_1^k(q_1^{k+1} t), \dots, \\ & z_1^k(q_{n_{k+1}}^{k+1} t), z_2^k(p_1^{k+1} t), \dots, z_2^k(p_{m_{k+1}}^{k+1} t)), \\ z_1(t_0) &= z_0^1, z_1'(t_0) = z_1^1, \dots, z_1^{k-1}(t_0) = z_{k-1}^1, \end{aligned} \quad (7)$$



$$\begin{aligned} z_2^k(t) &= g(t, z_1(t), \dots, z_1^{k-1}(t), z_2(t), \dots, z_2^k(t), z_1(r_1^1 t), \dots, z_1(r_{h_1}^1 t), z_2(s_1^1 t), \dots, \\ & z_2(s_{h_1}^1 t), z_1'(r_1^2 t), \dots, z_1'(r_{h_1}^2 t), z_2'(s_1^2 t), \dots, z_2'(s_{h_1}^2 t), \dots, z_1^k(r_1^{k+1} t), \dots, \\ & z_1^k(r_{h_1}^{k+1} t), z_2^k(s_1^{k+1} t), \dots, z_2^k(s_{h_1}^{k+1} t)), \\ z_2(t_0) &= z_2^0, z_2'(t_0) = z_1^2, \dots, z_2^{k-1}(t_0) = z_2^{k-1}, \end{aligned}$$

where  $n_j, m_j, l_j, h_j \in \mathbb{N}$  and all  $d_{i_1}^j, p_{i_2}^j, r_{i_3}^j, s_{i_4}^j \in (0, 1)$  for  $j = 1, \dots, k+1, i_1 = 1, \dots, n_j, i_2 = 1, \dots, m_j, i_3 = 1, \dots, l_j, i_4 = 1, \dots, h_j$ .

### Orthogonal neural network

In this section, we introduce the structure of a single-layered orthogonal neural network (ONN) model with an extreme learning machine (ELM) algorithm for training the network weights.

**Structure of orthogonal neural network (ONN).** Orthogonal neural network (ONN) is a single-layered feed-forward neural network, which consists of one input neuron  $t$ , one output neuron  $N(t, a, w)$  and a hidden layer is eliminated by the orthogonal functional expansion block. The architecture of an orthogonal neural network is depicted in Fig. 1.

Consider a 1-dimensional input neuron  $t$ . The enhanced pattern is obtained by orthogonal functional expansion block as follows:

$$\{P_0(a_0 t), P_1(a_1 t), \dots, P_n(a_n t)\}.$$

Here  $N(t, a, w) = \sum_{i=0}^n w_i P_i(a_i t)$  is the output of the orthogonal neural network, where  $a_i$ 's are randomly selected fixed weights and  $w_i$ 's are the weights to be trained.

**Extreme learning machine (ELM) algorithm.** For a given sample points  $(t_j, y_j)$ ,  $t_j \in \mathbb{R}^n$  and  $y_j \in \mathbb{R}$ , for  $j = 0, 1, \dots, m$ , a single-layer feed-forward neural network with  $(n+1)$  neurons has the following output:

$$\sum_{i=0}^n w_i g_i(a_i t_j), \quad j = 0, 1, \dots, m,$$

where  $g_i$  is the activation function of  $i$ -th neuron in a hidden layer,  $a_i$ 's are the randomly selected fixed weights between the input layer and hidden layer, and  $w_i$ 's are the weights between the hidden layer and output, which need to be trained.

When the neural network completely approximates the given data, i.e., the output of the neural network and actual data are equal, the following relation hold:

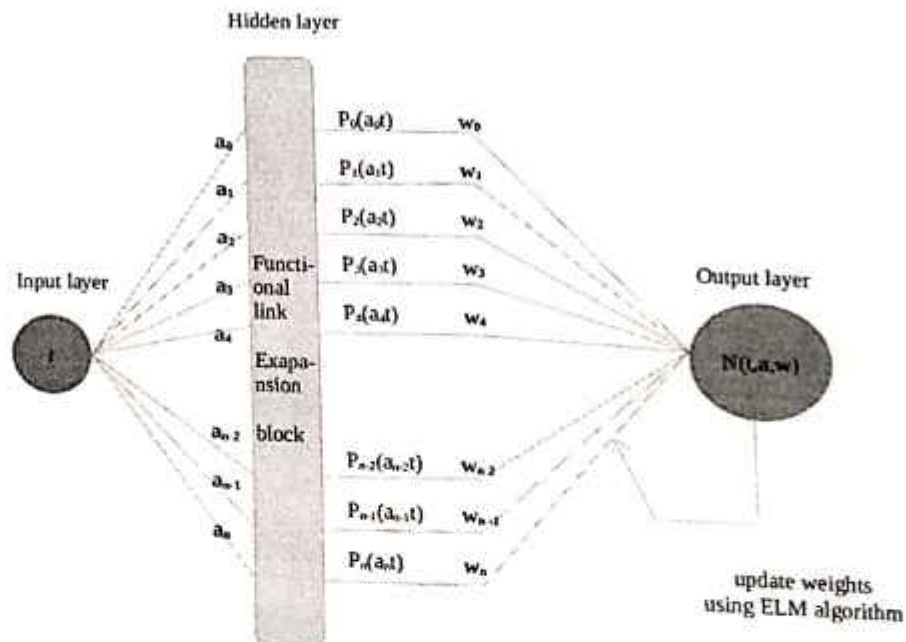


Figure 1. The structure of orthogonal neural network.

$$\sum_{i=0}^n w_i g_i(a, t_j) = y_j, \quad j = 0, 1, \dots, m. \quad (9)$$

Equation (9) can be written in matrix form as:

$$Aw = b, \quad (10)$$

where the hidden layer output matrix  $A$  is defined as follows:

$$A = \begin{pmatrix} g_0(a_0 t_0) & g_1(a_1 t_0) & \cdots & g_n(a_n t_0) \\ g_0(a_0 t_1) & g_1(a_1 t_1) & \cdots & g_n(a_n t_1) \\ \vdots & \vdots & \ddots & \vdots \\ g_0(a_0 t_m) & g_1(a_1 t_m) & \cdots & g_n(a_n t_m) \end{pmatrix}. \quad (11)$$

and  $w = [w_0, w_1, \dots, w_n]^T$ ,  $b = [y_0, y_1, \dots, y_m]^T$ .

For the given training points  $t_j \in \mathbb{R}^n$  and the weights  $a_j$ 's, the matrix  $A$  can be calculated and the weights  $w_j$ 's can be calculated by solving the linear system  $Aw = b$ .

**Theorem 2** The system  $Aw = b$  is solvable in the following several cases:

1. If  $A$  is a square matrix, then  $w = A^{-1}b$
2. If  $A$  is a rectangular matrix, then  $w = A^+b$ , and  $w$  is the minimal least square solution of  $Aw = b$ . Here  $A^+$  is a pseudo inverse of  $A$ .
3. If  $A$  is a singular matrix, then  $w = A^+b$  and  $A^+ = A^T(\lambda I + AA^T)^{-1}$ , where  $\lambda$  is the regularization coefficient. We can set a value of  $\lambda$  according to the specific instance.

### Error analysis

This section will discuss the convergence result and error analysis of the ONN-ELM method for solving the delay and neutral delay differential equations.

**Theorem 3** <sup>24</sup> Let single layer feed-forward orthogonal neural network  $N(t, a, w)$  be an approximate solution of one-dimensional neutral delay differential equation, for  $m+1$  arbitrary distinct sample points  $(t_j, y_j)$  for  $j = 0, 1, \dots, m$ , where  $t_j, y_j \in \mathbb{R}$ , then the orthogonal expansion layer output matrix  $A$  is invertible, and  $\|Aw - b\| = 0$ .

**Theorem 4** Let  $z \in C^\infty(t_0, t_m)$ ,  $\hat{z}_n = N(t, a, w)$  be the orthogonal neural network with  $n$  neurons in the hidden layer and  $e_n$  be the absolute error with  $n$  hidden neurons, then  $\|e_n\| \rightarrow 0$  as  $n \rightarrow \infty$ .

**Proof** The Taylor expansion formula gives us the following expression for  $z(t)$  on  $(t_0, t_m)$ :

$$z(t) = z(t_0^+) + z'(t_0^+)(t - t_0) + \frac{z''(t_0^+)}{2!}(t - t_0)^2 + \dots + \frac{z^{(n)}(c)}{n!}(t - t_0)^n, \quad c \in (t_0, t_1). \quad (12)$$

Let us define  $z_n(t) = \sum_{i=0}^{n-1} \frac{z^{(i)}(t_0^+)}{i!}(t - t_0)^i$ , then we get

$$\|z(t) - z_n(t)\| = \frac{1}{n!} \|z^{(n)}(c)(t - t_0)^n\|. \quad (13)$$

Let  $L = \text{span}\{P_0(t), P_1(t), \dots, P_n(t)\}$  and let  $\hat{z}_n(t)$  be the best approximation of  $z(t)$  in  $L$  given as,  $\hat{z}_n(t) = \sum_{i=0}^{n-1} w_i P_i(a, t)$ , where  $w_i$ 's are the weights obtained by ELM algorithm. we get

$$\|z(t) - \hat{z}_n(t)\| \leq \|z(t) - z_n(t)\|, \quad \forall z(t) \in L. \quad (14)$$

In particular, taking  $\hat{z}(t) = z_n(t)$  we have

$$\begin{aligned} \|e_n(t)\| &= \|z(t) - \hat{z}_n(t)\| \\ &\leq \|z(t) - z_n(t)\| \\ &= \frac{1}{n!} \|z^{(n)}(c)(t - t_0)^n\| \end{aligned} \quad (15)$$

Thus,

$$\begin{aligned} \|e_n(t)\| &\leq \left\| \frac{z^{(n)}(c)}{n!} (t - t_0)^n \right\| \\ &\leq \frac{M}{2^n}, \end{aligned} \quad (16)$$

where,  $M = \max \|z^{(n)}(c)(t - t_0)^n\|$ , for  $t \in (t_0, t_m)$ .

Moreover, from Eq. (16) we deduce that  $\|e_n(t)\| \rightarrow 0$  for large value of  $n$ . This shows that ONN has good representational abilities and it can approximate the exact solution with almost no error.

## Methodology

This section explains the method to obtain an approximate solution of second-order NDDE using the ONN-ELM algorithm. It can be easily extended to the higher-order NDDE and the higher-order DDE is a special case of the higher-order NDDE.

Consider the general form of linear second-order NDDE

$$z''(t) + a(t)z'(t) + b(t)z(t) + \sum_{j=1}^{m_1} c_j(t)z(\alpha_j t) + \sum_{k=1}^{m_2} d_k(t)z'(\beta_k t) + \sum_{l=1}^{m_3} e_l(t)z''(\gamma_l t) = f(t), \quad t \in (a, b), \quad (17)$$

with initial condition  $z(a) = z_0$  and  $z'(a) = z_1$  or boundary condition  $z(a) = z_2$  and  $z(b) = z_3$ , where  $z_0, z_1, z_2, z_3 \in \mathbb{R}$ ,  $a(t), b(t), c_j(t), d_k(t), e_l(t), f(t)$  are continuously differentiable function for  $t \in (a, b)$  and  $m_1, m_2, m_3 \in \mathbb{N}$ .

Using ONN-ELM with  $n$  neurons, an approximate solution of Eq. (17) is obtained in the form:

$$\tilde{z}_n(t) = \sum_{i=0}^n w_i P_i(t), \quad (18)$$

where  $w_i$ 's are the output weights that need to be trained and  $P_i(t)$  is the  $i$ -th orthogonal polynomial.

Since the approximate solution obtained by the ONN-ELM algorithm is the linear combination of the orthogonal polynomials, it is infinitely differentiable and we have,

$$\tilde{z}'_n(t) = \sum_{i=0}^n w_i P'_i(t), \quad (19)$$

$$\tilde{z}''_n(t) = \sum_{i=0}^n w_i P''_i(t), \quad (20)$$

$$\sum_{j=1}^{m_1} \tilde{z}_n(\alpha_j t) = \sum_{j=1}^{m_1} \sum_{i=0}^n w_i P_i(\alpha_j t), \quad (21)$$

$$\sum_{k=1}^{m_2} \tilde{z}'_n(\beta_k t) = \sum_{k=1}^{m_2} \sum_{i=0}^n \beta_k w_i P'_i(\beta_k t), \quad (22)$$

$$\sum_{l=1}^{m_3} \tilde{z}''_n(\gamma_l t) = \sum_{l=1}^{m_3} \sum_{i=0}^n \gamma_l^2 w_i P''_i(\gamma_l t). \quad (23)$$

Substituting Eqs. (18)–(23) into the second order neutral delay differential equation (17), we have

$$\begin{aligned} & \sum_{i=0}^n w_i P''_i(t) + a(t) \sum_{i=0}^n w_i P'_i(t) + b(t) \sum_{i=0}^n w_i P_i(t) + \sum_{j=1}^{m_1} w_i \sum_{i=0}^{m_1} c_j(t) P_i(\alpha_j t) \\ & + \sum_{i=0}^n w_i \sum_{k=1}^{m_2} \beta_k d_k(t) P'_i(\beta_k t) + \sum_{i=0}^n w_i \sum_{l=1}^{m_3} \gamma_l^2 e_l(t) P''_i(\gamma_l t) = f(t). \end{aligned} \quad (24)$$

We can write Eq. (24) as:

$$\sum_{i=0}^n w_i A_i(t) = f(t), \quad (25)$$

where,



$$A_i = P_i''(t) + a(t)P_i(t) + b(t)P_i(t) + \sum_{j=1}^{m_1} c_j(t)P_i(\alpha_j t) + \sum_{k=1}^{m_2} \beta_k d_k(t)P_i'(\beta_k t) + \sum_{l=1}^{m_3} \gamma_l^2 e_l(t)P_i'(\gamma_l t).$$

Using the discretization of interval  $[a, b]$  as  $a = t_0 < t_1 < \dots < t_m = b$  for  $m \in \mathbb{N}$ , define  $f_m = f(t_m)$ . At these discretized points, Eq. (25) is to be satisfied, that is:

$$\sum_{i=0}^n w_i A_i(t_m) = f(t_m), \quad \forall m \in \mathbb{N}. \quad (26)$$

Equation (26) can be written as a system of equations as:

$$A_1 w = b_1,$$

where  $w = [w_0, w_1, \dots, w_n]^T$ ,

$$A_1 = \begin{pmatrix} A_0(t_0) & A_1(t_0) & \dots & A_n(t_0) \\ A_0(t_1) & A_1(t_1) & \dots & A_n(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ A_0(t_m) & A_1(t_m) & \dots & A_n(t_m) \end{pmatrix},$$

and  $b_1 = [f(t_0), f(t_1), \dots, f(t_m)]^T$ .

Case:1 Consider Eq. (17) with the initial conditions. Then the following linear system is obtained:

$$\underbrace{\begin{pmatrix} A_0(t_0) & A_1(t_0) & \dots & A_n(t_0) \\ A_0(t_1) & A_1(t_1) & \dots & A_n(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ A_0(t_m) & A_1(t_m) & \dots & A_n(t_m) \\ P_0(a) & P_1(a) & \dots & P_n(a) \\ P_0'(a) & P_1'(a) & \dots & P_n'(a) \end{pmatrix}}_A \underbrace{\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}}_w \approx \underbrace{\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_m \\ z_0 \\ z_1 \end{pmatrix}}_b$$

Case:2 Consider Eq. (17) with the boundary conditions. Then the following linear system for NDDE is obtained:

$$\underbrace{\begin{pmatrix} A_0(t_0) & A_1(t_0) & \dots & A_n(t_0) \\ A_0(t_1) & A_1(t_1) & \dots & A_n(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ A_0(t_m) & A_1(t_m) & \dots & A_n(t_m) \\ P_0(a) & P_1(a) & \dots & P_n(a) \\ P_0(b) & P_1(b) & \dots & P_n(b) \end{pmatrix}}_A \underbrace{\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}}_w \approx \underbrace{\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_m \\ z_2 \\ z_3 \end{pmatrix}}_b$$

To calculate the weight vector  $w$  of the network, we use the extreme learning algorithm, that is:

$$w = A^+ b, \quad (27)$$

where  $A^+ = (A^T A)^{-1} A^T$  is the least square solution of Eq. (27).

Note: Similar methodology can be used for the higher order neutral delay differential equation and the system of higher order neutral delay differential equations.



**Algorithm 1** Pseudocode for solution of NDDE using ONN and ELM algorithm

**Require:**

- Generate the set of training points  $t_j$ 's from the problem domain  $D$ .
- Define the orthogonal polynomial  $P(n, t)$ , where  $P(n, t)$  is orthogonal polynomial of degree  $n$ .
- Define the derivatives of the orthogonal polynomial as  $DP(n, t)$ ,  $DDP(n, t)$ ,  $DDDP(n, t)$ , ...

**procedure**

for  $j \leq m$  do       $\triangleright$  Construct a residual matrix of given NDDE as  
for  $i \leq n$  do

$$A(i, j) = F(t_j, P(i, t_j), DP(i, t_j), \dots, P(i, \omega_1 t_j), DP(i, \omega_1 t_j), \dots)$$

end for

end for

- Add a row in the residual matrix corresponding to each initial/boundary condition.

for  $j \leq m$  do       $\triangleright$  Compute the form term vector of given NDDE as

$$b(j) = f(t_j)$$

end for

- Add initial/boundary values in vector  $b(j)$ .
- Write the obtained matrix system in the form

$$A \cdot X = b,$$

where  $X = [X_1, X_2, \dots, X_n]$

- Using the Moore-Penrose generalized inverse method, calculate the value of  $X$ .

**end procedure**

### • Steps of solving NDDEs using an ONN-ELM algorithm:

1. Discretize the domain as  $a = t_0 < t_1 < t_2 < \dots < t_m = b$ .
2. Construct the approximate solution by using the orthogonal polynomial as an activation function that is,

$$N(t, \mathbf{w}) = \sum_{i=0}^n w_i P_i(a, t),$$

where  $w_i$ 's are the randomly generated fixed weights.

3. At the discrete points, substitute the approximate solution and its derivatives into the differential equation and its boundary conditions and obtain the system of equations  $A\mathbf{w} = \mathbf{b}$ .
4. Solve the system of equations  $A\mathbf{w} = \mathbf{b}$  by ELM algorithm and obtain the network weights  $w_i$ 's.
5. Substitute the value of  $w_i$ 's and get an approximate solution of DDE.

### Numerical illustrations

This section considers the higher order delay and neutral delay differential equations with multiple delays and variable coefficients. We also consider the system of delay and neutral delay differential equations. In all the test examples, we use the special orthogonal polynomials based neural network like Legendre neural network, Laguerre neural network, Chebyshev neural network, and Hermite neural network. Further, to show the reliability and powerfulness of the presented method, we compare the approximate solutions with the exact solution. All computations are carried out using Python 3.9.7 on Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz and the Windows 10 operating system. We calculate the relative error which is defined as follows.

$$\text{Relative error} = \left\| \frac{\text{exact solution} - \text{numerical solution}}{\text{exact solution}} \right\|$$

**Example 6.1** Consider the second-order boundary valued proportional delay differential equation with variable coefficients

$$z''(t) = 0.5z(t) + e^{0.5t} z\left(\frac{t}{2}\right) - 2e^{-t},$$

$$z(0) = 0, \quad z(1) = e^{-1}$$

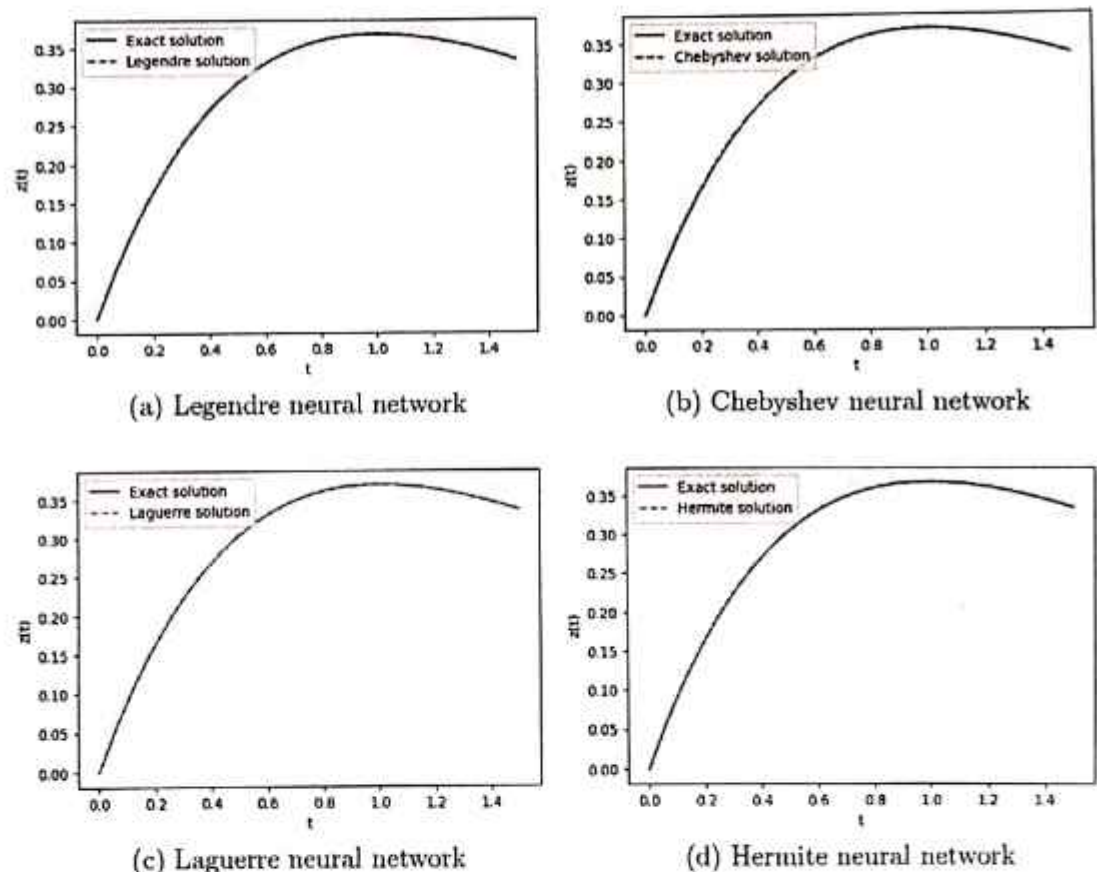
The exact solution of the given equation is  $te^{-t}$ .

We employ four ONNs to obtain the approximate solution of the given second-order DDE with variable coefficients. We choose ten uniformly distributed points in  $[0, 1]$ . The relative errors for all ONNs are shown in Fig. 3. Obtained relative errors for different orthogonal neural networks are reported in Table 1, and we compare the approximate solutions with the exact solution in Fig. 2.

Table 1 and Fig. 3 clearly show that the Chebyshev polynomial-based ONN performs best with the maximum relative error  $5.61 \times 10^{-8}$ . Table 2 shows the comparison of the maximum relative error for Example 6.1 using the Legendre, Laguerre, Hermite, and Chebyshev neural networks with various numbers of neurons ( $n = 5, 8$ , and  $11$ ) and their respective computational time. Additionally, Table 2 shows that all four neural networks satisfy Theorem 4, and for  $n = 5$ , all four orthogonal neural networks show similar accuracy. However, Chebyshev neural network performs better with  $n = 8, 11$ .

$t$	Legendre neural network	Hermite neural network	Laguerre neural network	Chebyshev neural network
0.1	1.56e-08	2.26e-08	1.39e-08	5.61e-08
0.2	6.20e-08	5.89e-08	6.49e-08	4.54e-08
0.3	4.57e-08	4.39e-08	4.89e-08	4.12e-08
0.4	1.07e-08	9.71e-09	1.40e-08	2.09e-08
0.5	1.94e-08	1.88e-08	2.26e-08	4.57e-08
0.6	5.66e-08	5.64e-08	5.98e-08	1.06e-08
0.7	6.84e-08	6.86e-08	7.17e-08	2.73e-08
0.8	5.09e-08	5.14e-08	5.42e-08	1.32e-08
0.9	6.99e-08	7.08e-08	7.34e-08	1.79e-08
1	7.08e-10	4.55e-10	2.93e-09	4.63e-09

**Table 1.** The relative error for Example 6.1 with different orthogonal neural networks.



**Figure 2.** Comparison of the exact solution with the obtained approximate solutions of Example 6.1.

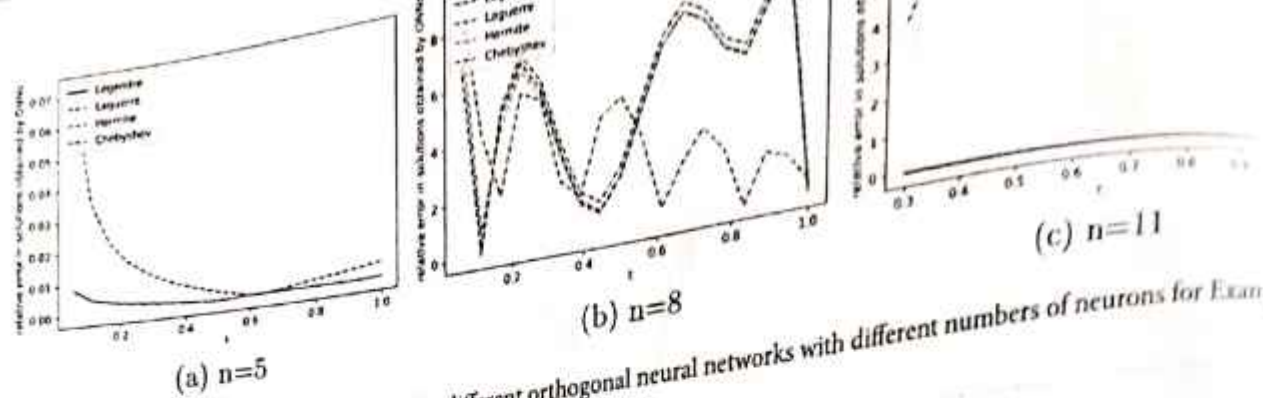


Figure 3. Error graph for different orthogonal neural networks with different numbers of neurons for Example 6.1.

n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
5	0.001	0.0049	0.009	0.0049	0.002	0.0049	0.008	0.0049
8	0.012	<b>7.07e-08</b>	0.011	<b>6.97e-08</b>	0.003	<b>7.07e-08</b>	0.043	<b>7.07e-08</b>
11	0.019	<b>1.82e-11</b>	0.011	<b>7.74e-06</b>	0.003	<b>6.81e-10</b>	0.043	<b>3.93e-12</b>

Table 2. Comparison of maximum relative error for Example 6.1 with different numbers of neurons. Significant values are in bold.

**Example 6.2** <sup>2</sup> Consider the second-order neutral delay differential equation with multiple delays

$$z''(t) = \frac{3}{4}z(t) + z\left(\frac{t}{2}\right) + z'\left(\frac{t}{2}\right) + 0.5z''\left(\frac{t}{2}\right) + f(t),$$

$$z(0) = 0, z'(0) = 0,$$

where  $f(t) = -t^2 - t + 1, t \in (0, 1)$ .

The exact solution of the given equation is  $z(t) = t^2$ .

This equation is solved using four ONNs architecture with ten uniformly distributed training points and with 6, 8, and 9 neurons in the hidden layer. Relative errors for the different ONNs with 6, 8, and 9 neurons as activation functions are reported in Table 3. Figure 4 shows an error graph of different orthogonal neural networks, and a comparison of approximate solutions with the exact solution is shown in Fig. 5.

From Table 4 and Fig. 4 we conclude that for the given second-order neutral delay differential equation, Chebyshev polynomial-based ONN performs best with the maximum relative error  $7.19 \times 10^{-14}$ . Additionally, Table 3 shows that all four neural networks satisfy Theorem 4.

**Example 6.3** <sup>3</sup> Consider the second-order neutral delay differential equation with variable coefficients

$$z''(t) = z'\left(\frac{t}{2}\right) - \frac{t}{2}z''\left(\frac{t}{2}\right) + 2, t \in (0, 1)$$

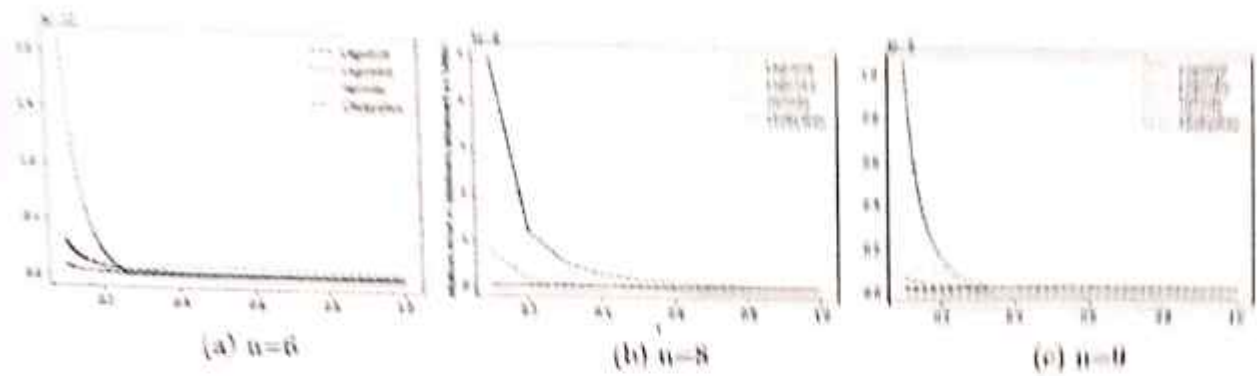
$$z(0) = 1, z'(0) = 0.$$

The exact solution of the given equation is  $z(t) = t^2 + 1$ .

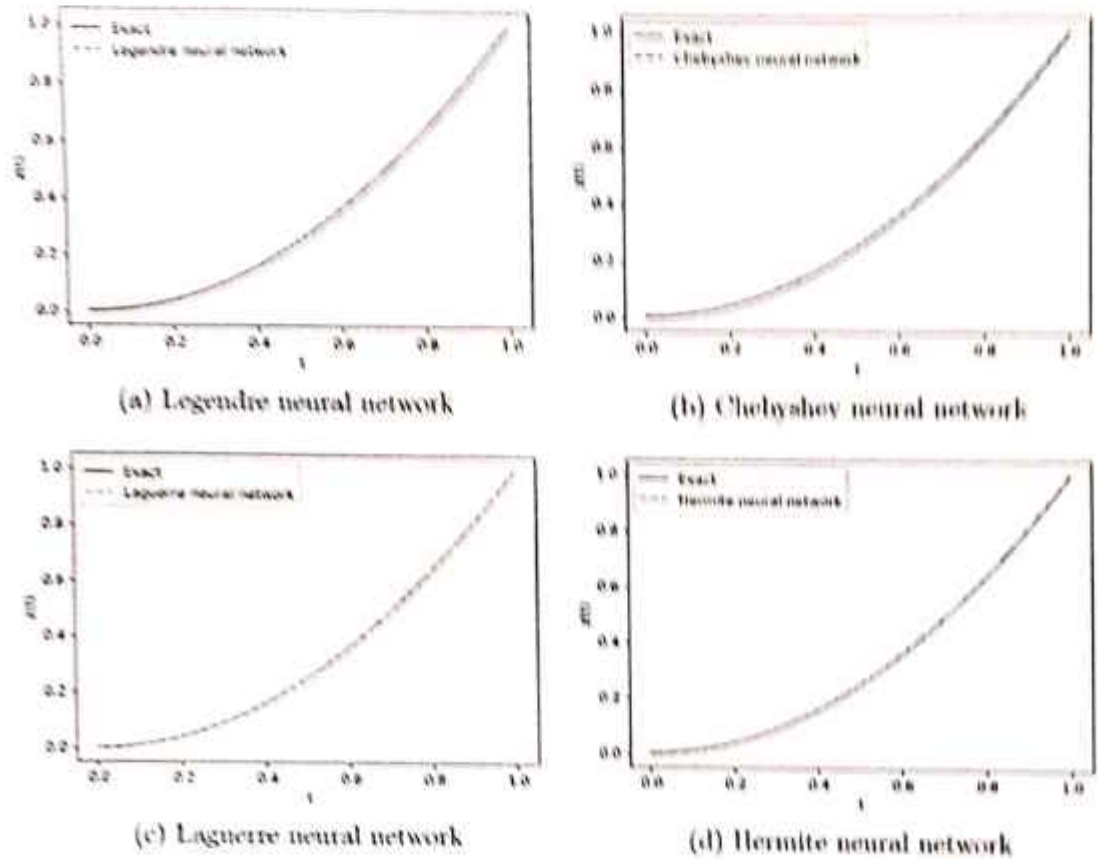
n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
6	0.007	<b>8.25e-14</b>	0.004	<b>2.06e-12</b>	0.002	<b>2.87e-13</b>	0.001	<b>3.17e-14</b>
8	0.009	<b>4.94e-08</b>	0.004	<b>7.29e-09</b>	0.004	<b>7.73e-13</b>	0.002	<b>7.19e-14</b>
9	0.019	<b>2.22e-14</b>	0.009	<b>1.57e-08</b>	0.017	<b>6.13e-13</b>	0.022	<b>6.77e-15</b>

Table 3. Comparison of maximum relative error for Example 6.2 with different numbers of neurons. Significant values are in bold.





**Figure 4.** Error graph for different orthogonal neural networks with different numbers of neurons for Example 6.2.



**Figure 5.** Comparison of the exact solution with the obtained approximate solutions of Example 6.2.

t	Legendre neural network	Hermite neural network	Laguerre neural network	Chebyshev neural network
0.1	4.94e-08	7.75e-15	7.79e-09	7.19e-14
0.2	1.19e-06	1.90e-15	1.32e-09	1.65e-14
0.3	5.05e-09	7.81e-14	3.83e-10	9.25e-15
0.5	1.49e-09	1.28e-14	8.31e-12	5.55e-15
0.6	8.88e-10	5.08e-15	5.15e-11	6.32e-15
0.7	5.20e-10	7.84e-15	5.12e-11	7.15e-15
0.8	2.81e-10	1.57e-14	5.79e-11	8.15e-15
0.9	1.17e-10	1.90e-14	5.85e-11	9.45e-15
1	2.80e-15	2.17e-14	5.98e-11	1.11e-14

**Table 4.** The relative error for Example 6.2 with different orthogonal neural networks.

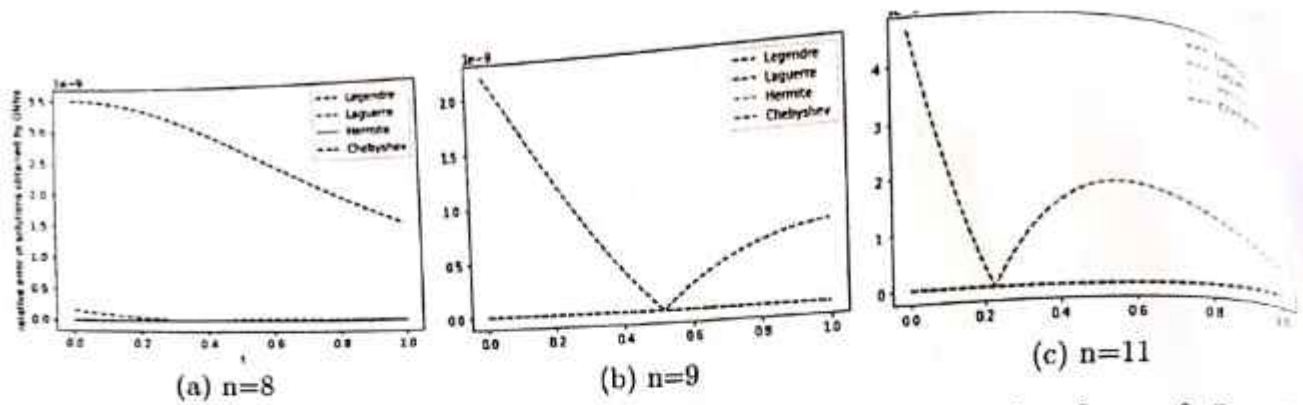


Figure 6. Error graph for different orthogonal neural networks with different numbers of neurons for Example 6.3.

To obtain the approximate solution of the given equation, we use four ONNs with ten uniformly distributed training points in  $[0,1]$  and with 8,9, and 11 neurons as activation functions in the hidden layer. Relative errors for the different ONNs and with different numbers of neurons are reported in Table 6. The exact and approximate solutions are compared in Fig. 7. Figures 6, 7 shows the absolute relative error of four special ONNs.

From Table 5 and Fig. 6, we conclude that for the given second-order neutral delay differential equation, Chebyshev polynomial-based ONN provides the best accurate solution with the maximum relative error  $2.29 \times 10^{-15}$ . Additionally, Table 6 shows that all four neural networks satisfy Theorem 4.

**Example 6.4** <sup>31</sup> Consider the third-order pantograph equation

$$z'''(t) = tz''(2t) - z'(t) - z\left(\frac{t}{2}\right) + t\cos(2t) + \cos\left(\frac{t}{2}\right), t \in (0,1)$$

$$z(0) = 1, z'(0) = 0, z''(0) = -1.$$

The exact solution of the given equation is  $z(t) = \cos(t)$ .

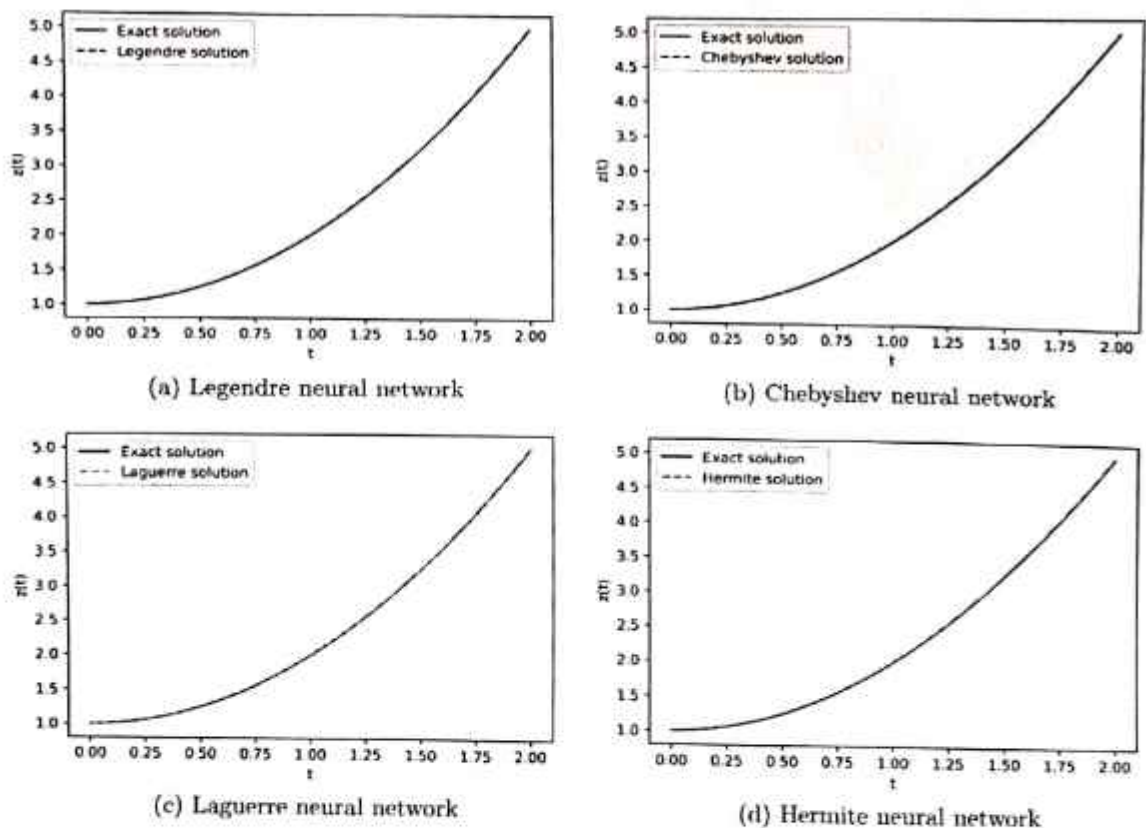


Figure 7. Comparison of the exact solution with the obtained approximate solutions of Example 6.3.

t	Legendre neural network	Hermite neural network	Laguerre neural network	Chebyshev neural network
0.0	3.50e-09	1.17e-13	1.52e-10	7.77e-16
0.1	3.46e-09	1.14e-13	8.90e-11	0.0
0.2	3.34e-09	1.01e-13	4.18e-11	1.06e-15
0.3	3.16e-09	8.06e-14	1.01e-11	1.83e-15
0.4	2.94e-09	5.28e-14	8.50e-12	2.29e-15
0.5	2.70e-09	1.98e-14	1.72e-11	2.13e-15
0.6	2.44e-09	1.58e-14	1.90e-11	1.63e-15
0.7	2.18e-09	5.27e-14	1.66e-11	1.04e-15
0.8	1.93e-09	8.93e-14	1.21e-11	2.70e-16
0.9	1.70e-09	1.24e-13	7.09e-12	4.90e-16
1.0	1.50e-09	1.52e-13	2.35e-12	8.88e-16

**Table 5.** The relative error for Example-6.3 with different orthogonal neural networks.

n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
8	0.007	3.50e-09	0.004	1.52e-10	0.002	1.52e-13	0.001	<b>2.29e-15</b>
9	0.009	3.07e-13	0.004	8.80e-11	0.004	1.69e-05	0.002	<b>2.29e-15</b>
11	0.019	3.07e-13	0.069	3.16e-11	0.017	1.80e-05	0.022	<b>1.32e-15</b>

**Table 6.** Comparison of maximum relative error for Example 6.3 with different numbers of neurons. Significant values are in bold.

To obtain the approximate solution of the given equation, we use four ONNs with ten uniformly distributed training points in  $[0,1]$  and with 8,11,13 neurons as activation functions in the hidden layer. Relative errors for the different ONNs with different numbers of neurons as activation functions are reported in Table 7. The exact and approximate solutions are compared in Fig. 8. Figure 9 shows the maximum relative error of four special ONNs with different numbers of neurons.

From Table 8 and Fig. 9, we conclude that for the given third-order neutral delay differential equation, Chebyshev polynomial-based ONN provides the best accurate solution with the maximum relative error  $3.77 \times 10^{-10}$ . Additionally, Table 7 shows that all four orthogonal neural networks satisfy Theorem 4.

### Comparative analysis

This section describes a comparative study of the proposed approach to the 1st-order pantograph equation and system of pantograph equations with other neural network approaches.

**Example 7.1** <sup>25</sup> Consider the pantograph equation with variable coefficients and multiple delays

$$z'(t) = 0.5z(t) + 0.5e^{0.5t}z\left(\frac{t}{2}\right) + \frac{3}{8}tz\left(\frac{t}{3}\right) + g(t),$$

$$z(0) = 0,$$

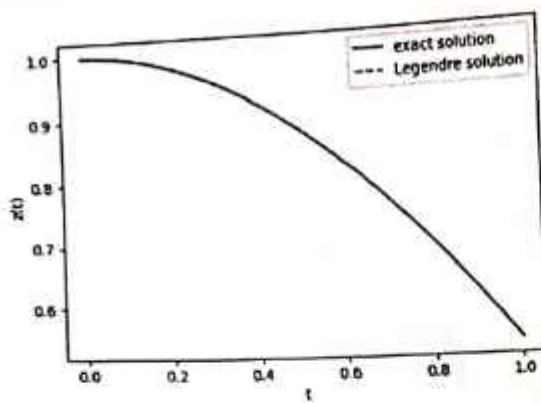
where,  $g(t) = \frac{1}{8}e^{-t}(12\sin(t) + 4e^t\sin(\frac{t}{2}) - 8\cos(t) + 3te^{\frac{2t}{3}}\sin(\frac{t}{3}))$ .

The exact solution of the given equation is  $z(t) = \sin(t)e^{-t}$ .

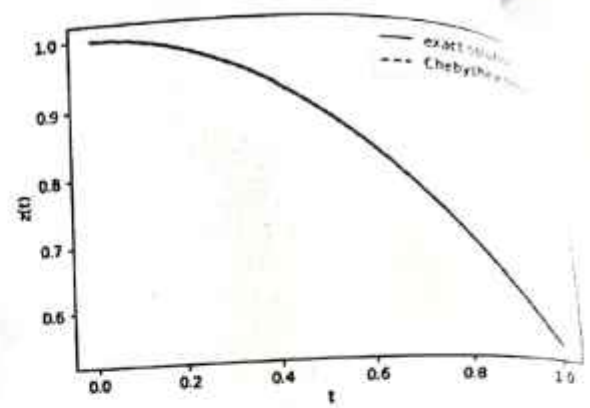
n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
8	0.007	1.11e-05	0.004	1.11e-05	0.002	1.11e-05	0.001	1.11e-05
11	0.019	<b>3.86e-08</b>	0.004	6.56e-08	0.004	6.06e-08	0.004	<b>3.86e-08</b>
13	0.019	1.12e-09	0.069	3.11e-09	0.017	1.20e-06	0.022	3.77e-10

**Table 7.** Comparison of maximum relative error for Example 6.4 with different numbers of neurons. Significant values are in bold.

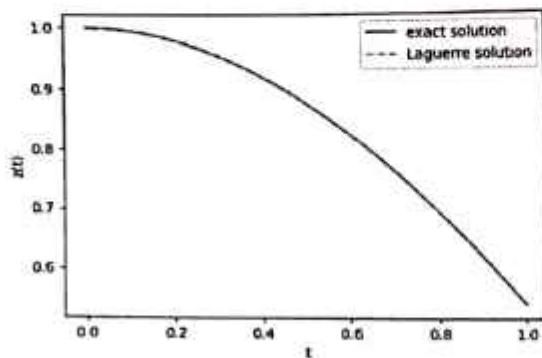




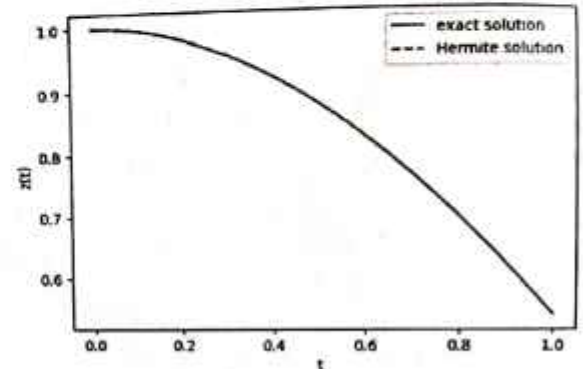
(a) Legendre neural network



(b) Chebyshev neural network



(c) Laguerre neural network



(d) Hermite neural network

Figure 8. Comparison of the exact solution with the obtained approximate solutions of Example 6.4.

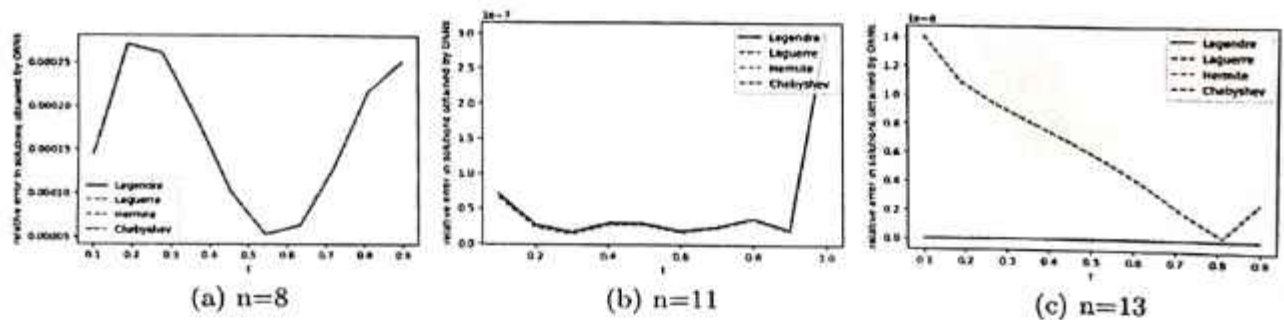


Figure 9. Error graph for different orthogonal neural networks with different numbers of neurons for Example 6.4.

We employ four ONNs to obtain the approximate solution of a given pantograph equation with multiple delays. We choose eight uniformly distributed points in  $[0, 1]$  with 5, 8 and 11 neurons in the hidden layer. The relative errors with all four ONNs with different numbers of neurons are shown in Fig. 11. Obtained relative errors for the different orthogonal neural networks are reported in Table 9, and we compare the approximate solutions with the exact solution in Fig. 10.

Table 9 and Fig. 11 clearly show that the Chebyshev polynomial-based ONN performs best with the maximum relative error  $3.40 \times 10^{-11}$ .

The maximum relative error of a simple feed-forward neural network (FNN) method in<sup>25</sup> is  $4.05 \times 10^{-10}$  and the maximum relative error of the proposed FLNN-based ONN method is  $3.40 \times 10^{-11}$ . This comparison shows that the ONN method can obtain a better accuracy solution than simple FNN. Additionally, Table 9 shows that all four orthogonal neural networks satisfy Theorem 4.

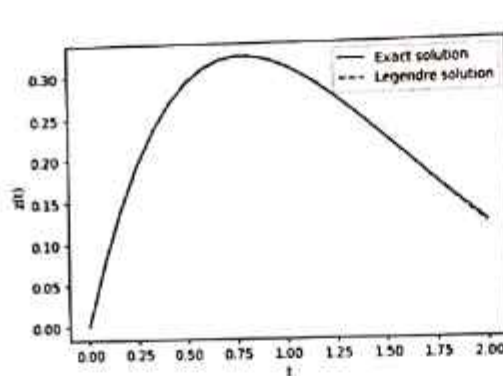
**Example 7.2** <sup>25</sup> Consider the system of pantograph equation

$t$	Legendre neural network	Hermite neural network	Laguerre neural network	Chebyshev neural network
0	3.79e-10	7.39e-07	5.91e-10	1.00e-11
0.1	3.82e-10	5.75e-07	6.05e-10	6.11e-12
0.2	3.89e-10	3.94e-07	6.33e-10	5.81e-12
0.3	4.02e-10	2.05e-07	6.75e-10	2.65e-11
0.4	4.21e-10	1.32e-08	7.33e-10	5.73e-11
0.5	4.49e-10	1.79e-07	8.11e-10	9.98e-11
0.6	4.88e-10	3.71e-07	9.22e-10	1.54e-10
0.7	5.49e-10	5.65e-07	1.10e-09	2.20e-10
0.8	6.19e-10	7.65e-07	1.41e-09	2.92e-10
0.9	8.21e-10	9.72e-07	2.00e-09	3.54e-10
1	1.12e-09	1.20e-06	3.11e-09	3.77e-10

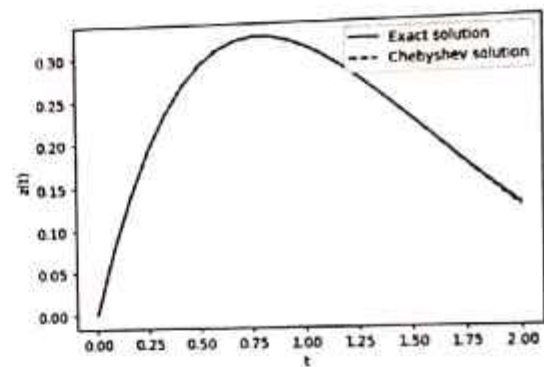
Table 8. The relative error for Example 6.4 with different orthogonal neural networks.

$n$	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
5	0.007	0.0014	0.004	0.0014	0.002	0.0014	0.001	0.0014
8	0.019	7.02e-08	0.004	6.56e-08	0.004	7.02e-08	0.004	7.02e-08
11	0.019	4.75e-11	0.069	1.06e-06	0.017	2.63e-09	0.022	3.46e-11

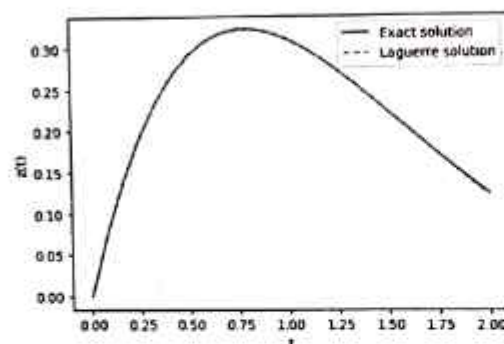
Table 9. Comparison of maximum relative error for Example 7.1 with different numbers of neurons. Significant values are in bold.



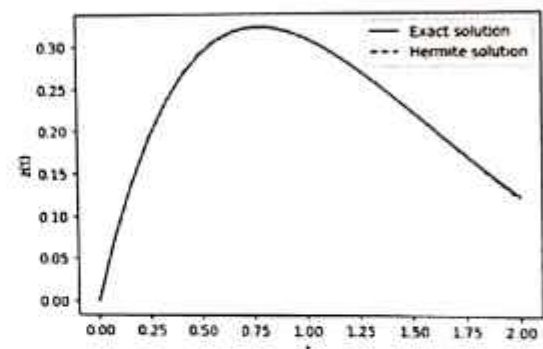
(a) Legendre neural network



(b) Chebyshev neural network



(c) Laguerre neural network



(d) Hermite neural network

Figure 10. Comparison of the exact solution with the obtained approximate solutions of Example 7.1.

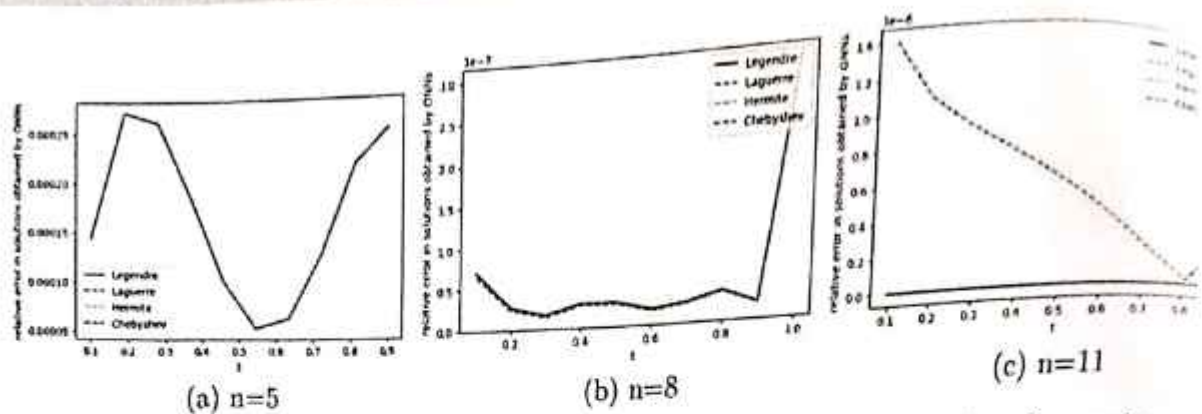


Figure 11. Error graph for different orthogonal neural networks with different numbers of neurons for Example 7.1.

$$\begin{aligned} z_1'(t) &= z_1(t) - z_2(t) + z_1\left(\frac{t}{2}\right) - e^{0.5t} + e^t, \\ z_2'(t) &= -z_1(t) - z_2(t) - z_2\left(\frac{t}{2}\right) + e^{-0.5t} + e^t, \\ z_1(0) &= 1, z_2(0) = 1. \end{aligned}$$

The exact solutions of the given system of pantograph equation is  $z_1(t) = e^t$  and  $z_2(t) = e^{-t}$ .

To obtain the approximate solutions of the given system of DDEs, we use four ONNs with twelve uniformly distributed training points in  $[0, 1]$  and with 5, 7, and 10 neurons in an orthogonal functional expansion block as activation functions. Relative errors for the different ONNs with 5, 7, and 10 neurons as activation functions are reported in Tables 10 and 11. Comparison between the exact solution and approximate solutions are presented in Figs. 14 and 15. Figures 12, 13, 14 and 15 show the absolute relative error between four special ONNs and exact solutions.

From Tables 10 and 11, we conclude that for the given system of delay differential equation, Chebyshev polynomial-based ONN provides the best accurate solution for  $z_1(t)$  and  $z_2(t)$  with the maximum relative errors  $1.60 \times 10^{-9}$  and  $5.11 \times 10^{-10}$ , respectively.

The maximum relative error of a simple feed-forward neural network (FNN) method in<sup>25</sup> for  $z_1(t)$  and  $z_2(t)$  with twelve training points are  $1.93 \times 10^{-9}$  and  $2.42 \times 10^{-9}$  respectively and the maximum relative error of the proposed FLNN-based ONN method for  $z_1(t)$  and  $z_2(t)$  with twelve training points are  $1.60 \times 10^{-9}$  and  $5.11 \times 10^{-10}$  respectively. This comparison shows that the ONN method can obtain a better accuracy solution than simple FNN. Additionally, Tables 10 and 11 show that all four orthogonal neural networks satisfy Theorem 4.

**Example 7.3** <sup>23</sup> Consider the system of pantograph equation

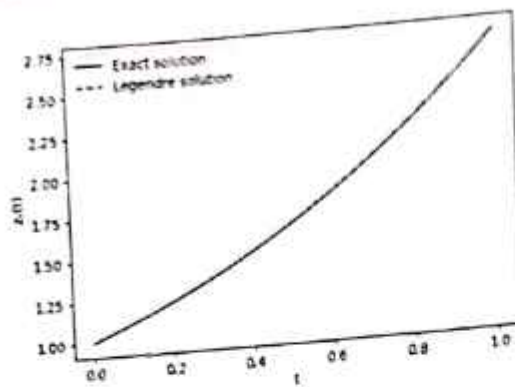
n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
5	0.005	0.06	0.004	0.0004	0.002	0.0004	0.001	0.0004
7	0.019	0.067	0.004	1.72e-06	0.004	1.93e-07	0.004	1.93e-07
10	0.019	3.25e-10	0.069	1.71e-06	0.017	1.81e-09	0.022	1.60e-10

Table 10. Comparison of maximum relative error of  $z_1(t)$  for Example 7.2 with different numbers of neurons. Significant values are in bold.

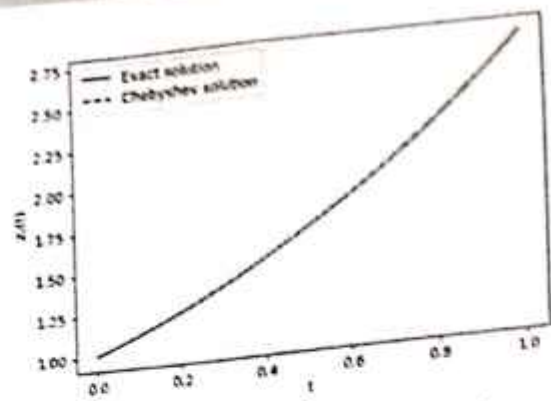
n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
5	0.005	0.312	0.004	0.0006	0.002	0.0006	0.001	0.0006
7	0.019	0.02	0.004	1.94e-06	0.004	2.00e-07	0.004	6.47e-09
10	0.019	1.42e-08	0.069	2.20e-08	0.017	5.91e-06	0.022	5.11e-10

Table 11. Comparison of maximum relative error of  $z_2(t)$  for Example 7.2 with different numbers of neurons. Significant values are in bold.

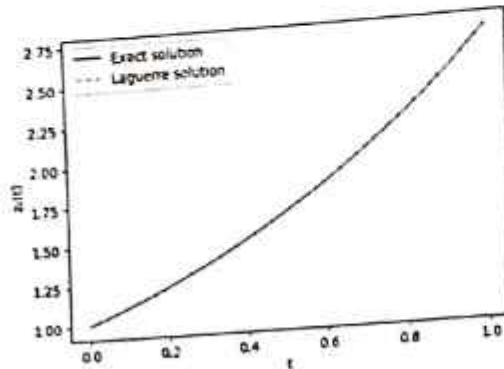




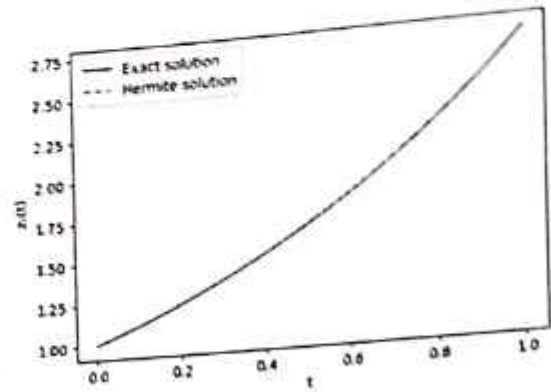
(a) Legendre neural network



(b) Chebyshev neural network



(c) Laguerre neural network



(d) Hermite neural network

Figure 12. Comparison of the exact solution  $z_1(t)$  with the obtained approximate solutions of Example 7.2.

$$\begin{aligned} z_1'(t) + z_2'(t) - 2z_3'(t) &= z_1(0.2t) + z_2(t) - z_2(0.3t) \\ &\quad - 2z_3(t) - z_3(0.3t) + z_3(0.5t) + f_1(t), \\ z_1'(t) - z_2'(t) &= z_1(t) - z_3(t) + 3z_1(0.5t) \\ &\quad - z_2(0.5t) + z_2(0.3t) + z_3(0.7t) + f_2(t), \\ z_2'(t) - 2z_3'(t) &= z_1(t) - z_3(0.8t) + 3z_2(t) \\ &\quad - z_1(0.2t) + z_3(0.8t) + f_3(t), \\ z_1(0) &= 0, z_2(0) = 1, z_3(0) = 1, \end{aligned}$$

$$\text{where, } f_1(t) = \cos(0.3t) - \sin(0.2t) - \sin(t) + e^{0.3t} - e^{0.5t},$$

$$f_2(t) = -\cos(0.3t) + \cos(0.5t) - 3\sin(0.5t) + \cos(t) - e^{0.7t} + e^t,$$

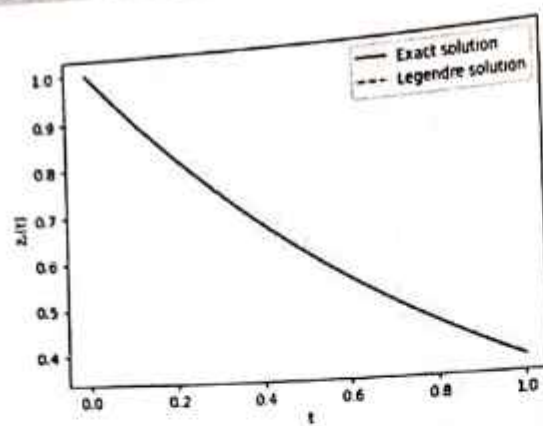
$$f_3(t) = -\cos(0.8t) + \sin(0.2t) - 3\cos(t) - 2\sin(t) + e^{0.8t} - 2e^t.$$

The exact solutions of the given system of pantograph equation are  $z_1(t) = \sin(t)$ ,  $z_2(t) = \cos(t)$ , and  $z_3(t) = e^t$ .

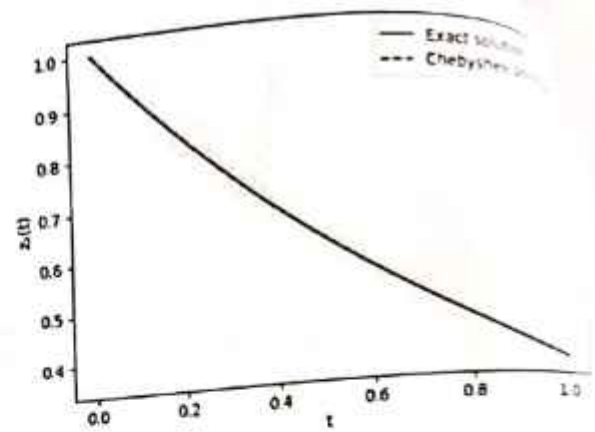
To obtain the approximate solution of the given system of DDEs, we use four ONNs with ten uniformly distributed training points in  $[0,1]$  and with 7, 10, and 13 neurons in an orthogonal functional expansion block as activation functions. Relative errors for the different ONNs with 7, 10, and 13 neurons as activation functions are reported in Tables 12, 13, and 14. Comparison between the exact solution and approximate solutions are presented in Figs. 16, 17, 18, and 19. Figures 16, 20, and 21 show the absolute relative error between four special ONNs and exact solutions.

From Tables 12, 13 and 14, we conclude that for the given system of delay differential equation, Chebyshev polynomial-based ONN provides the best accurate solutions of  $z_1(t)$ ,  $z_2(t)$  and  $z_3(t)$  with the maximum relative errors  $1.98 \times 10^{-10}$ ,  $3.11 \times 10^{-10}$  and  $5.74 \times 10^{-9}$  respectively.

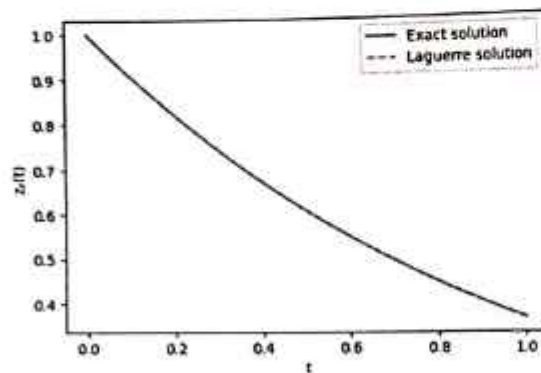
The maximum relative error of a simple feed-forward neural network (FNN) method in<sup>25</sup> for  $z_1(t)$ ,  $z_2(t)$  and  $z_3(t)$  with ten training points are  $8.78 \times 10^{-8}$ ,  $1.42 \times 10^{-8}$  and  $1.93 \times 10^{-7}$  respectively and the maximum relative error of the proposed FLNN-based ONN method for  $z_1(t)$ ,  $z_2(t)$  and  $z_3(t)$  with ten training points are



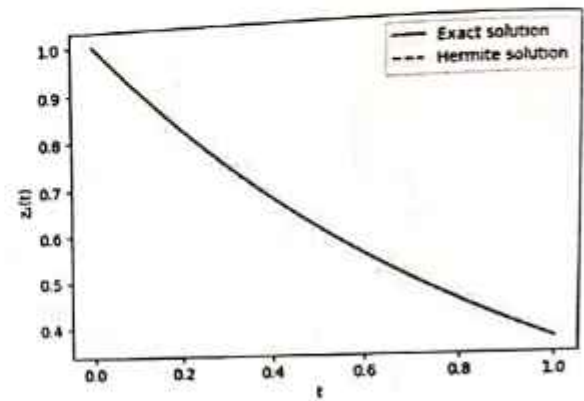
(a) Legendre neural network



(b) Chebyshev neural network



(c) Laguerre neural network



(d) Hermite neural network

Figure 13. Comparison of the exact solution  $z_1(t)$  with the obtained approximate solutions of Example 7.2.

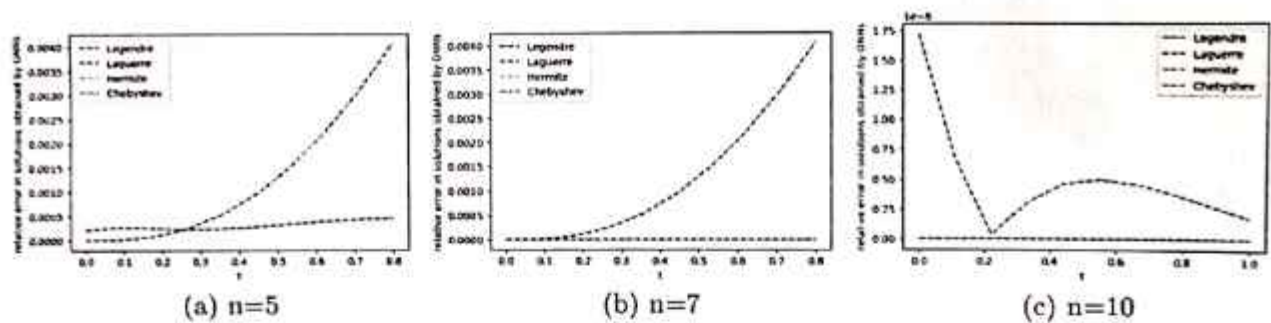


Figure 14. Error graph of  $z_1(t)$  for different orthogonal neural networks with different numbers of neurons for Example 7.2.

$1.98 \times 10^{-10}$ ,  $3.11 \times 10^{-10}$  and  $5.74 \times 10^{-9}$  respectively. This comparison shows that the ONN method can obtain a better accuracy solution than simple FNN. Additionally, Tables 12, 13 and 14 show that all four orthogonal neural networks satisfy Theorem 4.

## Conclusion

In this paper, we obtained approximate solutions of higher order NDDEs, as well as a system of DDEs with multiple delays and variable coefficients, using four single-layer orthogonal polynomial-based neural networks: (i) Legendre neural network, (ii) Chebyshev neural network, (iii) Hermite neural network, and (iv) Laguerre neural network. For training the network weights, the ELM algorithm is used. It is proved that the relative error between the exact solution and approximate solutions obtained by ONNs is of order  $2^{-n}$ , where  $n$  is the number of neurons. Further, it is shown that each orthogonal polynomial-based neural networks provide an approximate solution, that are in good agreement with the exact solution. However, it is observed that, among these four ONNs, the Chebyshev neural network provides the most accurate result.

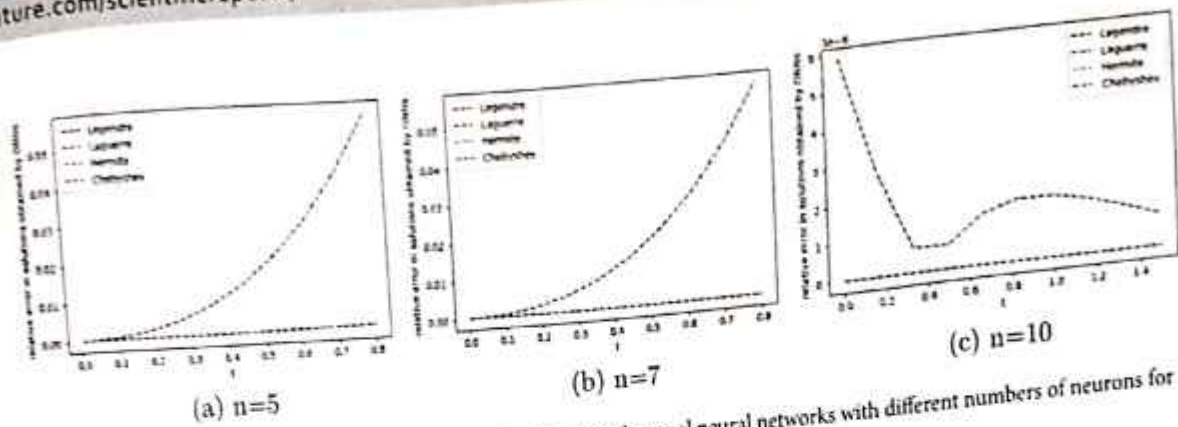


Figure 15. Error graph of  $z_1(t)$  for different orthogonal neural networks with different numbers of neurons for Example 7.2.

n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
7	0.007	<b>5.41e-07</b>	0.004	<b>6.20e-07</b>	0.002	<b>6.17e-07</b>	0.001	<b>6.16e-07</b>
10	0.019	<b>9.87e-08</b>	0.004	<b>6.97e-07</b>	0.004	<b>1.45e-09</b>	0.004	<b>1.53e-10</b>
13	0.019	<b>9.11e-11</b>	0.069	<b>5.79e-07</b>	0.017	<b>1.23e-09</b>	0.022	<b>1.98e-11</b>

Table 12. Comparison of maximum relative error of  $z_1(t)$  for Example 7.3 with different numbers of neurons. Significant values are in bold.

n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
7	0.007	<b>1.60e-07</b>	0.004	<b>3.18e-07</b>	0.002	<b>3.17e-05</b>	0.001	<b>2.93e-07</b>
10	0.019	<b>1.05e-09</b>	0.004	<b>5.71e-07</b>	0.004	<b>5.03e-10</b>	0.004	<b>3.70e-10</b>
13	0.019	<b>1.05e-09</b>	0.069	<b>5.24e-07</b>	0.017	<b>8.04e-09</b>	0.022	<b>3.11e-10</b>

Table 13. Comparison of maximum relative error of  $z_2(t)$  for Example 7.3 with different numbers of neurons. Significant values are in bold.

n	Legendre		Laguerre		Hermite		Chebyshev	
	Time(s)	Error	Time(s)	Error	Time(s)	Error	Time(s)	Error
7	0.007	<b>1.31e-07</b>	0.004	<b>1.76e-07</b>	0.002	<b>1.78e-07</b>	0.001	<b>1.74e-07</b>
10	0.019	<b>2.82e-08</b>	0.004	<b>4.05e-07</b>	0.004	<b>1.27e-08</b>	0.004	<b>3.91e-09</b>
13	0.019	<b>5.18e-08</b>	0.069	<b>5.65e-07</b>	0.017	<b>4.23e-08</b>	0.022	<b>5.74e-09</b>

Table 14. Comparison of maximum relative error of  $z_3(t)$  for Example 7.3 with different numbers of neurons. Significant values are in bold.

The results in the section (6), (7) demonstrate that the proposed method is simple to implement and a powerful mathematical technique for obtaining the approximate solution of the higher order NDDEs as well as the system of DDEs.



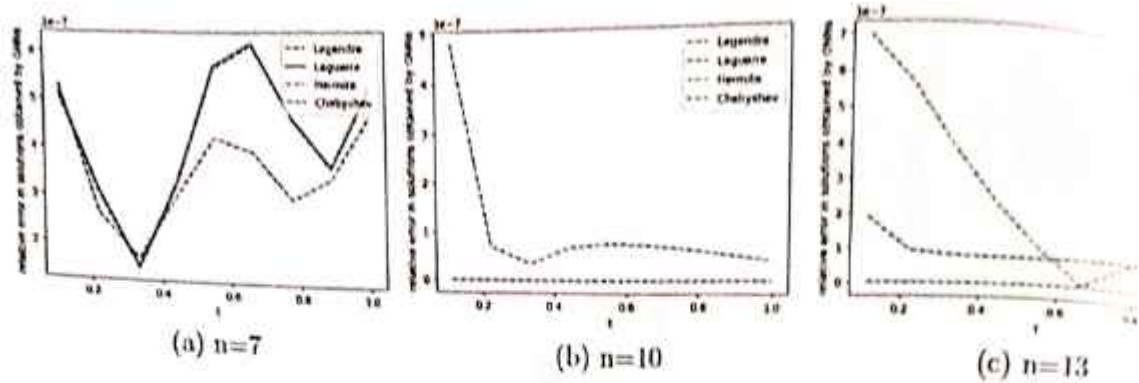


Figure 16. Error graph of  $z_1(t)$  for different orthogonal neural networks with different numbers of neurons, for Example 7.3.

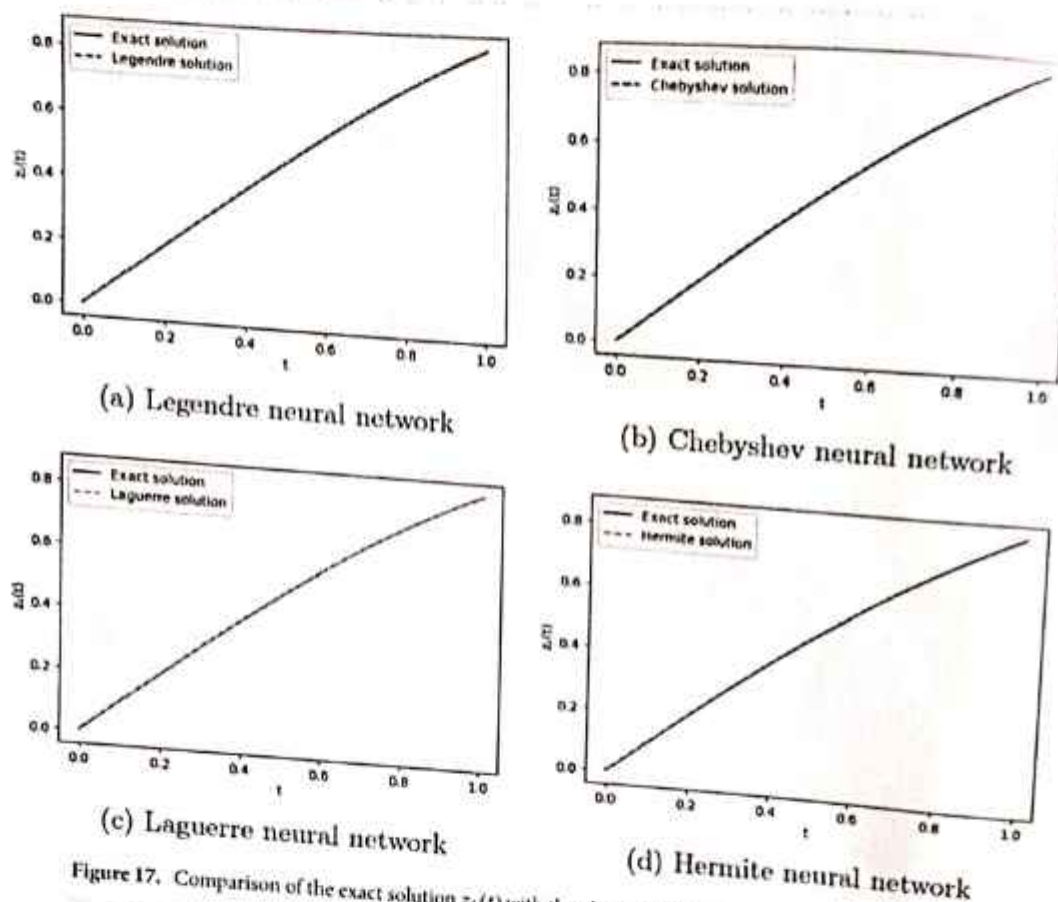
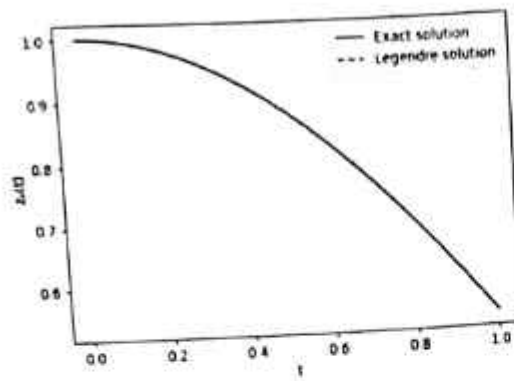
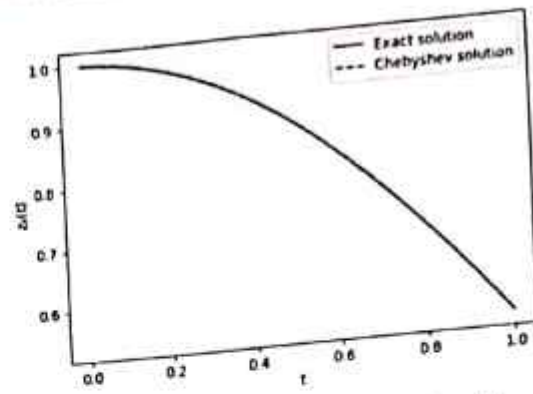


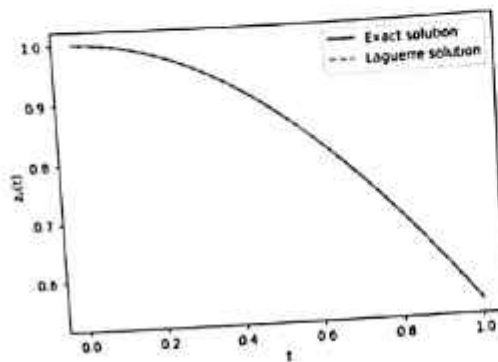
Figure 17. Comparison of the exact solution  $z_1(t)$  with the obtained approximate solutions of Example 7.3.



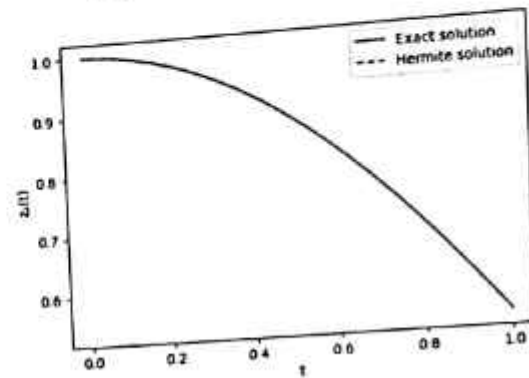
(a) Legendre neural network



(b) Chebyshev neural network

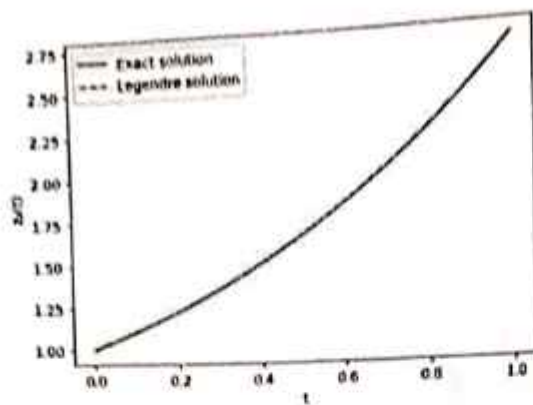


(c) Laguerre neural network

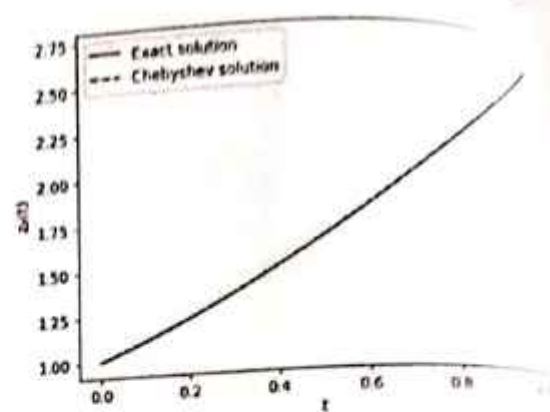


(d) Hermite neural network

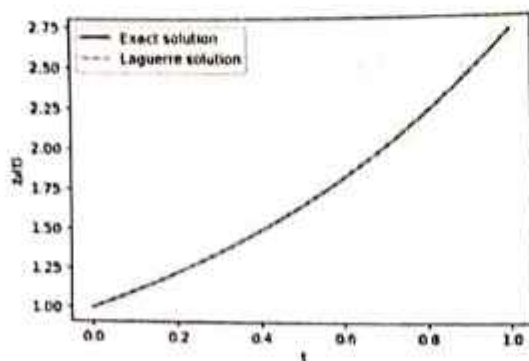
Figure 18. Comparison of the exact solution  $z_2(t)$  with the obtained approximate solutions of Example 7.3.



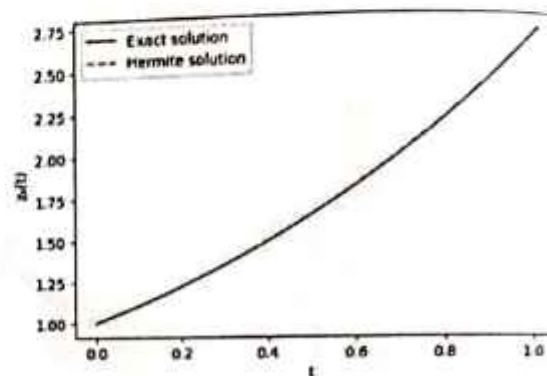
(a) Legendre neural network



(b) Chebyshev neural network

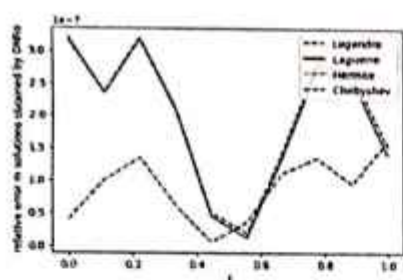


(c) Laguerre neural network

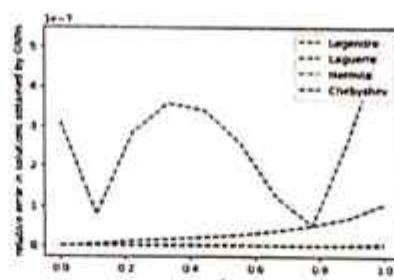


(d) Hermite neural network

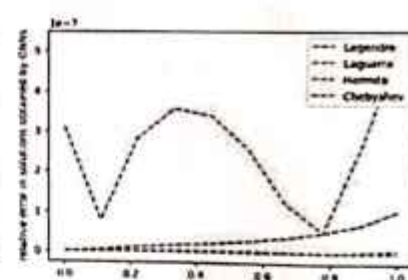
Figure 19. Comparison of the exact solution  $z_3(t)$  with the obtained approximate solutions of Example 7.3.



(a)  $n=7$



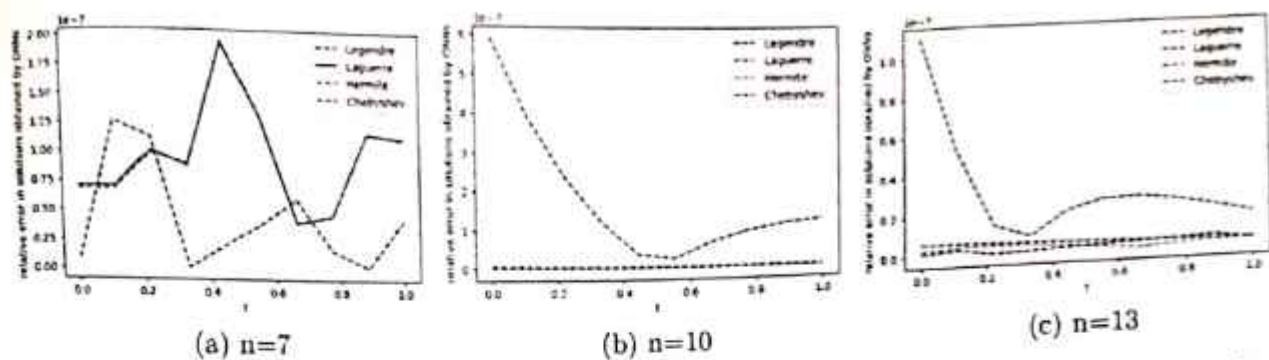
(b)  $n=10$



(c)  $n=13$

Figure 20. Error graph of  $z_2(t)$  for different orthogonal neural networks with different numbers of neurons for Example 7.3.





**Figure 21.** Error graph of  $z_3(t)$  for different orthogonal neural networks with different numbers of neurons for Example 7.3.

### Data availability

The data that support the findings of this investigation are accessible from the authors upon reasonable request. If necessary, you can contact by email [sdubey@iitm.ac.in](mailto:sdubey@iitm.ac.in).

Received: 19 October 2022; Accepted: 16 February 2023

Published online: 23 February 2023

### References

- Ockendon, J. R. & Tayler, A. B. The dynamics of a current collection system for an electronic locomotive. *Numer. Math.* **72**(2), 447–468 (1971).
- Biazar, J. & Ghanbari, B. The homotopy perturbation method for solving neutral functional-differential equations with proportional delays. *J. King Saud Univ. Sci.* **24**(1), 33–37 (2012).
- Bahji, M. M. & Çevik, M. Numerical solution of pantograph-type delay differential equations using perturbation-iteration algorithms. *J. Appl. Math.* **2015** (2015).
- Bahuguna, D. & Agarwal, S. Approximations of solutions to neutral functional differential equations with nonlocal history conditions. *J. Math. Anal. Appl.* **317**(2), 583–602 (2006).
- Dubey, S. A. The method of lines applied to nonlinear nonlocal functional differential equations. *J. Math. Anal. Appl.* **376**(1), 275–281 (2011).
- Ahina, M., Thakur, S. & Moyo, S. Exact solutions of nonlinear delay reaction-diffusion equations with variable coefficients. *Partial Differ. Equ. Appl. Math.* **4**, 100170 (2021).
- Mahata, A., Paul, S., Mukherjee, S. & Roy, B. Stability analysis and Hopf bifurcation in fractional order SEIRV epidemic model with a time delay in infected individuals. *Partial Differ. Equ. Appl. Math.* **5**, 100282 (2022).
- Cakmak, M. & Alkan, S. A numerical method for solving a class of systems of nonlinear pantograph differential equations. *Alex. Eng. J.* **61**(4), 2651–2661 (2022).
- Muslim, M. Approximation of solutions to history-valued neutral functional differential equations. *Comput. Math. Appl.* **51**(3–4), 537–550 (2006).
- 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).
- Aarts, L. P. & Van Der Veer, P. Neural network method for solving partial differential equations. *Neural Process. Lett.* **14**(3), 261–271 (2001).
- Mall, S. & Chakraverty, S. Application of Legendre neural network for solving ordinary differential equations. *Appl. Soft Comput.* **43**, 347–356 (2016).
- Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
- Panghal, S. & Kumar, M. Multilayer perceptron and Chebyshev polynomials based neural network for solving Emden–Fowler type initial value problems. *Int. J. Appl. Comput. Math.* **6**(6), 1–12 (2020).
- Ezadi S. & Parand N. An application of neural networks to solve ordinary differential equations (2013).
- Liu, Z., Yang, Y. & Cai, Q. Neural network as a function approximator and its application in solving differential equations. *Appl. Math. Mech.* **40**(2), 237–248 (2019).
- Pakdaman, M., Ahmadian, A., Effati, S., Salahshour, S. & Baleanu, D. Solving differential equations of fractional order using an optimization technique based on training artificial neural network. *Appl. Math. Comput.* **293**, 81–95 (2017).
- Nguyen, L., Raissi, M. & Seshaiyer, P. *Efficient Physics Informed Neural Networks Coupled with Domain Decomposition Methods for Solving Coupled Multi-physics Problems* 41–53 (Springer, 2022).
- Mall, S. & Chakraverty, S. Numerical solution of nonlinear singular initial value problems of Emden–Fowler type using Chebyshev neural network method. *Neurocomputing* **149**, 975–982 (2015).
- Dufera, T. T. Deep neural network for system of ordinary differential equations: Vectorized algorithm and simulation. *Mach. Learn. Appl.* **5**, 100058 (2021).
- Fang, J., Liu, C., Simos, T. & Fanelis, I. T. Neural network solution of single-delay differential equations. *Mediterr. J. Math.* **17**(1), 1–15 (2020).
- Hou, C.-C., Simos, T. E. & Fanelis, I. T. Neural network solution of pantograph type differential equations. *Math. Methods Appl. Sci.* **43**(6), 3369–3374 (2020).
- Panghal, S. & Kumar, M. Optimization free neural network approach for solving ordinary and partial differential equations. *Eng. Comput.* **37**(4), 2989–3002 (2021).
- Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **70**(1–3), 489–501 (2006).
- Panghal, S. & Kumar, M. Neural network method: delay and system of delay differential equations. *Eng. Comput.* **1**–10 (2021).
- Liu, H., Song, J., Liu, H., Xu, J. & Li, L. Legendre neural network for solving linear variable coefficients delay differential-algebraic equations with weak discontinuities. *Adv. Appl. Math. Mech.* **13**(1), 101–118 (2021).

27. Mall, S. & Chakraverty, S. *Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations*, 1st ed. 168 (2017).
28. Verma, A. & Kumar, M. Numerical solution of third-order Emden-Fowler type equations using artificial neural network technique. *Eur. Phys. J. Plus* **135**(9), 1–14 (2020).
29. Verma, A. & Kumar, M. Numerical solution of Bagley-Torvik equations using Legendre artificial neural network method. *Evol. Intell.* **14**(4), 2027–2037 (2021).
30. Serre, D. *Matrices: Theory and Applications* (Springer Inc, 2002).
31. Sezer, M. & Akyüz-Daşcıoğlu, A. A Taylor method for numerical solution of generalized pantograph equations with linear functional argument. *J. Comput. Appl. Math.* **200**(1), 217–225 (2007).

## Acknowledgements

Chavda Divyesh Vinodbhai acknowledges the financial support provided by the MoE (Ministry of Education), Government of India, to carry out the work. The second author is thankful for the financial support received from the Indian Institute of Technology Madras.

## Author contributions

The contributions of each authors are equal.

## Competing interests


The authors declare no competing interests.

## Additional information

**Correspondence** and requests for materials should be addressed to S.D.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2023