Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Application of Legendre Neural Network for solving ordinary differential equations

Q1 Susmita Mall, S. Chakraverty*

Department of Mathematics, National Institute of Technology, Rourkela 769008, Odisha, India

A R T I C L E I N F O

Article history:
Received 11 July 2014
Received in revised form 20 July 2015
Accepted 10 October 2015
Available online xxx

Keywords:
Differential equations
Feed forward neural network model
Back propagation algorithm
Functional Link Artificial Neural Network
Legendre polynomial

A B S T R A C T

In this paper, a new method based on single layer Legendre Neural Network (LeNN) model has been developed to solve initial and boundary value problems. In the proposed approach a Legendre polynomial based Functional Link Artificial Neural Network (FLANN) is developed. Nonlinear singular initial value problem (IVP), boundary value problem (BVP) and system of coupled ordinary differential equations are solved by the proposed approach to show the reliability of the method. The hidden layer is eliminated by expanding the input pattern using Legendre polynomials. Error back propagation algorithm is used for updating the network parameters (weights). Results obtained are compared with the existing methods and are found to be in good agreement.

© 2016 Published by Elsevier B.V.

1. Introduction

Q3

It is well known that the differential equations are back bone of physical systems. Many problems in engineering, mathematics, physics, economics etc. may be modeled by ordinary or partial differential equations [1–4]. In most cases analytical solutions of differential equations may not be obtained easily. So various numerical methods such as Runge–Kutta [5], predictor–corrector [6], finite difference, finite element [7], etc. have been developed to solve these equations. These numerical methods require the discretization of domain into the number of finite domains/points where the functions are approximated locally.

Recently, various machine intelligence methods in particular Artificial Neural Networks (ANN) are being used to solve initial and boundary value problems. The approximate solutions by ANN have many advantages. The trial solutions of ANN involve a single independent variable regardless of the dimension of the problem. The approximate solutions are continuous over all the domain of integration. Moreover, other numerical methods are usually iterative in nature, where we fix the step size before initiating the computation. After the solution is obtained, if we want to know the solution in between steps then again the procedure is to be repeated from initial stage. ANN may be one of the reliefs where we may overcome this repetition of iterations. Also we may use it as a black box to get numerical results at any arbitrary point in the domain.

In 1990, Lee and Kang [8] introduced a method to solve first order ordinary differential equation using Hopfield neural network models. Solution of linear and nonlinear ordinary differential equations using linear B_1 splines as basis function in feed forward neural network model has been approached by Meade and Fernandez [9,10]. Liu and Jammes [11] proposed a hybrid numerical method based on both neural network and optimization techniques to solve higher order ordinary differential equations. Lagaris et al. [12] used multi layer perceptron in their network architecture to solve both ordinary and partial differential equations. Malek and Beidokhti [13] solved higher order ordinary differential equations using artificial neural networks and optimization technique. An unsupervised version of kernel least mean square algorithm for solving first and second order ordinary differential equations has been developed by Yazdi et al. [14]. Selvaraju and Samant [15] proposed new algorithms based on neural network for solving matrix Riccati differential equations. Evolutionary algorithm with neural network training has been proposed by Aarts and Van der Veer [16] for solving partial differential equation and initial value problems. Another method for solving mixed boundary value problems on irregular domains have been implemented by Hoda and Nagla [17]. Shirvany et al. [18] used multilayer perceptron and radial basis function (RBF) neural networks with a new unsupervised training method to obtain numerical solution of non linear Schrodinger equation. Mcfall and Mahan [19] introduced an artificial neural network method for solution of mixed boundary value problems with irregular domain. A multi-quadric radial basis function neural network has been used to solve linear differential equations (ordinary and elliptic partial differential equations) by Mai-Duy and

* Corresponding author. Tel.: +91 661 2462713; fax: +91 661 2462713 2701.
E-mail address: sne_chak@yahoo.com (S. Chakraverty).

http://dx.doi.org/10.1016/j.asoc.2015.10.069
1568-4946/© 2016 Published by Elsevier B.V.

Tran-Cong [20]. Recently, Mall and Chakraverty [21,22] proposed regression based neural network model for solving lower as well as higher order ordinary differential equations. Also neural network method has been used by Ibraheem and Khalaf [23] to get the solution of boundary value problems. In another approach, Parisi et al. [24] steady-state heat transfer problem has been solved by using artificial neural network. Raja and Ahmad [25] implemented the solution of boundary value problems of one dimensional Bratu type equations using neural network.

A single layer Functional Link Artificial Neural Network (FLANN) model is introduced by Pao and Philips [26]. In FLANN, number of network parameters and number of iterations for training are less than that of multi layer perceptron (MLP) structure. In FLANN the hidden layer is replaced by a functional expansion block for enhancement of the input patterns using orthogonal polynomials such as Chebyshev, Legendre, etc. So single layer FLNN model is computationally efficient and having higher convergence speed. Chebyshev polynomial based Functional Link Artificial Neural Network has extensively applied to nonlinear dynamic system identification [27,28], digital communication [29], channel equalization [30], function approximation [31], etc.

1.1. Motivation

Here our target is to propose a single layer Legendre polynomial based Functional Link Artificial Neural Network called Legendre Neural Network (LeNN) to solve initial and boundary value problems. The Legendre Neural Network (LeNN) has been introduced by Yang and Tseng [32] for function approximation. Subsequently, LeNN has been used in channel equalization problems [33,34], system identification [35], for prediction of machinery noise [36], etc. To the best of our knowledge, present paper may be the first of its kind where we use Legendre Neural Network (LeNN) model in solving differential equations. We propose a single layer neural network with increasing the dimension of the input pattern using Legendre polynomials. A feed forward model with principle of error back propagation algorithm has been used here. Initial weights of the single layered network model are considered as random. Some of the advantages of the new single layer Legendre Neural Network (LeNN) based model for solving differential equations are as follows:

- It is a single layer neural network, so number of parameters is less than MLP.
- Simple implementation and easy computation.
- The hidden layers are removed.
- The back propagation algorithm is unsupervised.
- No optimization technique is used.

To validate the present method we have considered various numerical examples and a problem of astrophysics viz. Lane–Emden equation.

In astrophysics, nonlinear singular initial value problems which describe the thermal behavior of a spherical cloud of gas acting under the mutual attraction of its molecules and subject to the classical laws of thermodynamics had been proposed by Lane [37] and Emden [38]. The governing differential equation then was known as Lane–Emden type equations. Further Fowler [39] studied Lane–Emden type equations in greater detail. The Lane–Emden equation is generally expressed as

$$y'' + \frac{\alpha}{x}y' + f(x)h(y) = g(x) \quad x \geq 0, \quad \alpha \geq 0 \tag{1}$$

subject to $y(0) = a, y'(0) = b$where a, b are constants and $f(x)$, $h(y)$ and $g(x)$ are functions of x and y.

Lane–Emden type equations for $\alpha = 2, f(x) = 1$ and with some special form of $h(y)$ describes various phenomenon in astrophysics such as theory of stellar structure, thermal behavior of a spherical cloud of gas and theory of thermionic currents [40–42].

The Lane–Emden type equations are singular at $x = 0$. The solution of Lane–Emden equation and other nonlinear IVPs (as mentioned above) in astrophysics are challenging because of the singular point at the origin. So various techniques based on series solutions such as Adomian decomposition [43–45], variational iteration [46–48] and homotopy perturbation [49–52] methods have been used to handle the Lane–Emden equations.

Rest of the paper is organized as follows. In Section 2, we explain briefly the basic architecture of the Legendre Neural Network (LeNN) and its learning algorithm. Section 3 gives the details of problem formulation, construction of the appropriate form of LeNN trial solution and computation of gradient. Numerical examples and its results are presented in Section 4. In this section we also compare analytical and LeNN results and those are shown graphically. Conclusions are incorporated in the last section.

We now describe below the LeNN model and its learning method.

2. Legendre Neural Network (LeNN) model

In this section, we have introduced structure of single layered LeNN model and then the learning algorithm has been explained.

2.1. Structure of Legendre Neural Network (LeNN) model

Fig. 1 depicts the structure of single layer Legendre Neural Network (LeNN), which consists of single input node, one output layer and a functional expansion block based on Legendre polynomials. The hidden layer is eliminated by transforming the input pattern to

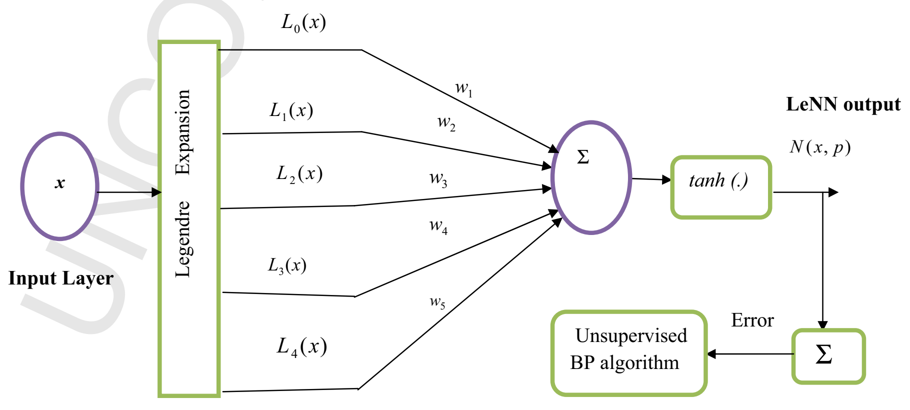

Fig. 1. Structure of Legendre Neural Network.

a higher dimensional space using Legendre polynomials. The Legendre polynomials are denoted by $L_n(u)$, here n is the order and $-1 < u < 1$ is the argument of the polynomial. Legendre polynomials constitute a set of orthogonal polynomials obtained as a solution of Legendre differential equation.

The first few Legendre polynomials are [35]

$$L_0(u) = 1$$
$$L_1(u) = u$$
$$L_2(u) = \frac{1}{2}(3u^2 - 1)$$

The higher order Legendre polynomials may be generated by the following recursive formula

$$L_{n+1}(u) = \frac{1}{n+1}[(2n+1)uL_n(u) - nL_{n-1}(u)] \tag{2}$$

As input data we consider a vector $x = (x_1, x_2, \ldots, x_h)$ of dimension h.

The enhanced pattern is obtained by using the Legendre polynomials

$$[1, L_1(x_1), L_2(x_1), L_3(x_1), \ldots, L_n(x_1); \quad 1, L_1(x_2), L_2(x_2), L_3(x_2), \ldots, L_n(x_2);$$
$$1, L_1(x_h), L_2(x_h), L_3(x_h), \ldots, L_n(x_h)]$$

Here h input data is expanded to n dimensional enhanced Legendre polynomials.

Architecture of the network with first five Legendre polynomials, single input (with one node) and output node are shown in Fig. 1.

2.2. Learning algorithm of Legendre Neural Network (LeNN)

Error back propagation learning algorithm is used for updating the network parameters (weights) of Legendre Neural Network (LeNN). As such, gradient of an error function with respect to the network parameters (weights) p is determined. The nonlinear tangent hyperbolic *tanh(.)* function is considered as activation function. The gradient descent algorithm is used for learning and the weights are updated by taking negative gradient at each iteration. The weights are initialized randomly and then the weights are update as follows

$$w_j^{k+1} = w_j^k + \Delta w_j^k = w_j^k + \left(-\eta \frac{\partial E(x,p)}{\partial w_j^k}\right) \tag{3}$$

where η is the learning parameter lies between 0 and 1, k is iteration step which is used to update the weights as usual in ANN and $E(x,p)$ is the error function.

In the next section, the procedure of applying ANN in the solution of differential equations is discussed.

3. Neural network model for solution of differential equations

In this section, we describe formulation of differential equations using ANN. In particular, the formulations for initial/boundary value problems and system of first order ordinary differential equations are incorporated in detail.

General form of differential equation (which represents ordinary as well as partial differential equations) may be written as [12]

$$G(x, y(x), \nabla y(x), \nabla^2 y(x), \ldots, \nabla^k y(x)) = 0 \quad x \in \bar{D} \subseteq R^n \tag{4}$$

subject to some initial or boundary conditions, where $y(x)$ is the solution, G is the function which defines the structure of the differential equation and ∇ is a differential operator. $\bar{D}$ denotes the discretized domain over finite set of points in R^n.

Let $y_t(x,p)$ denotes the trial solution with adjustable parameters (weights) p, thus the problem is transformed into the following minimization problem

$$\underset{p}{Min} \sum_{x_n \in \bar{D}} \left(G(x_n, y_t(x_n,p), \nabla y_t(x_n,p), \nabla^2 y_t(x_n,p), \ldots \nabla^k y_t(x_n,p))\right)^2. \tag{5}$$

The trial solution $y_t(x,p)$ satisfies the initial or boundary conditions and can be written as the sum of two terms:

$$y_t(x,p) = A(x) + F(x, N(x,p)) \tag{6}$$

where $A(x)$ satisfies initial or boundary conditions and contains no adjustable parameters. $N(x,p)$ is the single output of feed forward neural network with parameters p and input x The second term $F(x, N(x, p))$ makes no contribution to initial or boundary conditions but this is used to a neural network whose weights are adjusted to minimize the error function.

3.1. Proposed Legendre Neural Network (LeNN) algorithm

As mentioned above, the trial solution may be expressed as Eq. (6) where the first term $A(x)$ satisfies initial/boundary conditions. The second term viz. $F(x, N(x, p))$ contains single output $N(x,p)$ of LeNN model with one input node x (having h number of data) and adjustable parameters p.

Here

$$N(x,p) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{7}$$

$$\text{and} \quad z = \sum_{j=1}^{m} w_j v \quad j = 1, 2, \ldots, m \tag{8}$$

where x is the input data, $L_{j-1}(x)$ and w_j for $j = \{1, 2, 3, \ldots, m\}$ denote the expanded input data and the weight vectors respectively of the Legendre Neural Network.

General form of corresponding error function for the ODE may be formulated as

$$E(x,p) = \sum_{i=1}^{h} \frac{1}{2} \left\{ \frac{d^n y_t(x_i,p)}{dx^n} - f\left(x_i, y_t(x_i), \frac{dy_t(x_i,p)}{dx}, \ldots, \frac{d^{n-1} y_t(x_i,p)}{dx^{n-1}}\right) \right\}^2. \tag{9}$$

As discussed above for LeNN, x_is', $i = 1, 2, \ldots, h$ are the input data and the weights w_j from input to output layer are modified according to the back propagation learning algorithm given in Eq. (3).

Here

$$\frac{\partial E(x,p)}{\partial w_j}$$
$$= \frac{\partial}{\partial w_j}\left(\sum_{i=1}^{h} \frac{1}{2} \left\{ \frac{d^n y_t(x_i,p)}{dx^n} - f\left(x_i, y_t(x_i), \frac{dy_t(x_i,p)}{dx}, \ldots, \frac{d^{n-1} y_t(x_i,p)}{dx^{n-1}}\right) \right\}^2\right) \tag{10}$$

Below we consider some particular cases

3.1.1. First order ODE

The first order ODE may be represent as

$$\frac{dy}{dx} = f(x,y) \quad x \in [a,b] \tag{11}$$

subject to $y(a)=A$

The related LeNN trial solution is

$$y_t(x,p) = A + (x-a)N(x,p) \tag{12}$$

The error function is written as

$$E(x) = \sum_{i=1}^{h} \frac{1}{2}\left(\frac{dy_t(x_i,p)}{dx} - f[x_i, y_t(x_i,p)]\right)^2 \tag{13}$$

Derivative of $y_t(x,p)$ with respect to x is given as

$$\frac{dy_t(x,p)}{dx} = (x-a)N(x,p) + \frac{dN}{dx} \tag{14}$$

3.1.2. Second order ODE

Let us consider second order initial value problem as

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right) \quad x \in [a,b] \tag{15}$$

with initial conditions $y(a)=A$, $y'(a)=A'$

The LeNN trial solution may be expressed as

$$y_t(x,p) = A + A'(x-a) + (x-a)^2 N(x,p) \tag{16}$$

The trial solution $y_t(x,p)$ satisfies the initial conditions and the error function may be computed as follows

$$E(x) = \sum_{i=1}^{h} \frac{1}{2}\left(\frac{d^2y_t(x_i,p)}{dx^2} - f\left[x_i, y_t(x_i,p), \frac{dy_t(x_i,p)}{dx}\right]\right)^2 \tag{17}$$

From Eq. (17) we get (by differentiating)

$$\frac{dy_t(x,p)}{dx} = A' + 2(x-a)N(x,p) + (x-a)^2\frac{dN}{dx} \tag{18}$$

$$\frac{d^2y_t(x,p)}{dx^2} = 2N(x,p) + 4(x-a)\frac{dN}{dx} + (x-a)^2\frac{d^2N}{dx^2} \tag{19}$$

Next, a second order boundary value problem may be written as

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right) \quad x \in [a,b] \tag{20}$$

with boundary conditions $y(a)=A$, $y(b)=B$

Corresponding LeNN trial solution for the above boundary value problem is

$$y_t(x,p) = \frac{bA-aB}{b-a} + \frac{B-A}{b-a}x + (x-a)(x-b)N(x,p) \tag{21}$$

As such, the error function may be obtained as

$$E(p) = \sum_{i=1}^{h} \frac{1}{2}\left(\frac{d^2y_t(x_i,p)}{dx^2} - f\left(x_i \cdot y_t(x_i,p), \frac{dy_t(x_i,p)}{dx}\right)\right)^2 \tag{22}$$

here $$\frac{dy_t(x,p)}{dx} = \frac{B-A}{b-a} + (x-b)N(x,p) + (x-a)N(x,p) + (x-a)(x-b)\frac{dN}{dx} \tag{23}$$

3.1.3. System of first order ODEs

Here we consider the following system of initial value first order differential equations

$$\frac{dy_r}{dx} = f_r(x, y_1, \ldots, y_\ell) \quad r = 1, 2, \ldots, \ell \quad \text{and} \quad x \in [a,b] \tag{24}$$

subject to $y_r(a)=A_r$, $r=1, 2, \ldots, \ell$

Corresponding trial solution has the following form

$$y_{t_r}(x,p_r) = A_r + (x-a)N_r(x,p_r) \quad \forall r = 1, 2, \ldots, \ell \tag{25}$$

For each r, $N_r(x,p_r)$ is the output of the Legendre Neural Network with input x and parameter p_r.

Then the corresponding error function with adjustable network parameters may be written as

$$E(x) = \sum_{i=1}^{h}\sum_{r=1}^{\ell} \frac{1}{2}\left[\frac{dy_{t_r}(x_i,p_r)}{dx} - f_r\left(x_i, y_{t_1}(x_i,p_1), \ldots, y_{t_\ell}(x_i,p_\ell)\right)\right]^2 \tag{26}$$

From Eq. (26) we have

$$\frac{dy_{t_r}(x,p_r)}{dx} = (x-a)N_r(x,p_r) + \frac{dN_r}{dx} \quad \forall r = 1, 2, \ldots, \ell \tag{27}$$

3.2. Computation of gradient for LeNN

The error computation involves both output and derivatives of the network output with respect to the corresponding inputs. For minimizing the error function $E(x,p)$ we differentiate $E(x,p)$ with respect to the network parameters. Then the gradient of network output with respect to their inputs is computed as below.

The derivatives of $N(x,p)$ with respect to input x is expressed as

$$\frac{dN}{dx} = \sum_{j=1}^{m}\left[\left(\frac{\left(e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}\right)^2 - \left(e^{(w_jL_{j-1}(x))} - e^{-(w_jL_{j-1}(x))}\right)^2}{\left(e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}\right)^2}\right)\right] \times \left(w_jL'_{j-1}(x)\right) \tag{28}$$

Simplifying, Eq. (28) we have

$$\frac{dN}{dx} = \sum_{j=1}^{m}\left[1 - \left(\frac{e^{(w_jL_{j-1}(x))} - e^{-(w_jL_{j-1}(x))}}{e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}}\right)^2\right](w_jL'_{j-1}(x)) \tag{29}$$

Similarly we can compute the second derivative of $N(x,p)$ as

$$\frac{d^2N}{dx^2} = \sum_{j=1}^{m}\left[\frac{d}{dx}\left\{1 - \left(\frac{e^{(w_jL_{j-1}(x))} - e^{-(w_jL_{j-1}(x))}}{e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}}\right)^2\right\}(w_jL'_{j-1}(x)) + \frac{d}{dx}(w_jL'_{j-1}(x))\left\{1 - \left(\frac{e^{(w_jL_{j-1}(x))} - e^{-(w_jL_{j-1}(x))}}{e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}}\right)^2\right\}\right] \tag{30}$$

After simplifying the above we get

$$\frac{d^2N}{dx^2} = \sum_{j=1}^{m}\left[\left(\left\{2\left(\frac{e^{(w_jL_{j-1}(x))} - e^{-(w_jL_{j-1}(x))}}{e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}}\right)^3 - 2\left(\frac{e^{(w_jL_{j-1}(x))} - e^{-(w_jL_{j-1}(x))}}{e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}}\right)\right\}(w_jL'_{j-1}(x))^2\right) + \left(\left\{1 - \left(\frac{e^{(w_jL_{j-1}(x))} - e^{-(w_jL_{j-1}(x))}}{e^{(w_jL_{j-1}(x))} + e^{-(w_jL_{j-1}(x))}}\right)^2\right\}(w_jL''_{j-1}(x))\right)\right] \tag{31}$$

where w_j, $L'_{j-1}(x)$ and $L''_{j-1}(x)$ denote weights of network, first and second derivatives of Legendre polynomials respectively.

Next we include detail of the above procedure for traditional Multi Layer Perceptron (MLP)

3.3. Gradient computation for traditional (MLP) neural network

Let us consider a three layered neural network with one input node x with h number of data, a hidden layer with u nodes and one output node. For the given input x the output $N(x,p)$ may be formulated as [12]

$$N(x,p)=\sum_{t=1}^{u} v_t \tanh(r_t) \tag{32}$$

where $r_t=w_t x+b_t$, w_t denotes the weight from input unit to the hidden unit t, v_t denotes weight from the hidden unit t to the output unit, b_t is the biases and $\tanh(r_t)$ is the tangent hyperbolic activation function.

The derivative of $N(x,p)$ with respect to input x is

$$\frac{dN}{dx}=\sum_{t=1}^{u} v_t w_t(\tanh(r_t))'. \tag{33}$$

Then kth order derivative of $N(x,p)$ may be expressed as

$$\frac{d^kN}{dx^k}=\sum_{t=1}^{u} v_t w_t^k(\tanh(r_t))^{(k)} \tag{34}$$

where $(\tanh(r_t))^{(k)}$ denotes the kth order derivative of tangent hyperbolic function.

Let N_ϑ denotes the derivative of the network with respect to its inputs and then we have the relations [12]

$$N_\vartheta = D^kN=\sum_{t=1}^{u} v_t \mathrm{P}_t(\tanh(r_t))^{(k)} \tag{35}$$

where $\mathrm{P}_t=\prod_{k=1}^{n} w_t^{\lambda_k}$ and $\sigma=\sum_{k=1}^{n}\lambda_k$

The derivative of N_ϑ with respect to other parameters may be obtained as

$$\frac{\partial N_\vartheta}{\partial v_t}=\mathrm{P}_t\tanh(r_t)^{(\sigma)} \quad \text{for} \quad t=1,\ldots,u \tag{36}$$

$$\frac{\partial N_\vartheta}{\partial b_t}=v_t\mathrm{P}_t(\tanh(r_t))^{(\sigma+1)} \tag{37}$$

$$\frac{\partial N_\vartheta}{\partial w_t}=xv_t\mathrm{P}_t(\tanh(r_t))^{(\sigma+1)}+v_t\lambda_k w_t^{\sigma_t-1}\left(\prod_{k=1,k\neq t} w_t^{\lambda_k}\right)(\tanh(r_t))^{(\sigma)} \tag{38}$$

In this paper, we have considered five nodes for the hidden layer (in MLP), one input node x having h number of data and one output node.

In the similar way we may use unsupervised error back propagation algorithm for updating the network parameters (weights and biases) from input layer to hidden and from hidden to output layer as below

$$w_t^{k+1}=w_t^k+\Delta w_t^k=w_t^k+\left(-\eta\frac{\partial E(x,p)^k}{\partial w_t^k}\right). \tag{39}$$

$$v_t^{k+1}=v_t^k+\Delta v_t^k=v_t^k+\left(-\eta\frac{\partial E(x,p)^k}{\partial v_t^k}\right). \tag{40}$$

For nth order ODE

$$\frac{\partial E(x,p)}{\partial w_t}=\frac{\partial}{\partial w_t}\left(\sum_{i=1}^{h}\frac{1}{2}\left\{\frac{d^n y_t(x_i,p)}{dx^n}-f\left(x_i,y_t(x_i),\frac{dy_t(x_i,p)}{dx},\ldots,\frac{d^{n-1}y_t(x_i,p)}{dx^{n-1}}\right)\right\}^2\right) \tag{41}$$

$$\frac{\partial E(x,p)}{\partial v_t}=\frac{\partial}{\partial v_t}\left(\sum_{i=1}^{h}\frac{1}{2}\left\{\frac{d^n y_t(x_i,p)}{dx^n}-f\left(x_i,y_t(x_i),\frac{dy_t(x_i,p)}{dx},\ldots,\frac{d^{n-1}y_t(x_i,p)}{dx^{n-1}}\right)\right\}^2\right) \tag{42}$$

For system of first order ODEs (Eq. (26))

$$\begin{aligned}\frac{\partial E(x,p)}{\partial w_t}=\frac{\partial}{\partial w_t}\Bigg(\sum_{i=1}^{h}\frac{1}{2}\Bigg[&\left\{\frac{dy_{t_1}(x_i,p_1(x_i,y_{t_1}(x_i,p_1),\ldots,y_{t_\ell}(x_i,p_\ell)))}{dx}-f_1\right\}\\&+\left\{\frac{dy_{t_2}(x_i,p_2(x_i,y_{t_1}(x_i,p_1),\ldots,y_{t_\ell}(x_i,p_\ell)))}{dx}-f_2\right\}\\&+\cdots+\left\{\frac{dy_{t_\ell}(x_i,p_\ell(x_i,y_{t_1}(x_i,p_1),\ldots,y_{t_\ell}(x_i,p_\ell)))}{dx}-f_l\right\}\Bigg]\Bigg)^2\end{aligned} \tag{43}$$

$$\begin{aligned}\frac{\partial E(x,p)}{\partial v_t}=\frac{\partial}{\partial v_t}\Bigg(\sum_{i=1}^{h}\frac{1}{2}\Bigg[&\left\{\frac{dy_{t_1}(x_i,p_1(x_i,y_{t_1}(x_i,p_1),\ldots,y_{t_\ell}(x_i,p_\ell)))}{dx}-f_1\right\}\\&+\left\{\frac{dy_{t_2}(x_i,p_2(x_i,y_{t_1}(x_i,p_1),\ldots,y_{t_\ell}(x_i,p_\ell)))}{dx}-f_2\right\}\\&+\cdots+\left\{\frac{dy_{t_\ell}(x_i,p_\ell(x_i,y_{t_1}(x_i,p_1),\ldots,y_{t_\ell}(x_i,p_\ell)))}{dx}-f_l\right\}\Bigg]\Bigg)^2\end{aligned} \tag{44}$$

4. Numerical results and discussion

In this section, we consider various example problems viz. a nonlinear singular initial value problem, a boundary value problem and a system of coupled first order ordinary differential equations. It is worth mentioning that MATLAB code has been written for the present LeNN model and results are computed for various example problems.

Example 1. Let us take a nonlinear singular initial value problem of Lane–Emden type equation [48]

$$\frac{d^2y}{dx^2}+\frac{2}{x}\frac{dy}{dx}+4(2e^y+e^{y/2})=0$$

with initial conditions $y(0)=0$, $y'(0)=0$

The exact solution for above equation is

$$y(x)=-2\ln(1+x^2)$$

Following the procedure of the present method, we write the LeNN trial solution

$$y_t(x,p)=x^2N(x,p)$$

Table 1
Comparison among Analytical, LeNN and MLP results (Example 1).

Input data	Analytical [48]	LeNN	MLP
0.0000	0.0000	0.0000	0.0000
0.1000	−0.0199	−0.0195	−0.0191
0.2000	−0.0784	−0.0785	−0.0778
0.3000	−0.1724	−0.1725	−0.1782
0.4000	−0.2968	−0.2965	−0.3000
0.5000	−0.4463	−0.4468	−0.4421
0.6000	−0.6150	−0.6135	−0.6145
0.7000	−0.7976	−0.7975	−0.7990
0.8000	−0.9894	−0.9896	−0.9905
0.9000	−1.1867	−1.1869	−1.1839
1.0000	−1.3863	−1.3861	−1.3857

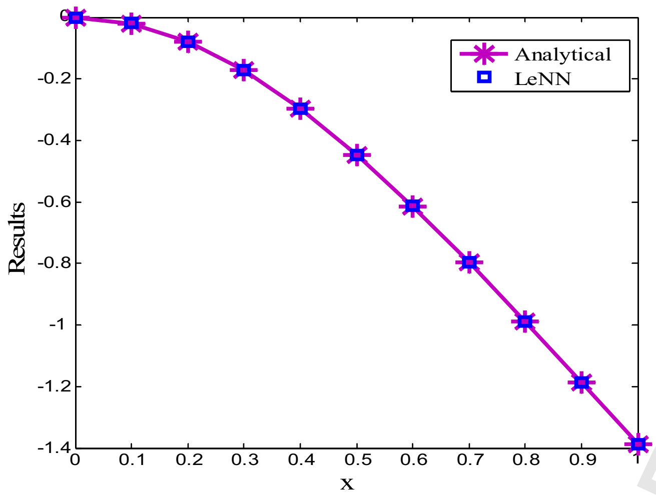

Fig. 2. Plot of Analytical and LeNN results (Example 1).

Ten equidistant points in [0,1] and five weights with respect to first five Legendre polynomials are considered. Comparison among analytical, Legendre neural (LeNN) and multi layer perceptron (MLP) results has been shown in Table 1. These comparisons are also depicted in Fig. 2. Plot of the error function between analytical and LeNN results is cited in Fig. 3. Finally results for some testing points are shown in Table 2. This testing is done to check whether the converged LeNN can give results directly by inputting the points which were not taken during training.

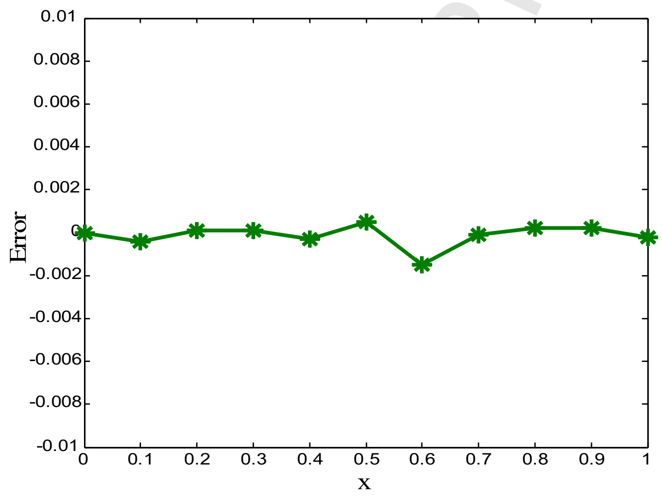

Fig. 3. Error plot between Analytical and LeNN results (Example 1).

Table 2
Analytical and LeNN results for testing points (Example 1).

Testing points	0.2040	0.4863	0.5191	0.7066	0.9837
Analytical [48]	−0.0815	−0.4245	−0.4772	−0.8100	−1.3537
LeNN	−0.0820	−0.4244	−0.4774	−0.8104	−1.3562

Table 3
Comparison among Analytical, LeNN and MLP results (Example 2).

Input data	Analytical [23]	LeNN	MLP
1.0000	1.0000	1.0000	1.0000
1.1000	1.0476	1.0475	1.0471
1.2000	1.0909	1.0919	1.0900
1.3000	1.1304	1.1291	1.1310
1.4000	1.1667	1.1663	1.1676
1.5000	1.2000	1.2001	1.1943
1.6000	1.2308	1.2302	1.2315
1.7000	1.2593	1.2590	1.2602
1.8000	1.2857	1.2858	1.2874
1.9000	1.3103	1.3100	1.3119
2.0000	1.3333	1.3333	1.3340

Example 2. We consider now a nonlinear boundary value problem [23]

$$\frac{d^2y}{dx^2} = \frac{1}{2x^2}(y^3 - 2y^2)$$

with boundary conditions $y(1) = 1$, $y(2) = 4/3$

The related LeNN trial solution is written as

$$y_t(x, p) = \frac{2}{3} + \frac{1}{3}x + (x-1)(x-2)N(x, p)$$

We train the network for ten equidistant points in the domain [1,2] with first five Legendre polynomials. Table 3 shows comparison among analytical, LeNN and MLP results. Comparison between analytical and LeNN results are also depicted in Fig. 4. Fig. 5 shows the error (between analytical and LeNN). Similar to previous example, the converged LeNN is used then to have the results for some testing points. As such Table 4 incorporates corresponding results directly by using the converged weights.

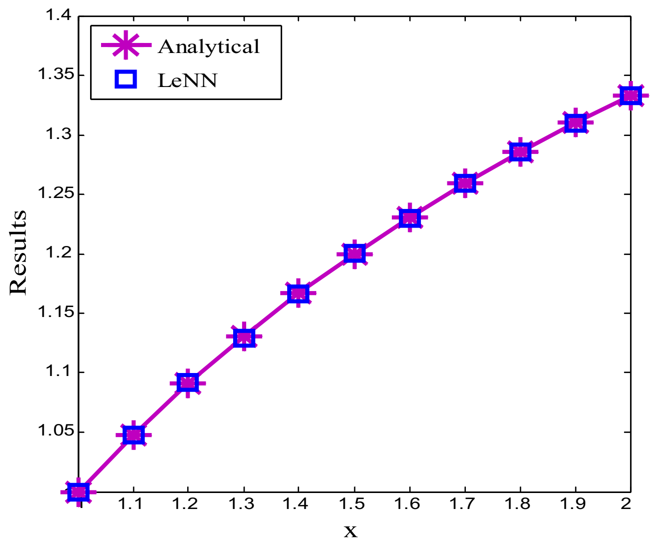

Fig. 4. Plot of Analytical and LeNN results (Example 2).

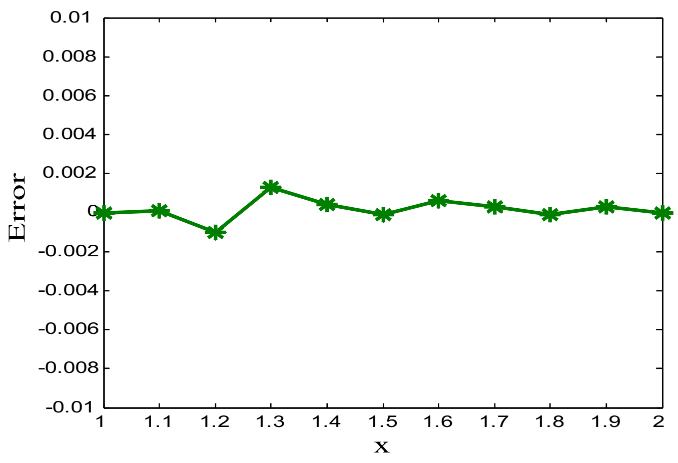

Fig. 5. Error plot between Analytical and LeNN results (Example 2).

Table 4
Analytical, LeNN and MLP results for testing points (Example 2).

Testing points	1.1320	1.3671	1.5980	1.8021	1.9540
Analytical [23]	1.0619	1.1551	1.2302	1.2862	1.3230
LeNN	1.0620	1.1554	1.2300	1.2859	1.3231
MLP	1.0625	1.1568	1.2289	1.2831	1.3216

Example 3. Next we take a system of coupled first order ordinary differential equations [12] Q4

$$\left.\begin{aligned} \frac{dy_1}{dx} &= \cos(x) + y_1^2 + y_2 - (1 + x^2 + \sin^2(x)) \\ \frac{dy_2}{dx} &= 2x - (1 + x^2)\sin(x) + y_1 y_2 \end{aligned}\right\} \quad x \in [0, 2]$$

with initial conditions $y_1(0) = 0$ and $y_2(0) = 1$

Corresponding analytical solutions are

$$\left.\begin{aligned} y_1(x) &= \sin(x) \\ y_2(x) &= 1 + x^2 \end{aligned}\right\}$$

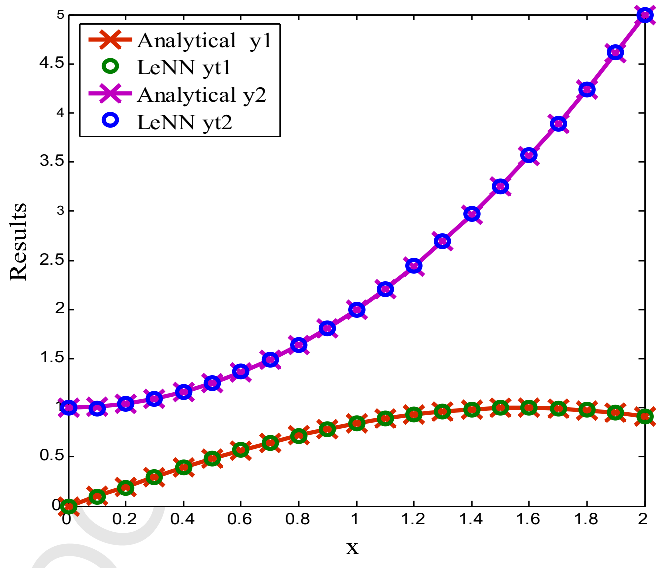

Fig. 6. Plot of Analytical and LeNN results (Example 3).

In this case, the trial LeNN solutions are

$$\left.\begin{aligned} y_{t_1}(x) &= xN_1(x, p_1) \\ y_{t_2}(x) &= 1 + xN_2(x, p_2) \end{aligned}\right\}$$

Here we consider twenty equidistant points in [0,2] and the results are compared between analytical and LeNN results. Comparison among analytical $(y_1(x) y_2(x))$, Legendre neural $(y_{t_1}(x) y_{t_2}(x))$ and MLP results are given in Table 5. Analytical and LeNN results are compared in Fig. 6 and the error plot is depicted in Fig. 7.

Example 4. Finally we consider another system of coupled first order ordinary differential equations

$$\left.\begin{aligned} \frac{dy_1}{dx} &= \frac{\cos(x) - \sin(x)}{y_2} \\ \frac{dy_2}{dx} &= y_1 y_2 + e^x - \sin(x) \end{aligned}\right\} \quad x \in [0, 2]$$

subject to $y_1(0) = 0$ and $y_2(0) = 1$

Table 5
Comparison among Analytical, LeNN and MLP results (Example 3).

Input data x_i	Analytical [12] $y_1(x)$	LeNN $y_{t_1}(x)$	MLP $y_{t_1}(x)$	Analytical [12] $y_2(x)$	LeNN $y_{t_2}(x)$	MLP $y_{t_2}(x)$
0	0	0	0.0001	1.0000	1.0000	1.0000
0.1000	0.0998	0.0995	0.1019	1.0100	0.9862	1.0030
0.2000	0.1987	0.1856	0.2027	1.0400	1.0397	1.0460
0.3000	0.2955	0.2970	0.2998	1.0900	1.0908	1.0973
0.4000	0.3894	0.3892	0.3908	1.1600	1.1603	1.1624
0.5000	0.4794	0.4793	0.4814	1.2500	1.2500	1.2513
0.6000	0.5646	0.5679	0.5689	1.3600	1.3687	1.3628
0.7000	0.6442	0.6422	0.6486	1.4900	1.4894	1.4921
0.8000	0.7174	0.7152	0.7191	1.6400	1.6415	1.6425
0.9000	0.7833	0.7833	0.7864	1.8100	1.8106	1.8056
1.0000	0.8415	0.8391	0.8312	2.0000	1.9992	2.0046
1.1000	0.8912	0.8919	0.8897	2.2100	2.2084	2.2117
1.2000	0.9320	0.9327	0.9329	2.4400	2.4418	2.4383
1.3000	0.9636	0.9633	0.9642	2.6900	2.6932	2.6969
1.4000	0.9854	0.9857	0.9896	2.9600	2.9689	2.9640
1.5000	0.9975	0.9974	0.9949	3.2500	3.2498	2.2542
1.6000	0.9996	0.9962	0.9960	3.5600	3.5705	3.5679
1.7000	0.9917	0.9911	0.9907	3.8900	3.8911	3.8970
1.8000	0.9738	0.9800	0.9810	4.2400	4.2402	4.2468
1.9000	0.9463	0.9464	0.9470	4.6100	4.6129	4.6209
2.0000	0.9093	0.9096	0.9110	5.0000	4.9995	5.0012

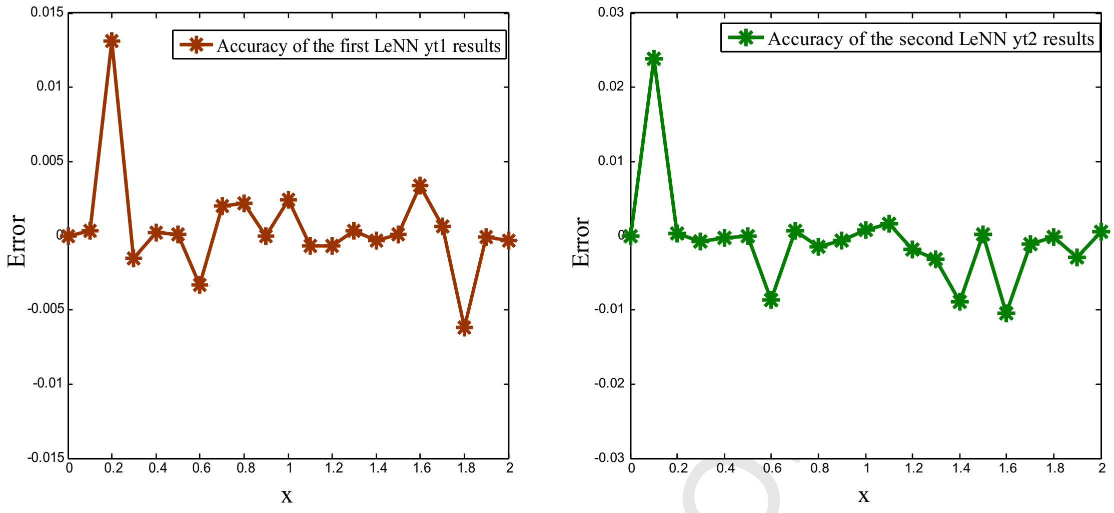

Fig. 7. Error plot between Analytical and LeNN results (Example 3).

Table 6
Comparison among Analytical, LeNN and MLP results (Example 4).

Input data x_i	Analytical $y_1(x)$	LeNN $y_{t_1}(x)$	MLP $y_{t_1}(x)$	Analytical $y_2(x)$	LeNN $y_{t_2}(x)$	MLP $y_{t_2}(x)$
0	0	0	0	1.0000	1.0000	1.0000
0.1000	0.0903	0.0907	0.0899	1.1052	1.1063	1.1045
0.2000	0.1627	0.1624	0.1667	1.2214	1.2219	1.2209
0.3000	0.2189	0.2199	0.2163	1.3499	1.3505	1.3482
0.4000	0.2610	0.2609	0.2625	1.4918	1.5002	1.4999
0.5000	0.2908	0.2893	0.2900	1.6487	1.6477	1.6454
0.6000	0.3099	0.3088	0.3111	1.8221	1.8224	1.8209
0.7000	0.3199	0.3197	0.3205	2.0138	2.0158	2.0183
0.8000	0.3223	0.3225	0.3234	2.2255	2.2246	2.2217
0.9000	0.3185	0.3185	0.3165	2.4596	2.4594	2.4610
1.0000	0.3096	0.3093	0.3077	2.7183	2.7149	2.7205
1.1000	0.2967	0.2960	0.2969	3.0042	3.0043	3.0031
1.2000	0.2807	0.2802	0.2816	3.3201	3.3197	3.3211
1.3000	0.2626	0.2632	0.2644	3.6693	3.6693	3.6704
1.4000	0.2430	0.2431	0.2458	4.0552	4.0549	4.0535
1.5000	0.2226	0.2229	0.2213	4.4817	4.4819	4.4822
1.6000	0.2018	0.2017	0.2022	4.9530	4.9561	4.9557
1.7000	0.1812	0.1818	0.1789	5.4739	5.4740	5.4781
1.8000	0.1610	0.1619	0.1605	6.0496	6.0500	6.0510
1.9000	0.1415	0.1416	0.1421	6.6859	6.6900	6.6823
2.0000	0.1231	0.1230	0.1226	7.3891	7.3889	7.3857

Analytical solutions for the above may be obtained as

$$\left.\begin{aligned} y_1(x) &= \frac{\sin(x)}{e^x} \\ y_2(x) &= e^x \end{aligned}\right\}$$

Corresponding trial LeNN solutions are

$$\left.\begin{aligned} y_{t_1}(x) &= xN_1(x, p_1) \\ y_{t_2}(x) &= 1 + xN_2(x, p_2) \end{aligned}\right\}$$

The network is trained here for twenty equidistant points in the given domain. As in previous cases the analytical, LeNN and MLP results are shown in Table 6. Comparisons between analytical $(y_1(x)y_2(x))$ and LeNN $(y_{t_1}(x)y_{t_2}(x))$ results are depicted in Fig. 8 and are found to be in excellent agreement. Plot of error function is cited in Fig. 9. Lastly, LeNN solutions for some testing points are given in Table 7.

The CPU time of computation for the proposed LeNN model and tradition neural network (MLP) are incorporated in Table 8. As such we may see that LeNN takes less time of computation than traditional MLP.

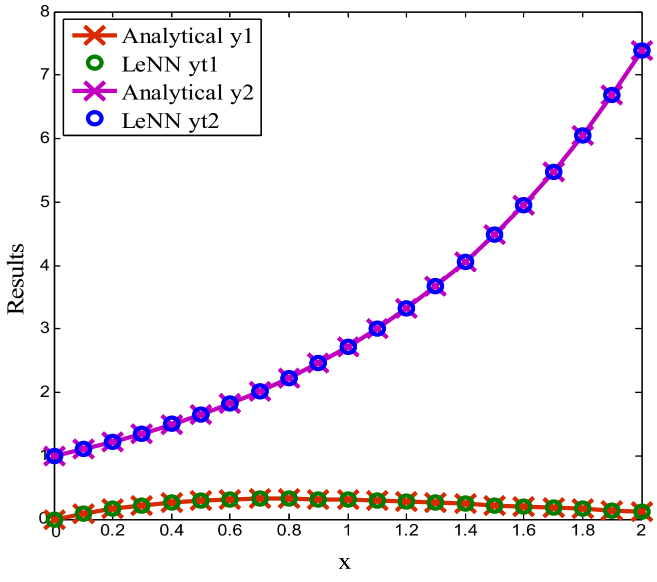

Fig. 8. Plot of Analytical and LeNN results (Example 4).

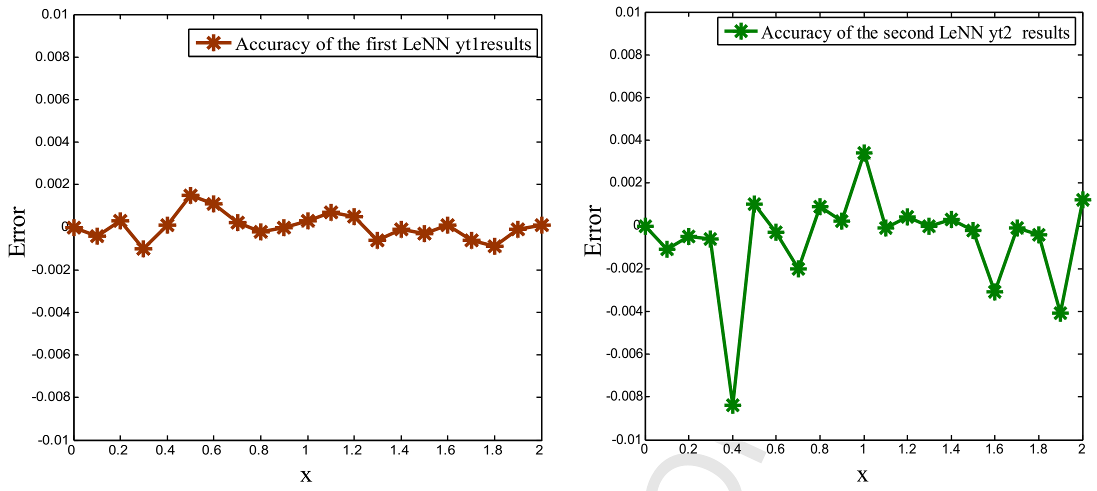

Fig. 9. Error plot between Analytical and LeNN results (Example 4).

Table 7
Analytical, LeNN results for testing points (Example 4).

Testing points	0.3894	0.7120	0.9030	1.2682	1.5870	1.8971
Analytical y1	0.2572	0.3206	0.3183	0.2686	0.2045	0.1421
LeNN yt1	0.2569	0.3210	0.3180	0.2689	0.2045	0.1420
Analytical y2	1.4761	2.0381	2.4670	3.5544	4.8891	6.6665
LeNN yt2	1.4760	2.0401	2.4672	3.5542	4.8894	6.6661

Table 8
CPU time of computation.

CPU time of computation (in s)	Example 1	Example 2	Example 3	Example 4
MLP	11,849.45	810,871	11,987.25	12,368.15
LeNN	9170.89	721,238	10,723.09	10,288.26

5. Conclusion

In this study, we have proposed a single layer Legendre Neural Network (LeNN) model to solve ordinary differential equations viz. nonlinear singular initial value problem of Lane–Emden type, second order boundary value problem and system of coupled first order ODEs. Here we have considered single layer Functional Link Artificial Neural Network (FLANN) architecture. In FLANN, the hidden layer is replaced by functional expansion block for enhancement of the input patterns. The dimension of input data is expanded using set of Legendre orthogonal polynomials. A feed forward neural network with error back propagation algorithm is used for updating the network parameters (weights). The numbers of network parameters of Legendre Neural Network (LeNN) are less than the multi layer neural network. So computational complexity of LeNN model is less than that of MLP. Excellent agreement of the results between analytical and LeNN show the powerfulness and reliability of the proposed method.

Acknowledgement

Q5 The first author would like to thank the Department of Science and Technology (DST), Government of India for financial support under Women Scientist Scheme-A. Also the authors would like to acknowledge Editor in chief and the Reviewers for their valuable suggestions to improve our work.

References

[1] H.J. Ricardo, A Modern Introduction to Differential Equations, second edition, Elsevier, 2009.
[2] Y. Pinchover, J. Rubinstein, An Introduction to Partial Differential Equation, Published in the United States of America by Cambridge University Press, New York, 2005.
[3] C.J. Budd, A. Iserles, Geometric integration: numerical solution of differential equations on manifolds, Philos. Trans. R. Soc.: Math. Phys. Eng. Sci. 357 (1999) 945–956. Q6
[4] R. Norberg, Differential equations for moments of present values in life insurance, Insur.: Math. Econ. 17 (1995) 171–180.
[5] A. Wambecq, Rational Runge–Kutta methods for solving systems of ordinary differential equations, Computing 20 (1978) 333–342.
[6] J. Douglas, B.F. Jones, Predictor–corrector methods for nonlinear parabolic differential equations, J. Ind. Appl. Math. 11 (1963) 195–204.
[7] J.N. Reddy, An Introduction to the Finite Element Method, McGraw-Hill, 1993.
[8] H. Lee, I. Kang, Neural algorithms for solving differential equations, J. Comput. Phys. 91 (1990) 110–117.
[9] A.J. Meade Jr., A.A. Fernandez, The numerical solution of linear ordinary differential equations by feed forward neural networks, Math. Comput. Model. 19 (1994) 1–25.
[10] A.J. Meade Jr., A.A. Fernandez, Solution of nonlinear ordinary differential equations by feed forward neural networks, Math. Comput. Model. 20 (1994) 19–44.
[11] Bo-An Liu, B. Jammes, Solving ordinary differential equations by neural networks, in: Proceeding of 13th European Simulation Multi-Conference Modelling and Simulation: A Tool for the Next Millennium, Warsaw, Poland, June 1–4, 1999.
[12] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (1998) 987–1000.
[13] A. Malek, R.S. Beidokhti, Numerical solution for high order deferential equations, using a hybrid neural network-optimization method, Appl. Math. Comput. 183 (2006) 260–271.
[14] H.S. Yazdi, M. Pakdaman, H. Modaghegh, Unsupervised kernel least mean square algorithm for solving ordinary differential equations, Nerocomputing 74 (2011) 2062–2071.
[15] N. Selvaraju, J. Abdul Samant, Solution of matrix Riccati differential equation for nonlinear singular system using neural networks, Int. J. Comput. Appl. 29 (2010) 48–54.
[16] L.P. Aarts, P. Van der Veer, Neural network method for solving partial differential equations, Neural Process. Lett. 14 (2001) 261–271.
[17] S.A. Hoda, H.A. Nagla, Neural network methods for mixed boundary value problems, Int. J. Nonlinear Sci. 11 (2011) 312–316.
[18] Y. Shirvany, M. Hayati, R. Moradian, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations, Appl. Soft Comput. 9 (2009) 20–29.
[19] K. McFall, J.R. Mahan, Artificial neural network for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions, IEEE Trans. Neural Netw. 20 (2009) 1221–1233.
[20] N. Mai-Duy, T. Tran-Cong, Numerical solution of differential equations using multi quadric radial basis function networks, Neural Netw. 14 (2001) 185–199.
[21] S. Mall, S. Chakraverty, Regression based neural network training for the solution of ordinary differential equations, Int. J. Math. Model. Numer. Optim. 4 (2013) 136–149.
[22] S. Chakraverty, S. Mall, Regression based weight generation algorithm in neural network for solution of initial and boundary value problems, Neural Comput. Appl. 25 (2014) 585–594.

[23] K.I. Ibraheem, B.M. Khalaf, Shooting neural networks algorithm for solving boundary value problems in ODEs, Appl. Appl. Math. 6 (2011) 1927–1941.
[24] D.R. Parisi, M.C. Mariani, M.A. Laborde, Solving differential equations with unsupervised neural networks, Chem. Eng. Process.: Process Intensif. 42 (2003) 715–721.
[25] M.A.Z. Raja, S.I. Ahmad, Numerical treatment for solving one-dimensional Bratu problem using neural network, Neural Comput. Appl. 24 (2014) 549–561.
[26] Y.H. Pao, S.M. Philips, The functional link net and learning optimal control, Neurocomputing 9 (1995) 149–164.
[27] S. Purwar, I.N. Kar, A.N. Jha, Online system identification of complex systems using Chebyshev neural network, Appl. Soft Comput. 7 (2007) 364–372.
[28] J.C. Patra, A.C. Kot, Nonlinear dynamic system identification using Chebyshev functional link artificial neural network, IEEE Trans. Syst. Man Cybern. Part B-Cybern. 32 (4) (2002) 505–511.
[29] J.C. Patra, M. Juhola, P.K. Meher, Intelligent sensors using computationally efficient Chebyshev neural networks, IET Sci. Meas. Technol. 2 (2008) 68–75.
[30] W.D. Weng, C.S. Yang, R.C. Lin, A channel equalizer using reduced decision feedback Chebyshev functional link artificial neural networks, Inf. Sci. 177 (2007) 2642–2654.
[31] T.T. Lee, J.T. Jeng, The Chebyshev-polynomials-based unified model neural networks for function approximation, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 28 (1998) 925–935.
[32] S.S. Yang, C.S. Tseng, An orthogonal neural network for function approximation, IEEE Trans. Syst. Man Cybern. Part-B 26 (5) (1996) 779–785.
[33] J.C. Patra, W.C. Chin, P.K. Meher, G. Chakraborty, Legendre-FLANN-based nonlinear channel equalization in wireless, IEEE Int. Conf. Syst. Man Cybern. (2008) 1826–1831.
[34] J.C. Patra, P.K. Meher, G. Chakraborty, Nonlinear channel equalization for wireless communication systems using Legendre neural networks, Signal Process. 89 (11) (2009) 2251–2262.
[35] J.C. Patra, C. Bornand, Nonlinear dynamic system identification using Legendre Neural Network, IEEE Int. Joint Conf. (2010).
[36] S.K. Nanda, D.P. Tripathy, Application of functional link artificial neural network for prediction of machinery noise in opencast mines, Adv. Fuzzy Syst. (2011).
[37] J.H. Lane, On the theoretical temperature of the sun under the hypothesis of a gaseous mass maintaining its volume by its internal heat and depending on the laws of gases known to terrestrial experiment, Am. J. Sci. Arts 50 (2) (1870) 57–74.
[38] E.R. Gaskugeln, Anwendungen der mechanischenWarmen-theorie auf Kosmologie and meteorologische Problem, Teubner, Leipzig, 1907.
[39] R.H. Fowler, The form near infinity of real, continuous solutions of a certain differential equation of the second order, Q. J. Math. (Oxford) 45 (1914) 341–371.
[40] H.T. Davis, Introduction to Nonlinear Differential and Integral Equations, Dover Publications Inc., New York, 1962.
[41] S. Chandrasekhar, Introduction to Study of Stellar Structure, Dover Publications Inc., New York, 1967.
[42] O.U. Richardson, The Emission of Electricity from Hot Bodies, Longmans Green and Company, London, 1921.
[43] A.M. Wazwaz, Adomian decomposition method for a reliable treatment of the Emden–Fowler equation, Appl. Math. Comput. 161 (2005) 543–560.
[44] E. Babolian, J. Biazar, Solution of nonlinear equations by modified Adomian decomposition method, Appl. Math. Comput. 132 (2002) 167–172.
[45] M. Kumar, N. Singh, Modified Adomian decomposition and computer implementation for solving singular boundary value problems arising in various physical problems, Comput. Chem. Eng. 34 (2010) 1750–1760.
[46] M. Dehghan, F. Shakeri, Approximate solution of a differential equation arising in astrophysics using the variational iteration method, New Astron. 13 (2008) 53–59.
[47] J.H. He, Variational approach to the Lane–Emden equation, Appl. Math. Comput. 143 (2003) 539–541.
[48] A. Yildirim, T. Ozis, Solutions of singular IVPs of Lane–Emden type by the variational iteration method, Nonlinear Anal. 70 (2009) 2480–2484.
[49] M.S.H. Chowdhury, I. Hashim, Solutions of a class of singular second order initial value problems by homotopy-perturbation method, Phys. Lett. A 365 (2007) 439–447.
[50] O.P. Singh, R.K. Pandey, V.K. Singh, Analytical algorithm of Lane–Emden type equation arising in astrophysics using modified homotopy analysis method, Comput. Phys. Commun. 180 (2009) 1116–1124.
[51] M.S.H. Chowdhury, I. Hashim, Solutions of Emden-Fowler Equations by homotopy-perturbation method, Nonlinear Anal.: Real Word Appl. 10 (2009) 104–115.
[52] A.S. Bataineh, M.S.M. Noorani, I. Hashim, Homotopy analysis method for singular IVPs of Emden-Fowler type, Commun. Nonlinear Sci. Numer. Simul. 14 (2009) 1121–1131.