DOI: 10.2478/awutm-2018-0001
DE GRUYTER OPEN

Analele Universităţii de Vest,
Timişoara
Seria Matematică – Informatică
LVI, 1, (2018), 3– 22

A Single Layer Functional Link Artificial Neural Network based on Chebyshev Polynomials for Neural Evaluations of Nonlinear Nth Order Fuzzy Differential Equations

Asmat Ara[1], Oyoon Abdul Razzaq[2], and Najeeb Alam Khan[3]

Abstract. Bearing in mind the considerable importance of fuzzy differential equations (FDEs) in different fields of science and engineering, in this paper, nonlinear *nth* order FDEs are approximated, heuristically. The analysis is carried out on using Chebyshev neural network (ChNN), which is a type of single layer functional link artificial neural network (FLANN). Besides, explication of generalized Hukuhara differentiability (gH-differentiability) is also added for the *nth* order differentiability of fuzzy-valued functions. Moreover, general formulation of the structure of ChNN for the governing problem is described and assessed on some examples of nonlinear FDEs. In addition, comparison analysis of the proposed method with Runge-Kutta method is added and also portrayed the error bars that clarify the feasibility of attained solutions and validity of the method.

AMS Subject Classification (2000). 34A07; 92B20
Keywords. gH-differentiability; fuzzy-valued function; nonlinear fuzzy differential equation; neural network

[1]Department of Computer Science, Muhammad Ali Jinnah University, Karachi, Pakistan, noble.asmat@yahoo.com
[2]Department of Humanities and Social Sciences, Bahria University, Karachi, Pakistan, oyoon.abdulrazzaq@yahoo.com
[3]Department of Mathematics, University of Karachi, Pakistan, njbalam@yahoo.com

1 Introduction

After the initiative of constructing fuzzy differential equations sequentially by Zadeh [1], Dubois et al. [2] and Kaleva [3], FDEs have been investigated by numerous authors to attain its numerical and analytical approximations. In recent times, many ways and means are being established to analyze and simulate fuzzy differential equations. For instance, Euler type methods [4], shooting method [5], fuzzy Picard method [6], fuzzy Laplace transform [7], fuzzy Sumudu transform [8], Runge-Kutta method [9]-[10], fuzzy variational iteration method [11], Adams predictor corrector [12], Taylor method [13], modified Homotopy perturbation method [14], to name a few. Along with these techniques, various papers are found where the latest methods, like different artificial neural networks [15]-[16], are also carried out for the evaluation of FDEs.

Recently, artificial neural networks (ANNs) is being widely used to solve linear and nonlinear ordinary and partial differential equations. It has gained immense attention of researchers for its universal approximating capabilities of initial and boundary value problems [17]-[19]. ANNs is considered to be more advantageous than the other numerical methods as it has lesser number of model parameters and on increasing the sampling points it does not go through any computational complexity. FLANN [20]-[21], multilayer perceptron (MLP) [22] and radial basis function (RBF) networks [23] are three different neural network structures that are mostly found in the literature. Among these, FLANNs has got abundant interest, for the reason that it involves polynomials functions for functional expansion and makes the structure simple.

Nonlinear FDEs are of significant importance, nowadays, as these envelop the nonlinearity and uncertainties of dynamical models, simultaneously. Therefore, successful scrutiny of these equations has augmented a notable contribution in the literature. In this scenario, we endeavor to utilize Chebyshev neural network to efficiently obtain the approximate solutions of nonlinear *nth* order FDEs. It is a single layer orthonormal FLANNs, where Chebyshev polynomials of second kind are chosen as the basis for the construction of activation function. The convergence rate of FLANNs is faster and has lesser computational complexity than MLP method. The rest of the paper is arranged as follows: In Section 2, preliminaries of fuzzy set theory are described briefly along with the explanation of *nth* order differentiability of fuzzy-valued functions under gH-differentiability. Structural elaboration of

ChNN is mentioned in Section 3. Section 4 illustrates general formulation of proposed algorithm for *nth* order FDEs. Graphical results of some nonlinear FDEs in comparison with the Runge-Kutta method, and error bars for each example are investigated in Section 5. In addition, constructive conclusion is drawn in Section 6 on the basis of the interpretations examined in Section 5.

2 Preliminary

This section contains brief description about fuzzy set theory and differentiability of fuzzy-valued functions that are considerably important for the remaining paper. The detailed definitions and properties of fuzzy set theory are greatly found in [1]-[14].

2.1 Fuzzy Set Theory

Λ^f is said to be set of fuzzy numbers ϖ, if $\varpi : \Re \to [0,1]$ is normal, fuzzy convex, upper semi continuous and compactly supported on $\Re$ (real line). Each ϖ can be represented by nonempty compact intervals as, $[\varpi]^\gamma = [\underline{\varpi}(\gamma), \overline{\varpi}(\gamma)]$ that are said to be γ-level sets of ϖ for $\gamma \in [0,1]$. Moreover, $\underline{\varpi}(\gamma)$ and $\overline{\varpi}(\gamma)$ are non-decreasing lower function and non-increasing upper function, respectively, such that both are bounded left continuous on $(0,1]$, right continuous at $\gamma = 0$ and $\underline{\varpi}(\gamma) \leq \overline{\varpi}(\gamma)$. The length of the γ-level set of ϖ is described as $\mathbf{L}[\varpi(\gamma)] = \overline{\varpi}(\gamma) - \underline{\varpi}(\gamma)$.
Let θ and ϑ be two fuzzy numbers, then gH-difference between these two fuzzy numbers is defined as:

$$\theta \ominus_{gH} \vartheta = \varsigma = \begin{cases} (a)\, \theta = \vartheta + \varsigma \\ (b)\, \vartheta = \theta + (-1)\,\varsigma \end{cases}$$

And in terminology of γ-level sets, for $\gamma \in [0,1]$

$$\theta(\gamma) \ominus_{gH} \vartheta(\gamma) = \big[\min\big\{\underline{\theta}(\gamma) - \underline{\vartheta}(\gamma), \overline{\theta}(\gamma) - \overline{\vartheta}(\gamma)\big\}, \max\big\{\underline{\theta}(\gamma) - \underline{\vartheta}(\gamma), \overline{\theta}(\gamma) - \overline{\vartheta}(\gamma)\big\}\big]$$

Additionally, gH-difference can also be illustrated with respect to the length of fuzzy numbers. Let, $\mathbf{L}[\theta(\gamma)]$ and $\mathbf{L}[\vartheta(\gamma)]$ be length of θ and ϑ,

respectively, then

a) $\theta(\gamma) \ominus_{gH} \vartheta(\gamma) = \left[\underline{\theta}(\gamma) - \underline{\vartheta}(\gamma), \overline{\theta}(\gamma) - \overline{\vartheta}(\gamma)\right]$ if $\mathbf{L}[\theta(\gamma)] \geq \mathbf{L}[\vartheta(\gamma)]$

b) $\theta(\gamma) \ominus_{gH} \vartheta(\gamma) = \left[\overline{\theta}(\gamma) - \overline{\vartheta}(\gamma), \underline{\theta}(\gamma) - \underline{\vartheta}(\gamma)\right]$ if $\mathbf{L}[\theta(\gamma)] < \mathbf{L}[\vartheta(\gamma)]$

2.2 Fuzzy-Valued Function

Any function $\widetilde{\mu}(\lambda)$ is said to be a fuzzy-valued function if $\widetilde{\mu} : \Re \to \Lambda^f$, for all $\lambda \in \Re$, where Λ^f is the space of all fuzzy numbers on $\Re$.

2.3 Generalized Hukuhara Differentiability

The gH-differentiability of fuzzy-valued functions was initially introduced by Bede et al. [24]. It has been followed by several authors for fuzzy differential equations of initial and boundary value problems [4]-[7]. Subsequent to the gH-difference, gH-differentiability of fuzzy-valued function is described as:
A continuous fuzzy-valued function $\widetilde{\chi} : (a, b) \to \Lambda^f$ is said to be
gH-differentiable at $\lambda_0 \in (a, b)$, if $\widetilde{\chi}'_{gH}(\lambda_0) \in \Lambda^f$ exists, such that

$$\widetilde{\chi}'_{gH}(\lambda_0) = \lim_{h \to 0} \frac{\widetilde{\chi}(\lambda_0 + h) \ominus_{gH} \widetilde{\chi}(\lambda_0)}{h} \tag{2.1}$$

where h is such that $(\lambda_0 + h) \in (a, b)$. For the γ-level sets of $\widetilde{\chi}(\lambda)$ i.e. $\chi(\lambda; \gamma) = \left[\underline{\chi}(\lambda; \gamma), \overline{\chi}(\lambda; \gamma)\right]$, it is said to be
gH-differentiable at λ, if $\underline{\chi}(\lambda; \gamma)$ and $\overline{\chi}(\lambda; \gamma)$ are differentiable i.e.

$$\widetilde{\chi}'_{gH}(\lambda; \gamma) = \left[\min\left\{\frac{d}{d\lambda}\underline{\chi}(\lambda; \gamma), \frac{d}{d\lambda}\overline{\chi}(\lambda; \gamma)\right\}, \max\left\{\frac{d}{d\lambda}\underline{\chi}(\lambda; \gamma), \frac{d}{d\lambda}\overline{\chi}(\lambda; \gamma)\right\}\right] \tag{2.2}$$

Concisely, $\widetilde{\chi}(\lambda)$ is said to be $gH^{(i)}$-differentiable at λ i.e.

$$\chi'_{gH}(\lambda; \gamma) = \left[\frac{d}{d\lambda}\underline{\chi}(\lambda; \gamma), \frac{d}{d\lambda}\overline{\chi}(\lambda; \gamma)\right]$$

if $\mathbf{L}\left[\chi\left(\lambda;\gamma\right)\right]$ is increasing in (a,b) for $\gamma\in[0,1]$ and $gH^{(ii)}$-differentiable at λ and

$$\chi'_{gH}\left(\lambda;\gamma\right)=\left[\frac{d}{d\lambda}\overline{\chi}\left(\lambda;\gamma\right),\frac{d}{d\lambda}\underline{\chi}\left(\lambda;\gamma\right)\right]$$

if $\mathbf{L}\left[\chi\left(\lambda;\gamma\right)\right]$ is decreasing in (a,b) for $\gamma\in[0,1]$. Analogously, extending the illustration of gH-differentiability for *nth* order derivative of $\widetilde{\chi}\left(\lambda\right)$, we have,
A continuous fuzzy-valued function $\chi:(a,b)\times\Lambda^{f}\times\cdots\times\Lambda^{f}\rightarrow\Lambda^{f}$, with $\widetilde{\chi}'_{gH}\left(\lambda\right),\cdots,\widetilde{\chi}_{gH}^{(n-1)}\left(\lambda\right)\in\Lambda^{f}$ also be continuous functions, is said to be *nth* order gH-differentiable atλ_0, if $\widetilde{\chi}_{gH}^{(n)}\left(\lambda_0\right)\in\Lambda^{f}$, $\forall n\in\aleph$ (natural numbers),

$$\widetilde{\chi}_{gH}^{(n)}\left(\lambda_0\right)=\lim_{h\rightarrow0}\frac{\widetilde{\chi}_{gH}^{(n-1)}\left(\lambda_0+h\right)\ominus_{gH}\widetilde{\chi}_{gH}^{(n-1)}\left(\lambda_0\right)}{h}\tag{2.3}$$

Sequentially, if $\widetilde{\chi}\left(\lambda\right),\widetilde{\chi}'_{gH}\left(\lambda\right),\ldots,\widetilde{\chi}_{gH}^{(n-1)}\left(\lambda\right)$ are $gH^{(i)}$-differentiable, i.e. $\mathbf{L}\left[\chi\left(\lambda;\gamma\right)\right],\mathbf{L}\left[\chi'_{gH}\left(\lambda;\gamma\right)\right],\ldots,\mathbf{L}\left[\chi_{gH}^{(n-1)}\left(\lambda;\gamma\right)\right]$ are increasing in (a,b) for $\gamma\in[0,1]$, or, if $\widetilde{\chi}\left(\lambda\right),\widetilde{\chi}'_{gH}\left(\lambda\right),\ldots,\widetilde{\chi}_{gH}^{(n-1)}\left(\lambda\right)$ are $gH^{(ii)}$-differentiable, i.e. $\mathbf{L}\left[\chi\left(\lambda;\gamma\right)\right],\mathbf{L}\left[\chi'_{gH}\left(\lambda;\gamma\right)\right],\ldots,\mathbf{L}\left[\chi_{gH}^{(n-1)}\left(\lambda;\gamma\right)\right]$ are decreasing in (a,b) for $\gamma\in[0,1]$, then

$$\chi_{gH}^{(n)}\left(\lambda;\gamma\right)=\left[\frac{d^{(n-1)}}{d\lambda^{(n-1)}}\underline{\chi}\left(\lambda;\gamma\right),\frac{d^{(n-1)}}{d\lambda^{(n-1)}}\overline{\chi}\left(\lambda;\gamma\right)\right]\tag{2.4}$$

Additionally, if $\widetilde{\chi}\left(\lambda\right)$ is $gH^{(i)}$-differentiable and $\widetilde{\chi}'_{gH}\left(\lambda\right),\ldots,\widetilde{\chi}_{gH}^{(n-1)}\left(\lambda\right)$ are $gH^{(ii)}$-differentiable, i.e. $\mathbf{L}\left[\chi\left(\lambda;\gamma\right)\right]$ is increasing and $\mathbf{L}\left[\chi'_{gH}\left(\lambda;\gamma\right)\right]$, …, $\mathbf{L}\left[\chi_{gH}^{(n-1)}\left(\lambda;\gamma\right)\right]$ are decreasing in (a,b) for $\gamma\in[0,1]$, or, if $\widetilde{\chi}\left(\lambda\right)$ is $gH^{(ii)}$-differentiable and $\widetilde{\chi}'_{gH}\left(\lambda\right),\ldots,\widetilde{\chi}_{gH}^{(n-1)}\left(\lambda\right)$ are $gH^{(i)}$-differentiable, i.e. $\mathbf{L}\left[\chi\left(\lambda;\gamma\right)\right]$ is decreasing and $\mathbf{L}\left[\chi'_{gH}\left(\lambda;\gamma\right)\right],\ldots,\mathbf{L}\left[\chi_{gH}^{(n-1)}\left(\lambda;\gamma\right)\right]$ are increasing in (a,b)

for $\gamma \in [0,1]$, then [24],

$$\chi_{gH}^{(n)}(\lambda;\gamma) = \left[\frac{d^{(n-1)}}{d\lambda^{(n-1)}}\overline{\chi}(\lambda;\gamma), \frac{d^{(n-1)}}{d\lambda^{(n-1)}}\underline{\chi}(\lambda;\gamma)\right] \tag{2.5}$$

3 Chebyshev Neural Network

The original goal of ANNs approach is that it solves the problems in the same way as a human brain, for instance passing information in the reverse direction and adjusting the network to reproduce that information. It is greatly exercised in several areas of science and engineering such as system identification, medical diagnosis, ocean modelling, geomorphology, etc. In this section, we define the basic structural algorithm of proposed Chebyshev neural network, which is a type of ANNs that has a single hidden layer [21].

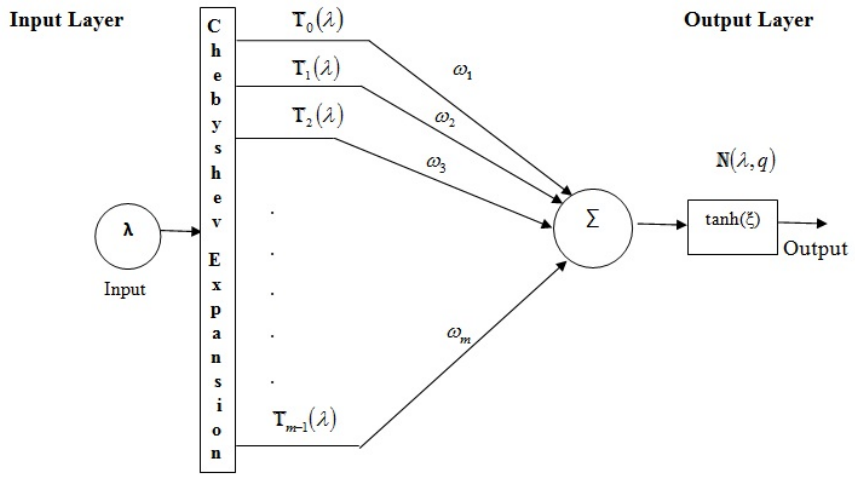

Figure 1: Schematic structure of Chebyshev neural network

3.1 Structure of Chebyshev Neural Network

Chebyshev neural network is a functional link artificial neural network, where the Chebyshev polynomials of second kind are taken into account as basis functions for the neural networks. The structural design of ChNN employed for the present problem is shown in Fig.1 that depicts the network connections from a single input node to a functional expansion block and then to a single output node. The neural model is configured with two parts, numerical transformation and training process. In numerical transformation part, each input data $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_h)^T$ is expanded to several terms using Chebyshev polynomials of second kind $\mathbf{T}(\lambda)$. Expansion of functions with truncated series of Chebyshev polynomials is highly encouraged as its expansions converge more rapidly than the other orthogonal polynomials. The Chebyshev polynomials of second kind may be obtained by the following recursive formula,

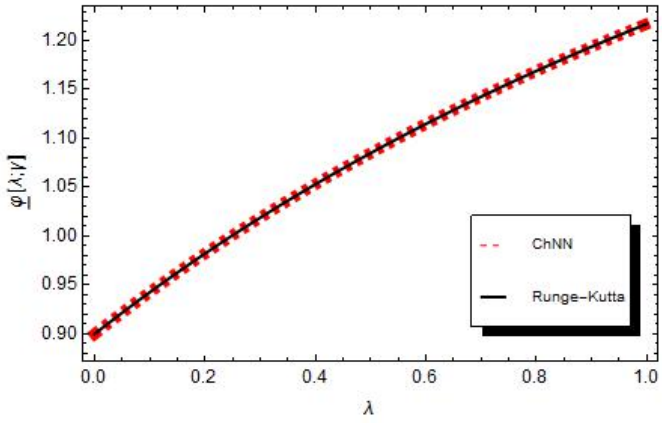

Figure 2: Comparison of lower approximate solution obtained by ChNN versus Runge-Kutta method of Problem 5.1 for $\gamma = 0.6$

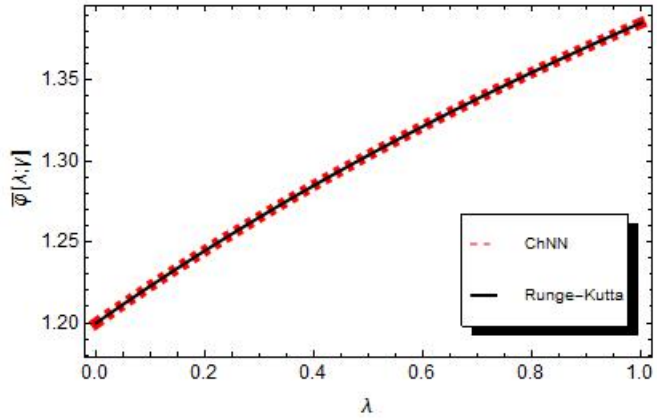

Figure 3: Comparison of upper approximate solution obtained by ChNN versus Runge-Kutta method of Problem 5.1 for $\gamma = 0.6$

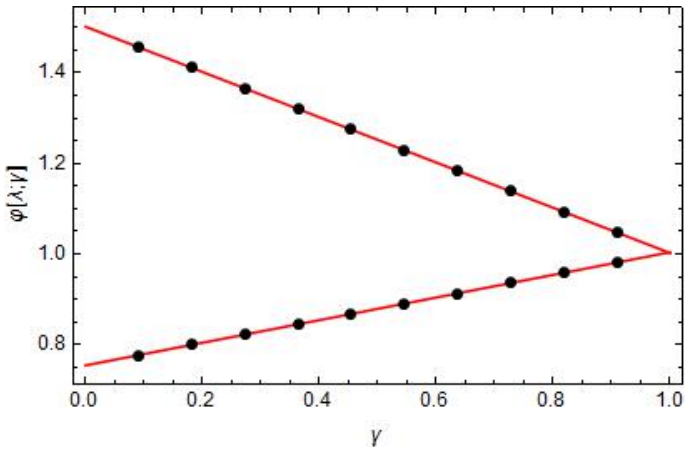

Figure 4: Fuzzy solution of Problem 5.1 obtained by ChNN for $\lambda = 0.009$

$$\mathbf{T}_{p+1}(\lambda) = 2\lambda \mathbf{T}_p(\lambda) - \mathbf{T}_{p-1}(\lambda) \tag{3.1}$$

where $\mathbf{T}_0(\lambda) = 1$, $\mathbf{T}_1(\lambda) = 2\lambda$ and $\mathbf{T}_p(\lambda)$ denotes *pth* Chebyshev polynomial of second kind. In ChNN to get the result the dimension of the input is increased through Chebyshev polynomial.

In the training process, network parameters are updated and an error func-

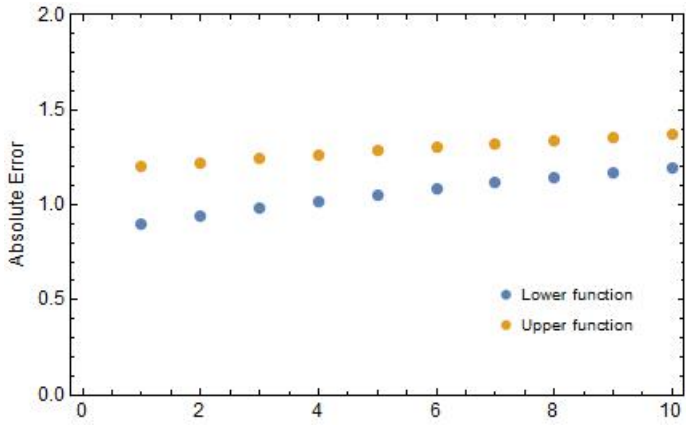

Figure 5: Error bar plots of obtained solutions with error variations of Problem 5.1

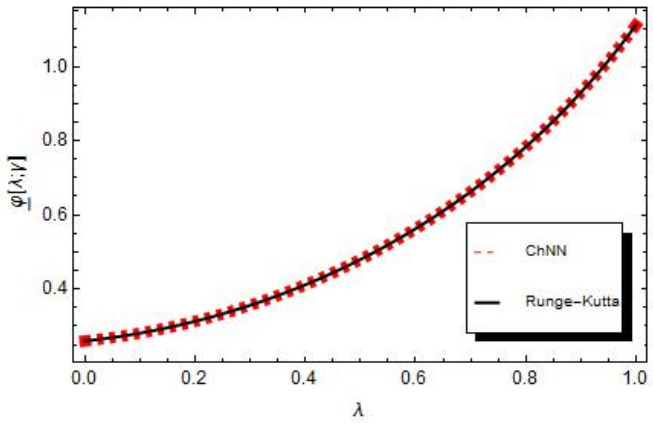

Figure 6: Comparison of lower approximate solution obtained by ChNN versus Runge-Kutta method of Problem 5.2 for $\gamma = 0.6$

tion is minimized using different algorithms. Here, error back propagation algorithm is exercised to reduce the error and update weights, until the network learns the training data. In this process gradient of an error function with respect to the network parameters q is calculated. The hyperbolic tangent function $\tanh(\lambda)$ is considered for the activation of the output function, i.e.

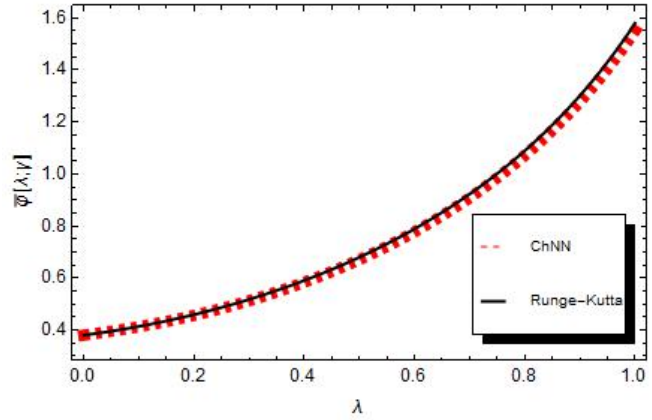

Figure 7: Comparison of upper approximate solution obtained by ChNN versus Runge-Kutta method of Problem 5.2 for $\gamma = 0.6$

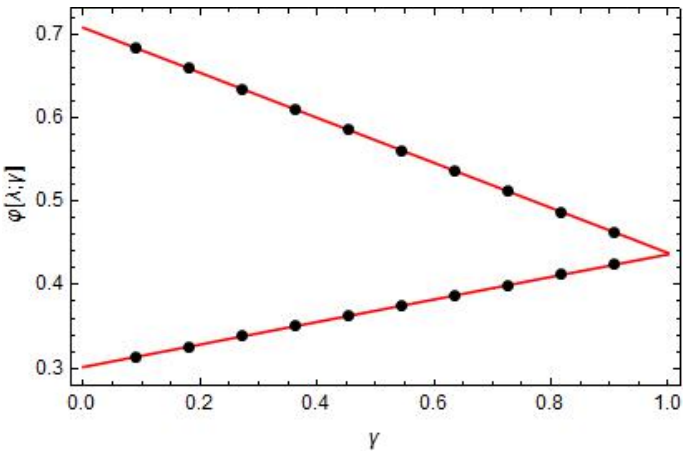

Figure 8: Fuzzy solution of Problem 5.2 obtained by ChNN for $\lambda = 0.009$

$$\mathbf{N}(\lambda, q) = \tanh(\xi) \tag{3.2}$$

with input data $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_h)^T$, parameters (weights) q and ξ be the linear weighted sum of Chebyshev polynomials that can be expressed as,

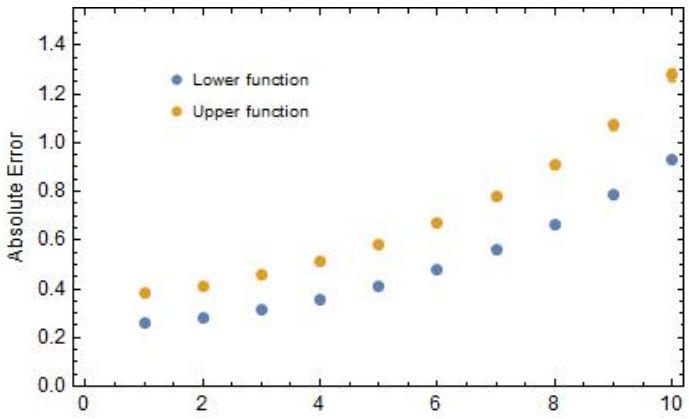

Figure 9: Error bar plots of obtained solutions with error variations of Problem 5.2

$$\xi = \sum_{j=1}^{m} \omega_j \mathbf{T}_{j-1}(\lambda) \tag{3.3}$$

where $\mathbf{T}_{j-1}(\lambda)$ denote the $(j-1)\,th$ Chebyshev polynomial and ω_j are the weight vectors of the ChNN. Now, for modification of weights of ChNN the principle of back propagation is given as

$$\omega_j^{k+1} = \omega_j^k + \Delta\omega_j^k = \omega_j^k + \left(-\tau \frac{\partial \mathbf{E}(\lambda, q)}{\partial \omega_j^k}\right) \tag{3.4}$$

where $\mathbf{E}(\lambda, q)$ is the error function that is to be minimized, τ is learning parameter that can have any value in $[10^{-1}, 10^{-3}]$ and k is iteration step.

4 Formulation of ChNN for Nonlinear Nth Order Fuzzy Differential Equation

In this section, we formulate the trial and error function of ChNN method to approximate nonlinear *nth* order FDEs. Analytical and numerical investigations of *nth* order FDEs are accomplished by various authors, for instance, Jayakumar et al. [10] used Runge-Kutta method of order five to numerically

solve *nth* order FDEs, Salahshour [25] proposed an integral form to examine analytical solutions of these equations, Ahmady [26] proposed piecewise approximation method to discuss the solutions of *nth* order FDEs etc. Comparatively, ChNN for its less computational complexity, executes the appropriate analytical approximations more rapidly. In a nutshell, the following attributes of ChNN has made it more efficient approximator than the other existing methods:

- As learning proceeds, the weights vary and controls the strength of the signals between neurons that it sends to the activation function.
- The passing process of weighted sum through a nonlinear activation function proficiently bounds the values of the neurons so that the neural network is not paralyzed by divergent neurons.
- Using gradient descent method, for training algorithm, changes the weights of the network in such a manner that the error of lower and upper functions is minimized, separately.
- Its process of expanding the functions in truncated series of Chebyshev polynomials and insertions of neurons, completely discretizes each lower and upper functions of FDE to an algebraic equation.

As a result, the intricacy of integrations, which exist in other methods in case of nonlinear functions [14], become extinct. Let the considered nonlinear FDE be of the form

$$\widetilde{\varphi}_{gH}^{(n)}(\lambda) = f\left(\lambda, \widetilde{\varphi}(\lambda), \widetilde{\varphi}'_{gH}(\lambda), \ldots, \widetilde{\varphi}_{gH}^{(n-1)}(\lambda)\right) + g(\lambda) \tag{4.1}$$

with the initial conditions $\widetilde{\varphi}(\lambda_0) = u_0$, $\widetilde{\varphi}'_{gH}(\lambda_0) = u_1$, ..., $\widetilde{\varphi}_{gH}^{(n-1)}(\lambda_0) = u_{n-1}$, where $\widetilde{\varphi}(\lambda)$ is a fuzzy-valued function,
$f\left(\lambda, \widetilde{\varphi}(\lambda), \widetilde{\varphi}'_{gH}(\lambda), \ldots, \widetilde{\varphi}_{gH}^{(n-1)}(\lambda)\right)$ describes the nonlinear function of λ and $\widetilde{\varphi}(\lambda), \widetilde{\varphi}'_{gH}(\lambda), \ldots, \widetilde{\varphi}_{gH}^{(n-1)}(\lambda)$, $g(\lambda)$is the nonhomogeneous part of the equation and u_0, u_1,..., u_{n-1} represent fuzzy numbers.
Now, let $\widetilde{\varphi}_t(\lambda, q)$ be the trial solution with adjustable parameters q, then Eq. (4.1) changes into

$$\widetilde{\varphi}_{tgH}^{(n)}(\lambda, q) = f\left(\lambda, \widetilde{\varphi}_t(\lambda, q), \widetilde{\varphi}'_{tgH}(\lambda, q), \ldots, \widetilde{\varphi}_{tgH}^{(n-1)}(\lambda, q)\right) + g(\lambda) \tag{4.2}$$

In ChNN, $\widetilde{\varphi}_t(\lambda, q)$, for the case of Eq. (4.1) is expressed as:

$$\widetilde{\varphi}_t(\lambda, q) = \widetilde{\varphi}(\lambda_0) + (\lambda - \lambda_0)\,\widetilde{\varphi}'(\lambda_0) + (\lambda - \lambda_0)^2\,\widetilde{\varphi}''(\lambda_0) + \cdots + (\lambda - \lambda_0)^n\,\mathbf{N}(\lambda, q) \tag{4.3}$$

where the last term $\mathbf{N}(\lambda, q)$ is defined in Eq. (3.2) as the output of ChNN, which contains the parameters that are to be modified, whereas all the other remaining terms satisfy the initial conditions and do not contain adjustable parameters. The γ-level set of trial solution $\varphi_t(\lambda, q; \gamma) = \left[\underline{\varphi}_t(\lambda, q; \gamma), \overline{\varphi}_t(\lambda, q; \gamma)\right]$ can be written as, for all $\gamma \in [0, 1]$,

$$\underline{\varphi}_t\left(\lambda, \underline{q}; \gamma\right) = \underline{\varphi}(\lambda_0; \gamma) + (\lambda - \lambda_0)\,\underline{\varphi}'(\lambda_0; \gamma) + (\lambda - \lambda_0)^2\,\underline{\varphi}''(\lambda_0; \gamma) + \cdots + (\lambda - \lambda_0)^n\,\mathbf{N}\left(\lambda, \underline{q}\right) \tag{4.4}$$

$$\overline{\varphi}_t(\lambda, \overline{q}; \gamma) = \overline{\varphi}(\lambda_0; \gamma) + (\lambda - \lambda_0)\,\overline{\varphi}'(\lambda_0; \gamma) + (\lambda - \lambda_0)^2\,\overline{\varphi}''(\lambda_0; \gamma) + \cdots + (\lambda - \lambda_0)^n\,\mathbf{N}(\lambda, \overline{q}) \tag{4.5}$$

$\underline{q}$ and $\overline{q}$ are assigned merely to make a distinction between the parameters of lower and upper functions, respectively, during the network training for lower and upper functions, independently. Next, define the error function for the considered FDE with input data $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_h)^T$ and parameters q as:

$$\widetilde{\mathbf{E}}(\lambda, q) = \frac{1}{2}\sum_{i=1}^{h}\left(\widetilde{\varphi}^{(n)}_{tgH}(\lambda_i, q) - f\left(\lambda_i, \widetilde{\varphi}_t(\lambda_i, q), \widetilde{\varphi}'_{tgH}(\lambda_i, q), \ldots, \widetilde{\varphi}^{(n-1)}_{tgH}(\lambda_i, q)\right) - g(\lambda_i)\right)^2 \tag{4.6}$$

for $i = 1, 2, 3, \ldots$ and $\forall \gamma \in [0, 1]$,

$$\underline{\mathbf{E}}\left(\lambda, \underline{q}; \gamma\right) = \frac{1}{2}\sum_{i=1}^{h}\left(\underline{\varphi}^{(n)}_{tgH}\left(\lambda_i, \underline{q}; \gamma\right) - f\left(\lambda_i, \underline{\varphi}_t\left(\lambda_i, \underline{q}; \gamma\right), \underline{\varphi}'_{tgH}\left(\lambda_i, \underline{q}; \gamma\right), \ldots, \underline{\varphi}^{(n-1)}_{tgH}\left(\lambda_i, \underline{q}; \gamma\right)\right) - g(\lambda_i)\right)^2 \tag{4.7}$$

$$\overline{\mathbf{E}}\left(\lambda,\overline{q};\gamma\right)=\frac{1}{2}\sum_{i=1}^{h}\Big(\overline{\varphi}_{tgH}^{(n)}\left(\lambda_i,\overline{q};\gamma\right)-f\left(\lambda_i,\overline{\varphi}_t\left(\lambda_i,\overline{q};\gamma\right),\overline{\varphi}'_{tgH}\left(\lambda_i,\overline{q};\gamma\right),\ldots,\overline{\varphi}_{tgH}^{(n-1)}\left(\lambda_i,\overline{q};\gamma\right)\Big)-g\left(\lambda_i\right)\Big)^2 \quad (4.8)$$

To update the weights, derivatives of $\underline{\mathbf{E}}\left(\lambda,\underline{q};\gamma\right)$ and $\overline{\mathbf{E}}\left(\lambda,\overline{q};\gamma\right)$ are taken with respective to the weights $\underline{\omega}_j$ and $\overline{\omega}_j$, accordingly, i.e.

$$\frac{\partial\underline{\mathbf{E}}\left(\lambda,\underline{q};\gamma\right)}{\partial\underline{\omega}_j}=\frac{\partial}{\partial\underline{\omega}_j}\left(\frac{1}{2}\sum_{i=1}^{h}\Big(\underline{\varphi}_{tgH}^{(n)}\left(\lambda_i,\underline{q};\gamma\right)-f\Big(\lambda_i,\underline{\varphi}_t\left(\lambda_i,\underline{q};\gamma\right),\underline{\varphi}'_{tgH}\left(\lambda_i,\underline{q};\gamma\right),\ldots,\underline{\varphi}_{tgH}^{(n-1)}\left(\lambda_i,\underline{q};\gamma\right)\Big)-g\left(\lambda_i\right)\Big)^2\right) \quad (4.9)$$

and

$$\frac{\partial\overline{\mathbf{E}}\left(\lambda,\overline{q};\gamma\right)}{\partial\overline{\omega}_j}=\frac{\partial}{\partial\overline{\omega}_j}\left(\frac{1}{2}\sum_{i=1}^{h}\Big(\overline{\varphi}_{tgH}^{(n)}\left(\lambda_i,\overline{q};\gamma\right)-f\left(\lambda_i,\overline{\varphi}_t\left(\lambda_i,\overline{q};\gamma\right),\overline{\varphi}'_{tgH}\left(\lambda_i,\overline{q};\gamma\right),\ldots,\overline{\varphi}_{tgH}^{(n-1)}\left(\lambda_i,\overline{q};\gamma\right)\Big)-g\left(\lambda_i\right)\Big)^2\right) \quad (4.10)$$

Using Eqs. (4.9) and (4.10) in Eq. (3.4) and working out iteratively, the weights $\underline{\omega}_j$ and $\overline{\omega}_j$ are modified until the network learns the data and minimum error is obtained for lower and upper functions together. Throughout this process, different values of $\tau\in[10^{-1},10^{-3}]$ are taken for each iteration. The modified weights are then plugged into trial solutions i.e. in Eqs. (4.4) and (4.5), to obtain the approximate solutions of lower and upper functions, accordingly of fuzzy-valued function $\widetilde{\varphi}\left(\lambda\right)$.

5 Illustrative Examples

In this section, we have obtained the solutions of some nonlinear FDEs for eleven equidistant input points (neurons) in $[0,1]$ and considered eleven

weights together with ten Chebyshev polynomials of second kind for the activation of ξ. Consequently, results are plotted in comparison with the analytical solutions and Runge-Kutta method [10]. Exact solutions of each governing problem are obtained using *Mathematica*. Moreover, the accuracy of the algorithm is elaborated through error bar plots, which is displayed by plotting the deviations of exact and approximated values, against the approximated values.
Since, the computational procedure is same for all the cases of gH-differentiability, for brief exemplification here, we have just considered $\mathbf{L}\left[\varphi\left(\lambda;\gamma\right)\right]$, $\mathbf{L}\left[\varphi'_{gH}\left(\lambda;\gamma\right)\right]$, ..., $\mathbf{L}\left[\varphi^{(n-1)}_{gH}\left(\lambda;\gamma\right)\right]$ to be increasing. Therefore, only the case of $gH^{(i)}$-differentiability is deliberated in the following problems.

Problem 5.1. Consider Eq. (4.1) for $n=1$, $f\left(\lambda,\widetilde{\varphi}\left(\lambda\right)\right)=\exp\left(-\widetilde{\varphi}\left(\lambda\right)^2\right)$ and $g\left(\lambda\right)=0$ with initial conditions $\varphi\left(0;\gamma\right)=\left(0.75+0.25\gamma,1.5-0.5\gamma\right)$, for all $\gamma\in[0,1]$. The attained equation is found in [4] where its numerical solutions are tabulated and graphically represented on operating Euler type methods. Here, we procure its analytical solutions on employing ChNN method. Following the algorithm, enlightened in previous section, after few iterations we attained the weights as,

$$\begin{aligned}\underline{q}=\{&5.052\times10^{-1},-1.209\times10^{-1},2.735\times10^{-2},-4.702\times10^{-3},\\&2.587\times10^{-4},2.133\times10^{-4},-1.112\times10^{-4},3.311\times10^{-5},\\&-6.771\times10^{-6},9.058\times10^{-7},-6.129\times10^{-8}\}\end{aligned}$$

for lower function and

$$\begin{aligned}\overline{q}=\{&2.477\times10^{-1},-3.773\times10^{-2},6.369\times10^{-3},-1.059\times10^{-3},\\&1.645\times10^{-4},2.264\times10^{-5},2.513\times10^{-6},-1.717\times10^{-7},\\&-4.900\times10^{-9},2.881\times10^{-9},-2.661\times10^{-10}\}\end{aligned}$$

for the upper function. On substituting the attained values of $\underline{q}$, $\overline{q}$, $\lambda_0=0$ and the initial condition $\varphi\left(0;\gamma\right)$ in Eqs. (4.4) and (4.5) the analytical solution of the fuzzy function $\widetilde{\varphi}\left(\lambda\right)$ is acquired. The solutions $\underline{\varphi}\left(\lambda;\gamma\right)$ and $\overline{\varphi}\left(\lambda;\gamma\right)$ are depicted by Figs. 2 and 3, respectively, in comparison with the Runge-Kutta method. Fig. 4 shows the fuzzy solution for $\lambda=0.009$ and different values of $\gamma\in[0,1]$. Fig. 5 plots pointwise error variations between exact solutions and approximated values of lower and upper functions, where the negligible variation clarify the solutions to be in good agreement with the

exact ones.

Problem 5.2. Next, take Eq. (4.1) for $n = 2$, $f\left(\lambda, \widetilde{\varphi}(\lambda), \widetilde{\varphi}'_{gH}(\lambda)\right) = 2\widetilde{\varphi}(\lambda) \bullet \left(\left(\widetilde{\varphi}(\lambda)\right)^2 + \lambda\right)$ and $g(\lambda) = 1$ with initial conditions $\varphi(0;\gamma) = (0.2 + 0.1\gamma, 0.5 - 0.2\gamma)$ and $\varphi'_{gH}(0;\gamma) = (0.1 + 0.1\gamma, 0.4 - 0.2\gamma)$, for all $\gamma \in [0,1]$. The equation so obtained represents the Painlevé II differential equation with fuzzy initial conditions. It is the second nonlinear second order irreducible differential equation among the six Painlevé differential equations that are also known as Painlevé transcendents. Painlevé differential equations are significantly exercised to illustrate different physical procedures [27]. In this endeavor, after employing few iterations of ChNN algorithm, we attained the weights as,

$$\underline{q} = \{5.381, -1.885, 8.701, -7.353, 5.072, -2.853, 1.305, \\ -0.471, 0.128, -2.369, 2.331 \times 10^{-3}\}$$

for lower function and

$$\overline{q} = \{1801.38, -2980.93, 3249.83, -2747.15, 1880.52, \\ -1051.68, 476.06, -169.861, 45.379, -8.180, 0.762\}$$

for upper function. On substituting the above values of $\underline{q}$, $\overline{q}$, $\lambda_0 = 0$ and the initial conditions $\varphi(0;\gamma)$ and $\varphi'_{gH}(0;\gamma)$ in Eqs. (4.4) and (4.5) the analytical solution of the fuzzy function $\widetilde{\varphi}(\lambda)$ is attained. In Figs. 6 and 7 pictorial solutions of lower and upper functions, $\underline{\varphi}(\lambda;\gamma)$ and $\overline{\varphi}(\lambda;\gamma)$ are shown in comparison with the Runge-Kutta method, correspondingly. Additionally, Fig. 8 represents fuzzy solution for $\lambda = 0.35$ and different values of $\gamma \in [0,1]$. In Fig. 9, where each point on the graph represents the calculated solution together with the error variation bars, illustrates the accuracy of proposed algorithm, because the error bars are barely visible.

6 Conclusions

In this work, we studied Chebyshev neural networks method for nonlinear *nth* order fuzzy differential equations. Firstly, the *nth* order differentiability of fuzzy-valued functions under the theory of gH-differentiability was elaborated. Secondly, structural algorithm of ChNN was explained generally and

later for the proposed FDEs. At last, for numerical illustration, we considered two examples of nonlinear FDEs. Thus, following facts are achieved conclusively from the whole study,

- Dealing together with the nonlinearity and impreciseness of the functions, the nonlinear FDEs are widely utilized for modelling different aspects of biology and physics. The immediate, appropriate solutions of these equations augment a great contribution in probing the dynamical effects of existing parameters imprecisely. Hence, this endeavor will provide a new efficient methodology to obtain the effective series solutions of these models.

- Error minimization process in the Chebyshev neural network method benefits to attain feasible solutions with more accuracy, which is observed from the error bar graphs of discussed examples.

- ChNN discretizes all the functions and differential terms due to which the differential equation converts into algebraic equations and hence removes all the difficulties of integrating the terms, mainly in case of nonlinearity.

- Inclusion of orthogonal polynomials as basis functions reduces the hidden layers, which as a result produces the output more rapidly as compared to other multiple layer neural networks and thus lessens the computational time.

- The computational complexity of ChNN can occur in the gradient process of the error function, for the large number of neurons in addition with the nonlinearity of the activation function. However, this step is eased by using *Mathematica*.

- As compared to other methods, such as Euler type methods, Runge-Kutta method, where in each iteration the increment parameter is not modified according to minimization of the error, ChNN, modifies the unknown weights through the error minimization process that highly rectifies the solutions.

- In Runge-Kutta method numerical results are produced, hence for each set of input, the whole program is constructed again, whereas ChNN produces series solution, which is once constructed using modified weights can be utilized for any specified domain of the problem under consideration.

- After the whole manipulation, in Runge-Kutta method, the convergence through the absolute error is calculated to check the accuracy, which consume more time to run the whole program again, in the case of diverging results, whereas the activation function of ChNN controls the divergence of the parameters initially during the learning process.

- Approximated solutions so obtained using ChNN can then be used to measure the solution of FDE at any arbitrary point within the interval defined for the training points. For this reason, without any ambiguity fuzzy solutions were attained for different values of λ, efficiently.

- The architecture of ChNN once established can be employed on FDEs of any order $n > 2$, by simply increasing the order of expansion of the trial solution for higher order.

- The undergone nonlinear examples have wide physical applications, for instance quantum theory, mechanics, nonlinear waves, etc., thus, the involvement of fuzzy theory overcomes the lack of precision of the parameters and variables. In addition, view of the fact that the ChNN models are good universal approximators to nonlinear functions, in this attempt, solving nonlinear *nth* order fuzzy differential equations have developed a new pace for the numerical investigations of fuzzy differential equations.

References

[1] **L.A Zadeh**, Fuzzy Sets, *Information and Control*, **8**, (1965), 338–353

[2] **D.Dubois and H. Prade**, Towards fuzzy differential calculus part 3, *Fuzzy Sets and Systems*, **8**, (1982), 225–233

[3] **O. Kaleva**, Fuzzy differential equations, *Fuzzy Sets and Systems*, **24**, (1987), 301–317

[4] **N.A. Khan O.A. Razzaq and F. Riaz**, Numerical simulations for solving fuzzy fractional differential equations by max-min improved Euler methods, *Journal of Applied Computer Science Methods*, **7**, (2015), 53–83

[5] **L. Jamshidi and L. Avazpour**, Solution of the fuzzy boundary value differential equations under generalized differentiability by shooting method, *Journal of Fuzzy Set Valued Analysis*, (2012)

[6] **Sh.S. Behzadi B. Vahdani T. Allahviranloo and S. Abbasbandy**, Application of fuzzy Picard Method for solving fuzzy quadratic Riccati and fuzzy Painlevé I equations, *Applied Mathematical Modelling*, **17/18**, (2016), 8125–8137

[7] **S. Salahshour and T. Allahviranloo**, Applications of fuzzy Laplace transforms, *Soft Computing*, **17**, (2013), 145–158

[8] **N.A. Khan O.A. Razzaq and M. Ayyaz**, On the solution of fuzzy differential equations by Fuzzy Sumudu Transform, *Nonlinear Engineering*, **4**, (2015), 49–60

[9] **A. Ahmadian S. Salahshour C.S. Chan and D. Baleanu**, Numerical solutions of fuzzy differential equations by an efficient Runge-Kutta method with generalized differentiability, *Fuzzy Sets and Systems*, (2016)

[10] **T. Jayakumar and S. Indrakumar K. Kanagarajan**, Numerical solution of Nth-order fuzzy differential equation by Runge-Kutta method of order five, *International Journal of Mathematical Analysis*, **6**, (2012), 2885–2896

[11] **T. Allahviranloo S. Abbasbandy and Sh.S. Behzadi**, Solving nonlinear fuzzy differential equations by using fuzzy variational iteration method, *Soft Computing*, **18**, (2014), 2191–2200

[12] **D. Shang and X. Guo**, Adams predictor-corrector systems for solving fuzzy differential equations, *Mathematical Problems in Engineering*, **2013**, (2013)

[13] **S. Narayanamoorthy S.P. Sathiyapriya and M. Santhiya**, Solving hybrid fuzzy differential equations using Taylor series method, *International Journal of Pure and Applied Mathematics*, **106**, (2016), 1–10

[14] **N.A. Khan F. Riaz and O.A. Razzaq**, A comparison between numerical methods for solving fuzzy fractional differential equations, *Nonlinear Engineering*, **3**, (2014), 155–162

[15] **N.A. Khan A. Shaikh M.A.Z. Raja and S. Khan**, A neural computational intelligence method based on Legendre polynomials for fuzzy fractional order differential equation, *Journal of Applied Mathematics, Statistics and Informatics*, **12**, (2016), 67–82

[16] **M. Mosleh**, Fuzzy neural network for solving a system of fuzzy differential equations, *Applied Soft Computing*, **13**, (2013), 3597–3607

[17] **S. Mall and S. Chakraverty**, Comparison of artificial neural network architecture in solving ordinary differential equations, *Advances in Artificial Neural Systems*, **2013**, (2013)

[18] **M.A.Z. Raja J.A. Khan A. Zameer**, Numerical treatment of nonlinear singular Flierl-Petviashivili systems using neural networks models, *Neural Computing and Applications*, (2017), 1–24

[19] **K.J. Sabouri S. Effati and M. Pakdaman**, A neural network approach for solving a class of solving fractional optimal control problems, *Neural Processing Letters*, **4**, (2017), 59–74

[20] **H. Zhao and J. Zhang**, Functional link neural network cascaded with Chebyshev orthogonal polynomial for nonlinear channel equalization, *Neural Processing Letters*, **88**, (2008), 1946–1957

[21] **S. Mall and S. Chakraverty**, Chebyshev neural network based model for solving Lane-Emden type equations, *Applied Mathematics and Computation*, **247**, (2014), 100–114

[22] **Y. Shirvany M. Hayati and R. Moradian**, Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations, *Applied Soft Computing*, **9**, (2009), 20–29

[23] **L. Jianyu L. Siwei Q. Yingjian**, Numerical solution of elliptic partial differential equation using radial basis function neural networks, *Neural Networks*, **16**, (2003), 729–734

[24] **B. Bede L. Stefanini**, Generalized differentiability of fuzzy-valued functions, *Fuzzy Sets and Systems*, **230**, (2013), 119–141

[25] **S. Salahshour**, Nth-order fuzzy differential equations under generalized differentiability, *Journal of Fuzzy Set Valued Analysis*, (2011)

[26] **E. Ahmady**, Nth-order fuzzy differential equations under generalized differentiability, *Journal of Fuzzy Set Valued Analysis*, **2015**, (2015), 146–153

[27] **N.A. Khan M. Jamil and N.A. Khan**, Approximations of the nonlinear Painlevé transcendent, *Communications in Numerical Analysis*, **2013**, (2013)

Received: 14.03.2016

Accepted: 4.11.2017

Revised: 8.10.2017