NUMERICAL SOLUTION OF FUZZY DIFFERENTIAL EQUATIONS
BY USING MODIFIED FUZZY NEURAL NETWORK

EMAN A. HUSSIAN*
Al-Mustansiriyah University, Baghdad, Iraq.

MAZIN H. SUHHEM
Asst. Lec., Dep. of statistics,
College of Adm. and Econ. University of sumar, Baghdad, Iraq.

(Received On: 31-05-15; Revised & Accepted On: 22-06-15)

ABSTRACT
In this paper, we introduce a hybrid approach based on modified fuzzy neural network and optimization technique to solve fuzzy differential equations. Using modified fuzzy neural network makes that training points should be selected over an open interval without training the network in the range of first and end points. Therefore, the calculating volume involving computational error is reduced. In fact, the training points depending on the distance selected for training neural network are converted to similar points in the open interval by using a new approach, then the network is trained in these similar areas. In comparison with existing similar neural networks proposed model provides solutions with high accuracy. The proposed method is illustrated by four numerical examples.

Keywords: Fuzzy differential equation, Modified fuzzy neural network, Feedforward neural network, BFGS method, Hyperbolic tangent function.

1. INTRODUCTION
Nowadays, fuzzy differential equations are a popular topic studied by many researchers since it is utilized widely for the purpose of modeling problems in science and engineering. Most of the practical problems require the solution of a fuzzy differential equation which satisfies fuzzy initial or boundary conditions. The theory of fuzzy differential equations was treated by Buckley and Feuring [9], Kaleva [16,17], Nieto [28], Ouyang and Wu [32], Seikkala also recently there appeared the papers of Bede, Bede and Gal [8], Diamond [10,11], Georgion et al. [14] Nieto and Rodriguez-Lopez [29]. In the following, we have mentioned some numerical solutions which have proposed by other scientists. Abbasbandy and Allahviranloo have solved fuzzy differential equations by Runge-Kuta and Taylor methods [1, 2]. Also, Allahviranloo et al. solved differential equations by predictor-corrector and transformation methods [3, 4, 5]. Ghazanfari and Shakerami developed Runge-Kuta like formula of order 4 for solving fuzzy differential equations [13]. Nystrom method has been introduced for solving fuzzy differential equations [18].


In this work we propose a new solution method for the approximated solution of FDEs, this hybrid method can result in improved numerical methods for solving fuzzy differential equations. In this proposed method, fuzzy neural network model (FNNM) is applied as universal approximator. We use fuzzy trial function; this fuzzy trial function is a combination of two terms. A first term is responsible for the fuzzy condition while the second term contains the fuzzy neural network adjustable parameters to be calculated. The main aim of this paper is to illustrate how fuzzy connection weights are adjusted in the learning of fuzzy neural networks. Our fuzzy neural network in this paper is a three-Layer feed-forward neural network where connection weights and biases are fuzzy numbers.

*Corresponding Author: Dr. Eman A. Hussian
2. PRELIMINARIES

In this section the basic notations used in fuzzy calculus are introduced.

**Definition 2.1:** [23] A fuzzy number \( u \) is completely determined by any pair \( u = (\bar{u}, \tilde{u}) \) of functions \( \bar{u}(r), \tilde{u}(r) : \mathbb{R} \rightarrow [0,1] \) satisfying the conditions:

1. \( \bar{u}(r) \) is a bounded, monotonic, increasing (non-decreasing) left continuous function for all \( r \in (0,1) \) and right continuous for \( r=0 \).
2. \( \tilde{u}(r) \) is a bounded, monotonic, decreasing (non-increasing) left continuous function for all \( r \in (0,1) \) and right continuous for \( r=0 \).
3. For all \( r \in (0,1) \) we have \( \bar{u}(r) \leq \tilde{u}(r) \).

For every \( u = (\bar{u}, \tilde{u}), v = (\bar{v}, \tilde{v}) \) and \( k > 0 \) we define addition and multiplication as follows:

1. \( (u + v)(r) = \bar{u}(r) + \bar{v}(r) \) \hspace{1cm} (1)
2. \( (u + v)(r) = \tilde{u}(r) + \tilde{v}(r) \) \hspace{1cm} (2)
3. \( (ku)(r) = K \bar{u}(r), (ku)(r) = K \tilde{u}(r) \) \hspace{1cm} (3)

The collection of all fuzzy numbers with addition and multiplication as defined by Eqs. (1) \( \rightarrow \) (3) is denoted by \( \mathcal{E}^1 \).

For \( r \in (0,1) \), we define the \( r \)-cuts of fuzzy number \( u \) with \( [u]_r = \{ x \in \mathbb{R} | u(x) \geq r \} \) and for \( r = 0 \), the support of \( u \) is defined as \( [u]_0 = \{ x \in \mathbb{R} | u(x) > 0 \} \).

**Definition 2.2** [28]: The function \( f : \mathbb{R} \rightarrow \mathcal{E}^1 \) is called a fuzzy function. Now if, for an arbitrary fixed \( t_0 \in \mathbb{R} \) and \( \epsilon > 0 \) there exist a \( \delta > 0 \) such that: \( |t-t_0| < \delta \Rightarrow d(f(t), f(t_0)) < \epsilon \)

Then \( f \) is said to be continuous function.

**Definition 2.3** [12]: Let \( u, v \in \mathcal{E}^1 \). If there exist \( w \in \mathcal{E}^1 \) such that \( u = v + w \) then \( w \) is called the H-difference (Hukuhara-difference) of \( u, v \) and it is denoted by \( w = u \ominus v \).

In this paper the \( \Theta \) sign stands always for \( H \)-difference, and let us remark that \( u \ominus v \neq u + (-1)v \).

**Definition 2.4** [14]: Let \( f : [a, b] \rightarrow \mathcal{E}^1 \) and \( t_0 \in [a,b] \). We say that \( f \) is H-differential (Hukuhara-differential) at \( t_0 \), if there exists an element \( f(t_0) \in \mathcal{E}^1 \) such that for all \( h > 0 \) sufficiently small, \( \exists f(t_0 + h) \ominus f(t_0), f(t_0) \ominus f(t_0 - h) \) and the limits

\[
\lim_{h \to 0} \frac{f(t_0 + h) \ominus f(t_0)}{h} = \lim_{h \to 0} \frac{f(t_0) \ominus f(t_0 - h)}{h} = f'(t_0) .
\]

3. FUZZY NEURAL NETWORK

A fuzzy neural network or neuro-fuzzy system is a learning machine that finds the parameters of a fuzzy system (i.e. fuzzy sets, fuzzy rules) by exploiting approximation from neural network [23]. Combining fuzzy system with neural network. Both neural network and fuzzy system have some things in common. Artificial neural networks are an exciting form of the artificial intelligence which mimic the learning process of the human brain in order to extract patterns from historical data. Simple perceptrons need a teacher to tell the network what the desired output should by. These are supervised networks. In an unsupervised net, the network adapts purely in response to its input.

Such network can learn to pick out structure in their input. Fig.1 shows typical three-layered perceptron. multi-layered perceptrons with more than three layers, use more hidden layers. Multi-layered perceptrons correspond the input units to the output units by a specific nonlinear function. From kolmogorov existence theorem we know that a three-layered perceptron with \( n(2n+1) \) nodes can compute any continuous function of \( n \) variables [15] .

4. OPERATION OF FUZZY NUMBERS AND ACTIVATION FUNCTION

we briefly on mention fuzzy numbers operation defined by the extension principle. since input vector of feed-forward neural network is fuzzy in this paper, the following addition, multiplication and nonlinearity mapping of fuzzy number are necessary for defining our fuzzy neural network [23]:

1. \( m_{A+B} (z) = \max \{ m_A (x) \land m_B (y) \mid z = x + y \} \) \hspace{1cm} (5)
2. \( m_{A \cdot B} (z) = \max \{ m_A (x) \land m_B (y) \mid z = x \cdot y \} \) \hspace{1cm} (6)
3. \( m_{f(x)} (z) = \max \{ m_{act} (x) \mid z = f(x) \} \) \hspace{1cm} (7)

© 2015, IJMA. All Rights Reserved
Where A, B and net are fuzzy number, \( \mu (\ast) \) denotes the membership function of each fuzzy number, \( \lambda \) is the Minimum operator and \( f(\cdot) \) is a continuous activation function (such as Hyperbolic tangent function) inside the hidden neurons. the above operations of fuzzy numbers are numerically performed on level sets (i.e. r-cuts).

The r-level set of a fuzzy number A is defined as:

\[
[A]_r = \{ x \in \mathbb{R} \mid \mu_A(x) \geq r \}, 0 < r \leq 1
\]  

Since level sets of fuzzy numbers become closed intervals we denote \([A]_r\) as:

\[
[A]_r = [A]_L^r, [A]_U^r
\]

Where \([A]_L^r\) and \([A]_U^r\) are the lower limit and the upper limit of the r-level set \([A]_r\) respectively [30], from interval arithmetic, the above operations of fuzzy number are written for r-level set as follows:

\[
[A]_r + [B]_r = [A]_L^r + [B]_L^r, [A]_U^r + [B]_U^r
\]

\[
[A]_r \cdot [B]_r = \text{Min} \{[A]_L^r, [B]_L^r, [A]_U^r, [B]_U^r\}, \text{Max} \{[A]_L^r, [B]_L^r, [A]_U^r, [B]_U^r\}
\]

\[f([\text{net}]_r) = f\left(\left([\text{net}]_L^r, [\text{net}]_U^r\right)\right) = f\left([\text{net}]_L^r\right), f\left([\text{net}]_U^r\right)\]

Remark 4.1: In the case of \(0 \leq [B]_L^r \leq [B]_U^r\) Eq(10) can be simplified as:

\[
[A]_r \cdot [B]_r = \text{Min} \{[A]_L^r, [B]_L^r, [A]_U^r, [B]_U^r\}, \text{Max} \{[A]_L^r, [B]_L^r, [A]_U^r, [B]_U^r\}
\]

5. NUMERICAL SOLUTION OF FUZZY DIFFERENTIAL EQUATIONS BASED ON THE POLYNOMIAL

\(H(x) = \varepsilon(x^2 + x + 1)\) BY USING FUZZY NEURAL NETWORK

In this section we will discuss how we can use the fuzzy neural networks based on the polynomial \(H(x) = \varepsilon(x^2 + x + 1)\), \(\varepsilon \in (0, 1)\), to solve the fuzzy differential equations. Our fuzzy neural network is a three-layered feed forward neural network where the connections weights, biases and targets are given as fuzzy numbers and inputs are given as real numbers. for convenience in this discussion, FNNM with an input layer, a single hidden layer, and an output Layer is represented as a basic structural architecture. Here the dimension of FNNM is denoted by the number of neurons in each layer, that is \(n \times m \times s\), where \(n, m\) and \(s\) are the number of the neurons in the input layer, the hidden layer and the output layer, respectively(see Fig.1) The architecture of the model shows how FNNM transform the n inputs \((x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n)\) into the s outputs \((y_1, y_2, \ldots, y_k, y_{k+1}, \ldots, y_s)\) throughout the m hidden neurons \((Z_1, Z_2, \ldots, Z_j, Z_{j+1}, \ldots, Z_m)\) where the cycles represent the neurons in each Layer. Let \(B_j\) be the bias for the neurons \(Z_j\), and let the \(C_k\) be the bias for the neurons \(y_k\), and let \(W_{ij}\) be the Weights connecting the neurons \(x_i\) to the neurons \(Z_j\) to the neurons \(y_k\). When an n-dimensional input vector \((x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_n)\) is presented to our fuzzy neural network, its Input-output relations can be written as:
Input Units:
\[ o_i = H(x_i) = \epsilon (x_i^2 + x_i + 1), \quad i = 1, \ldots, n, \quad \epsilon \in (0,1) \]  
(12)

Hidden units:
\[ Z_j = f(\text{net}_j), \quad j = 1, 2, \ldots, m \]  
(13)
\[ \text{net}_j = \sum_{i=1}^{n} o_i W_{ji} + B_j \]  
(14)

Output units:
\[ y_k = f(\text{net}_k), \quad k = 1, 2, \ldots, s \]  
(15)
\[ \text{net}_k = \sum_{j=1}^{m} W_{kj} \cdot Z_j + C_k \]  
(16)

Where connection weights, biases and target are fuzzy numbers and the inputs are real numbers. The input-output relations in Eqs (12) - (16) is defined by the extension principle. The fuzzy output from each unit in Eqs (12) - (16) is numerically calculated for real inputs and level sets of fuzzy weights and fuzzy biases. The input-output relations of our fuzzy neural network can be written for r-level sets:

Input Units:
\[ o_i = H(x_i) = \epsilon (x_i^2 + x_i + 1), \quad i = 1, \ldots, n, \quad \epsilon \in (0,1) \]  
(17)

Hidden Unit:
\[ [Z_j]_r = f([\text{net}_j]_r), \quad j = 1, 2, \ldots, m \]  
(18)
\[ [\text{net}_j]_r = \sum_{i=1}^{n} o_i [W_{ji}]_r + [B_j]_r \]  
(19)

Output Units:
\[ [y_k]_r = f([\text{net}_k]_r), \quad k = 1, 2, \ldots, s \]  
(20)
\[ [\text{net}_k]_r = \sum_{j=1}^{m} [W_{kj}]_r \cdot [Z_j]_r + [C_k]_r \]  
(21)

From Eqs (17) - (21), we can see that the r-level sets of the fuzzy outputs \( y_k \)'s are calculated from those of the fuzzy weights, fuzzy biases and crisp inputs. From the operations of fuzzy numbers, the above relations are rewritten as follows when the inputs \( x_i \)'s are non-negative, i.e., \( x_i \geq 0 \):

Input Units:
\[ o_i = H(x_i) = \epsilon (x_i^2 + x_i + 1), \quad i = 1, \ldots, n, \quad \epsilon \in (0,1) \]  
(22)

Hidden Unit:
\[ [Z_j]_r = [([Z_j])^L_r, [Z_j])^U_r] = [f([\text{net}_j])^L_{r'}, f([\text{net}_j])^U_{r'}] \]  
(23)
\[ [\text{net}_j]_r^L = \sum_{i=1}^{n} o_i [W_{ji}]^L_r + [B_j]^L_r \]  
(24)
\[ [\text{net}_j]_r^U = \sum_{i=1}^{n} o_i [W_{ji}]^U_r + [B_j]^U_r \]  
(25)

Output Units:
\[ [y_k]_r = [([y_k])^L_r, [y_k]^U_r] = [f([\text{net}_k])^L_{r'}, f([\text{net}_k])^U_{r'}] \]  
(26)
\[ [\text{net}_k]_r^L = \sum_{j=\ell} \sum_{a} [W_{kj}]^L_r \cdot [Z_j]^L_r + \sum_{j=\ell} \sum_{b} [W_{kj}]^U_r \cdot [Z_j]^U_r + [C_k]^L_r \]  
(27)
\[ [\text{net}_k]_r^U = \sum_{j=\ell} \sum_{a} [W_{kj}]^U_r \cdot [Z_j]^L_r + \sum_{j=\ell} \sum_{d} [W_{kj}]^L_r \cdot [Z_j]^U_r + [C_k]^U_r \]  
(28)

For \([Z_j]^L_r \geq [Z_j]^U_r \geq 0, \) where \( a = \{ j : [W_{kj}]^L_r \geq 0 \}, \) \( b = \{ j : [W_{kj}]^U_r < 0 \}, \) \( c = \{ j : [W_{kj}]^U_r \geq 0 \}, \) \( d = \{ j : [W_{kj}]^U_r < 0 \}, \) and \( a \cup b = \{ 1, \ldots, m \} \) and \( c \cup d = \{ 1, \ldots, m \} \).

One drawback of the fully fuzzy neural network with fuzzy connection weights is long computation time. Another drawback is that the learning algorithm is complicated. For reducing the complexity of the learning algorithm, we propose a partially fuzzy neural network (PFNN) architecture, where connection weights to output units are fuzzy numbers while connection weights and biases to hidden units are real numbers [20,23,24]. In this paper, the (PFNN) is of dimension \((1 \times m \times 1)\) (see Fig. (2)).

For every entry \( x \), the input to the hidden neurons is
\[ \text{net}_j = \epsilon (x^2 + x + 1) \cdot W_j + B_j, \quad j = 1, 2, \ldots, m, \quad \epsilon \in (0,1) \]  
(29)
where \( W_j \) is a weight parameter from input layer to the \( j \)th unit in the hidden layer, \( B_j \) is an \( j \)th bias for the unit in the hidden layer. The output, in the hidden neurons is

\[
Z_j = s(\text{net}_j), j = 1, 2, \ldots, m
\]

where \( s(\cdot) \) is the hyperbolic tangent activation function, and the output \( N(x, p) \) of the network is

\[
N = \sum_{j=1}^{m} v_j Z_j
\]

Where \( v_j \) is a weight parameter from the \( j \)th unit in the hidden layer to the output layer. From Eqs (22)-(28), we can be rewritten for \( r \)-level sets of the Eqs (29)-(31). For reducing the complexity of the learning algorithm, the input \( x \) usually assumed as non-negative in fuzzy neural network:

**Input Unit:**

\[
o = H(x) = \epsilon (x^2 + x + 1), \epsilon \in (0, 1)
\]

**Hidden Unit:**

\[
Z_j = s(\text{net}_j), j = 1, 2, \ldots, m
\]

\[
\text{net}_j = o W_j + B_j
\]

**Output Unit:**

\[
[N]_r = \left[ [N]^L_r, [N]^U_r \right] = \left[ \sum_{j=1}^{m} [v_j]^L Z_j, \sum_{j=1}^{m} [v_j]^U Z_j \right]
\]

### 5.1. Solution of First Order Fuzzy Differential Equation

Let us consider the FDE:

\[
y' = f(x, y), \quad x \in [a, b] \text{ and } y(a) = A
\]

Where \( y \) is a fuzzy function of \( x \), \( f(x, y) \) is a fuzzy function of the crisp variable \( x \) and the fuzzy variable \( y \), and \( y' \) is the fuzzy derivative of \( y \), and \( A \) is a fuzzy number in \( E^1 \) with \( r \)-level set:

\[
[A]^L_r = \left[ [A]^L_r, [A]^U_r \right], \quad r \in (0, 1]
\]

the related trial function will be in the form:

\[
y_t(x, p) = A + (x - a) N(x, p),
\]

this solution by intention satisfies the initial condition in Eq. (36).

The error function that must be minimized for problem (36) has the form [34, 38, 39]:

\[
E = \sum_{r=1}^{g} \left( E^L_r + E^U_r \right),
\]

Where:

\[
E^L_r = \left( \left( \frac{dy_t(x, p)}{dx} \right)^L - \left( f(x, y_t(x, p)) \right)^L \right)^2
\]

\[
E^U_r = \left( \left( \frac{dy_t(x, p)}{dx} \right)^U - \left( f(x, y_t(x, p)) \right)^U \right)^2
\]
and \( \{x_i\}_{i=1}^g \) are discrete points belonging to the interval \([a, b]\), \( E_{ir}^L \) and \( E_{ir}^U \) can be viewed as the squared errors for the lower and the upper limits of the \( r \)-level sets.

Now differentiating the trial function \( y_t(x, p) \) in Eq (37), we obtain:

\[
\frac{\partial [y_t(x, p)]^L}{\partial x} = [N(x, p)]^L + (x-a) \frac{\partial [N(x, p)]^L}{\partial x} \tag{40}
\]

\[
\frac{\partial [y_t(x, p)]^U}{\partial x} = [N(x, p)]^U + (x-a) \frac{\partial [N(x, p)]^U}{\partial x} \tag{41}
\]

where

\[
[N(x, p)]^L = \sum_{j=1}^m [v_j]^L \ s(H(x)W_j + B_j) \tag{42}
\]

\[
[N(x, p)]^U = \sum_{j=1}^m [v_j]^U \ s(H(x)W_j + B_j) \tag{43}
\]

\[
\frac{\partial [N(x, p)]^L}{\partial x} = \sum_{j=1}^m \varepsilon (2x_i + 1)W_j [v_j]^L \ s'(H(x)W_j + B_j) \tag{44}
\]

\[
\frac{\partial [N(x, p)]^U}{\partial x} = \sum_{j=1}^m \varepsilon (2x_i + 1)W_j [v_j]^U \ s'(H(x)W_j + B_j) \tag{45}
\]

5.2. Solution of Second Order Fuzzy Differential Equation

Now, we consider the second order fuzzy differential equation:

\[
y'' = f(x, y, y'), x \in [a, b] \tag{46}
\]

\[
y(a) = A, y'(a) = B. \tag{47}
\]

such that the functions:

\[ y: [a, b] \rightarrow E^1 \quad \text{and} \quad f: [a, b] \times E^1 \times E^1 \rightarrow E^1 \]

where \( y \) is a function with fuzzy derivative \( y' \), also \( A \) and \( B \) are fuzzy numbers in \( E^1 \) with \( r \)-level sets:


The related trial function will be in the form:

\[
y_t(x, p) = A + B(x-a) + (x-a)^2 N(x, p), \tag{48}
\]

This solution by intention satisfies the conditions in Eq. (46).

Also, the error function that must be minimized for problem (46) is

\[ E = \sum_{i=1}^g (E_{ir}^L + E_{ir}^U), \]

where

\[ E_{ir}^L = (y'' - f(x, y(x, p), y'(x, p)))^2 \tag{49}
\]

\[ E_{ir}^U = (y'' - f(x, y(x, p), y'(x, p)))^2 \tag{50}
\]

and \( \{x_i\}_{i=1}^g \) are discrete points belonging to the interval \([a, b]\).

Now differentiating the trial function \( y_t(x, p) \) in eq. (47), we obtain:

\[
\frac{\partial [y_t(x, p)]^L}{\partial x} = [B]^L + 2(x-a)[N(x, p)]^L + (x-a)^2 \frac{\partial [N(x, p)]^L}{\partial x} \tag{51}
\]

\[
\frac{\partial [y_t(x, p)]^U}{\partial x} = [B]^U + 2(x-a)[N(x, p)]^U + (x-a)^2 \frac{\partial [N(x, p)]^U}{\partial x} \tag{52}
\]

\[
\frac{\partial^2 [y_t(x, p)]^L}{\partial x^2} = 2[N(x, p)]^L + 4(x-a) \frac{\partial [N(x, p)]^L}{\partial x} + (x-a)^2 \frac{\partial^2 [N(x, p)]^L}{\partial x^2} \tag{53}
\]

\[
\frac{\partial^2 [y_t(x, p)]^U}{\partial x^2} = 2[N(x, p)]^U + 4(x-a) \frac{\partial [N(x, p)]^U}{\partial x} + (x-a)^2 \frac{\partial^2 [N(x, p)]^U}{\partial x^2} \tag{54}
\]

where

\[
[N(x, p)]^L = \sum_{j=1}^m [v_j]^L \ s(H(x)W_j + B_j) \tag{55}
\]

\[
[N(x, p)]^U = \sum_{j=1}^m [v_j]^U \ s(H(x)W_j + B_j) \tag{56}
\]

\[
\frac{\partial [N(x, p)]^L}{\partial x} = \sum_{j=1}^m \varepsilon (2x + 1)W_j [v_j]^L \ s'(H(x)W_j + B_j) \tag{57}
\]

\[
\frac{\partial [N(x, p)]^U}{\partial x} = \sum_{j=1}^m \varepsilon (2x + 1)W_j [v_j]^U \ s'(H(x)W_j + B_j) \tag{58}
\]

To find a numerical solution for third (and more) order fuzzy differential equation and and fuzzy partial differential equation by using this method, we apply the same procedure in the subsections (5.1) and (5.2).
6. NUMERICAL EXAMPLES

To show the behavior and properties of the new method, four problem will be solved in this subsection. For each example, the accuracy of the method is illustrated by computing the deviations $E(x, r)$ and $\bar{E}(x, r)$

Where $E(x, r) = |y_e(x, r) - y_a(x, r)|$ and $\bar{E}(x, r) = |\bar{y}_e(x, r) - \bar{y}_a(x, r)|$.

And $y_a(x, r) = \left[ y_a(x, r) \right]$ is the known exact solution and $y_e(x, r) = \left[ y_e(x, r) \right]$ is the trial (approximated) solution. Note that, for all examples, a multilayer perceptron consisting of one hidden layer with 10 hidden units and one linear output unit is used. We will use the quasi–Newton method to minimize the error function.

**Example 1:** Consider the following fuzzy initial value problem:

$y' = (1 + 0.01 r) - x$, $x \in [0, 1]$

$y(0)_r = [0.96 + 0.04 r, 1.01 - 0.01 r]$, where $r \in [0, 1]$.

The fuzzy exact solution is:

$y_a(x, r) = [x + (0.96 + 0.04 r)e^{-x}, x + (1.01 - 0.01 r)e^{-x}]$

The fuzzy trial solution for this problem is:

$y_e(x, r) = [0.96 + 0.04 r, 1.01 - 0.01 r] + x \left[ N(x, p) \right]_r$, Fig. (1) shows the exact solution and the approximated solution for $x = 0.1$ and $\epsilon = 0.6$. Numerical results for $x = 0.2$ and $r = 0.5$ can be found in table (1).

Note that for $x = 0.2$ and $r = 0.5$, the fuzzy exact solution is

$y_a(0.2, 0.5) = [1.0023561, 1.0228244]$, and $H(x) = \epsilon (x^2 + x + 1) = 1.24 \epsilon$, $\epsilon \in (0, 1)$.

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>$y_e(0.2, 0.5)$</th>
<th>$E(0.2, 0.5)$</th>
<th>$\bar{E}(0.2, 0.5)$</th>
<th>$\bar{E}(0.2, 0.5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.02372368</td>
<td>0.02136754</td>
<td>0.06426973</td>
<td>0.01944532</td>
</tr>
<tr>
<td>0.2</td>
<td>1.01359991</td>
<td>0.01124377</td>
<td>0.01386140</td>
<td>0.01578959</td>
</tr>
<tr>
<td>0.3</td>
<td>1.00667758</td>
<td>0.00432144</td>
<td>0.01237469</td>
<td>0.00092255</td>
</tr>
<tr>
<td>0.4</td>
<td>1.00646736</td>
<td>0.00411122</td>
<td>0.01316802</td>
<td>0.00085621</td>
</tr>
<tr>
<td>0.5</td>
<td>1.00312854</td>
<td>0.00077240</td>
<td>0.01233592</td>
<td>0.00053479</td>
</tr>
<tr>
<td>0.6</td>
<td>1.00238529</td>
<td>0.00002915</td>
<td>0.0102284285</td>
<td>0.00001844</td>
</tr>
<tr>
<td>0.7</td>
<td>1.01221038</td>
<td>0.00985424</td>
<td>0.013882331</td>
<td>0.01599890</td>
</tr>
<tr>
<td>0.8</td>
<td>1.02857729</td>
<td>0.02622115</td>
<td>0.0106269903</td>
<td>0.03987462</td>
</tr>
<tr>
<td>0.9</td>
<td>1.03357745</td>
<td>0.03122131</td>
<td>0.0109397723</td>
<td>0.07115282</td>
</tr>
</tbody>
</table>

**Table-1:** Comparison of the exact and approximated solution for example (1), for $x = 0.2$ and $r = 0.5$.

**Example 2:** Consider the following nonlinear FIVP:

$y'(x) = 3Ay^2$, $x \in [0, 1]$

$y(0)_r = [0.5 \sqrt{r} - 0.2 \sqrt{1 - r} + 0.5]$, where $r \in [0, 1]$, and $A = [1 + r, 3 - r]$ is a fuzzy parameter.

For this problem, the fuzzy exact solution is:

$y_a(x, r) = \left[ \frac{0.5 \sqrt{r}}{1 - 3(1 + r) (0.5 \sqrt{r})^x}, \frac{0.2 \sqrt{1 - r} + 0.5}{1 - 3(3 + r) (0.2 \sqrt{1 - r} + 0.5)^x} \right]$

while the fuzzy trial solution is:

$y_e(x, r) = [0.5 \sqrt{r} - 0.2 \sqrt{1 - r} + 0.5] + x \left[ N(x, p) \right]_r$

Fig. (2) shows the exact solution and the approximated solution for $x = 0.1$ and $\epsilon = 0.7$. Numerical results can be found in table (2). Note that for $r = 0.2$, the fuzzy exact solution is:

$y_a(0.2, 0.2) = \left[ 0.22360680, 0.67888544 \right]$, $H(x) = 0.7(x^2 + x + 1)$.
Table-2: Comparison of the exact and approximated solution for example (2), for \( r = 0.2 \) and \( \epsilon = 0.7 \).

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y_0(x,0.2) )</th>
<th>( y_1(x,0.2) )</th>
<th>( E(x,0.2) )</th>
<th>( \bar{y}_0(x,0.2) )</th>
<th>( \bar{y}_1(x,0.2) )</th>
<th>( \bar{E}(x,0.2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2236068</td>
<td>0.2236068</td>
<td>0.0000000</td>
<td>0.6788854</td>
<td>0.6788854</td>
<td>0.0000000</td>
</tr>
<tr>
<td>0.01</td>
<td>0.2254214</td>
<td>0.2254873</td>
<td>0.0000659</td>
<td>0.7199411</td>
<td>0.7199199</td>
<td>0.0000212</td>
</tr>
<tr>
<td>0.02</td>
<td>0.2272657</td>
<td>0.2272736</td>
<td>0.000079</td>
<td>0.7662820</td>
<td>0.7662744</td>
<td>0.0000076</td>
</tr>
<tr>
<td>0.03</td>
<td>0.2291404</td>
<td>0.2291883</td>
<td>0.0000479</td>
<td>0.8189991</td>
<td>0.8190363</td>
<td>0.0000372</td>
</tr>
<tr>
<td>0.04</td>
<td>0.2310463</td>
<td>0.2310662</td>
<td>0.000199</td>
<td>0.8795055</td>
<td>0.8794966</td>
<td>0.000089</td>
</tr>
<tr>
<td>0.05</td>
<td>0.2329842</td>
<td>0.2329901</td>
<td>0.000059</td>
<td>0.9496653</td>
<td>0.9496574</td>
<td>0.000079</td>
</tr>
<tr>
<td>0.06</td>
<td>0.2349549</td>
<td>0.2350173</td>
<td>0.000624</td>
<td>1.0319090</td>
<td>1.0320011</td>
<td>0.000121</td>
</tr>
<tr>
<td>0.07</td>
<td>0.2369592</td>
<td>0.2369680</td>
<td>0.000088</td>
<td>1.1299402</td>
<td>1.1299366</td>
<td>0.000036</td>
</tr>
<tr>
<td>0.08</td>
<td>0.2389980</td>
<td>0.2389134</td>
<td>0.000846</td>
<td>1.2484354</td>
<td>1.2484338</td>
<td>0.000016</td>
</tr>
<tr>
<td>0.09</td>
<td>0.2410721</td>
<td>0.2410563</td>
<td>0.000158</td>
<td>1.3946951</td>
<td>1.3946728</td>
<td>0.0000223</td>
</tr>
<tr>
<td>0.10</td>
<td>0.2431826</td>
<td>0.2431805</td>
<td>0.000021</td>
<td>1.5797724</td>
<td>1.5797694</td>
<td>0.000030</td>
</tr>
</tbody>
</table>

Fig. -1: Analytical and trial solutions for example (1), for \( x = 0.1 \) and \( \epsilon = 0.6 \)

Fig.-2: Analytical and trial solutions for example (2) for \( x = 0.1 \) and \( \epsilon = 0.7 \)

Example 3: Consider the homogeneous second order FDE:
\[
y(\ddot{x}) - 4y(\dot{x}) + 4y(x) = 0, \quad x \in [0,1].
\]
\[
y(0), = [2 + r, 4 - r], \quad [y(0)], = [5 + r, 7 - r], \quad r \in [0,1].
\]
The fuzzy exact solution is:

\[ y_0(x, r) = [(2 + r)e^{2x} + (1-r)xe^{2x}, (4-r)e^{2x} + (r-1)xe^{2x}] \]

The fuzzy trial solution for this problem is:

\[ y_t(x, r) = [2 + r, 4 - r] + [5 + r, 7 - r]x + x^2 \cdot [N(x, p)]_r \]

Table (3) shows the exact solution and the trial solution for \( x = 0.1 \) and \( r = 0.5 \). Numerical results for \( \varepsilon = 0.4 \) and \( r = 0.5 \) can be found in table (4).

Note that for \( x = 0.1 \) and \( r = 0.5 \), the fuzzy exact solution is:

\[ y_e(0.1, 0.5) = [3.11457703, 4.21383952], \quad H(x) = 1.11e \]

and for \( \varepsilon = 0.4 \) and \( r = 0.5 \), the fuzzy exact solution is:

\[ y_e(x, 0.5) = [2.5 e^{2x} + 0.5 xe^{2x}, 3.5 e^{2x} - 0.5 x e^{2x}], \quad H(x) = 0.4 (x^2 + x + 1). \]

### Table-3: Comparison of the exact and trial solution for example (3), for \( x = 0.1, r = 0.5 \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y_e(x, 0.5) )</th>
<th>( y_t(x, 0.5) )</th>
<th>( E(0.1, 0.5) )</th>
<th>( y_e(x, 0.5) )</th>
<th>( y_t(x, 0.5) )</th>
<th>( E(0.1, 0.5) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.5000000</td>
<td>2.5000000</td>
<td>0.0000000</td>
<td>3.5000000</td>
<td>3.5000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>0.1</td>
<td>3.1145770</td>
<td>3.1145794</td>
<td>0.0000176</td>
<td>4.2138395</td>
<td>4.2137520</td>
<td>0.0000875</td>
</tr>
<tr>
<td>0.2</td>
<td>3.8787442</td>
<td>3.8788188</td>
<td>0.0000746</td>
<td>5.0722040</td>
<td>5.0722893</td>
<td>0.000853</td>
</tr>
<tr>
<td>0.3</td>
<td>4.8286148</td>
<td>4.8285781</td>
<td>0.0000367</td>
<td>6.1040980</td>
<td>6.1041436</td>
<td>0.0000456</td>
</tr>
<tr>
<td>0.4</td>
<td>6.0089605</td>
<td>6.0089507</td>
<td>0.0000998</td>
<td>7.3442851</td>
<td>7.3442226</td>
<td>0.0000625</td>
</tr>
<tr>
<td>0.5</td>
<td>7.4752750</td>
<td>7.4752796</td>
<td>0.0000406</td>
<td>8.8341589</td>
<td>8.8341835</td>
<td>0.0000266</td>
</tr>
<tr>
<td>0.6</td>
<td>9.2963274</td>
<td>9.2963053</td>
<td>0.0000221</td>
<td>10.6243742</td>
<td>10.6244242</td>
<td>0.0000482</td>
</tr>
<tr>
<td>0.7</td>
<td>11.5573199</td>
<td>11.5573149</td>
<td>0.0000050</td>
<td>12.7738799</td>
<td>12.7738714</td>
<td>0.0000855</td>
</tr>
<tr>
<td>0.8</td>
<td>14.3637940</td>
<td>14.3637205</td>
<td>0.0000735</td>
<td>15.3544005</td>
<td>15.3544753</td>
<td>0.0000748</td>
</tr>
<tr>
<td>0.9</td>
<td>17.8464600</td>
<td>17.8464675</td>
<td>0.0000075</td>
<td>18.4514248</td>
<td>18.4514645</td>
<td>0.0000397</td>
</tr>
<tr>
<td>1</td>
<td>22.1671683</td>
<td>22.1671053</td>
<td>0.0000630</td>
<td>22.1671683</td>
<td>22.1671535</td>
<td>0.0000148</td>
</tr>
</tbody>
</table>

### Table-4: Comparison of the exact and trial solution for example (3), for \( \varepsilon = 0.4, r = 0.5 \)

**Example (4):** Consider the nonhomogeneous second order FDE:

\[ y'' + 4y = 2\cos x, \quad x \in [0, 1], \]

\[ y(0), y'(0) = [2r, 4-2r], \quad y'(0) = [2r, 4-2r], \quad r \in [0, 1]. \]

The fuzzy exact solution is:

\[ y_e(x, r) = [(2 - \frac{r}{2}) \cos 2x + (r - 1) \sin 2x + \frac{2}{3} \cos x, (\frac{10}{3} - 2r) \cos 2x + (1 - r) \sin 2x + \frac{2}{3} \cos x]. \]

The fuzzy trial solution for this problem is:

\[ y_t(x, r) = [2r, 4-2r] + [-2 + 2r, 2-2r]x + x^2 \cdot [N(x, p)]_r. \]

Table (5) shows the exact and the trial solution for \( x = 1 \) and \( r = 0.7 \).

Numerical results for \( \varepsilon = 0.8 \) and \( r = 0.7 \) can be found in table (6).

Note that for \( x = 1 \) and \( r = 0.7 \), the fuzzy exact solution is:

\[ y_e(1, 0.7) = [-0.21776204, -0.17155979], \quad H(x) = 3\varepsilon \]

And for \( \varepsilon = 0.8, r = 0.7 \), the fuzzy exact solution is

\[ y_e(x, 0.7) = [0.73333 \cos 2x - 0.3 \sin 2x + \frac{2}{3} \cos x, 1.93333 \cos 2x + 0.3 \sin 2x + \frac{2}{3} \cos x]. \]
In this paper, we presented a hybrid approach based on modified fuzzy neural networks for solving fuzzy differential equations. We demonstrate, for the first time, the ability of modified fuzzy neural networks to approximate the solutions of FDEs. By comparing our results with the results obtained by using numerical methods, it can be observed that the proposed method yields more accurate approximations. Even better results may be possible if one uses more neurons or more training points. Moreover, after solving a FDE the solution is obtainable at any arbitrary point in the training interval.

Table-5: Comparison of the exact and the trial solution for example (4) for $x = 1$ and $r = 0.7$.

<table>
<thead>
<tr>
<th>$y(x,0.7)$</th>
<th>$\hat{y}(x,0.7)$</th>
<th>$E(x,0.7)$</th>
<th>$\hat{y}(x,0.7)$</th>
<th>$\hat{E}(x,0.7)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>-0.3028857</td>
<td>0.0851237</td>
<td>-0.1216841</td>
<td>0.0498757</td>
</tr>
<tr>
<td>0.2</td>
<td>-0.2135259</td>
<td>0.0042361</td>
<td>-0.1456746</td>
<td>0.0258852</td>
</tr>
<tr>
<td>0.3</td>
<td>-0.2256072</td>
<td>0.0078451</td>
<td>-0.1723489</td>
<td>0.0007891</td>
</tr>
<tr>
<td>0.4</td>
<td>-0.2166399</td>
<td>0.0111221</td>
<td>-0.1719289</td>
<td>0.0003691</td>
</tr>
<tr>
<td>0.5</td>
<td>-0.2177031</td>
<td>0.0000590</td>
<td>-0.1716162</td>
<td>0.0000564</td>
</tr>
<tr>
<td>0.6</td>
<td>-0.2177965</td>
<td>0.0000344</td>
<td>-0.1715105</td>
<td>0.0000493</td>
</tr>
<tr>
<td>0.7</td>
<td>-0.2177688</td>
<td>0.0000686</td>
<td>-0.1715653</td>
<td>0.0000555</td>
</tr>
<tr>
<td>0.8</td>
<td>-0.2177586</td>
<td>0.0000344</td>
<td>-0.1715626</td>
<td>0.0000285</td>
</tr>
<tr>
<td>0.9</td>
<td>-0.2176746</td>
<td>0.0000875</td>
<td>-0.1720495</td>
<td>0.0004898</td>
</tr>
</tbody>
</table>

Table-6: Comparison of the exact and the trial solution for example (4), for $\epsilon = 0.8$ and $r = 0.7$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y(x,0.7)$</th>
<th>$\hat{y}(x,0.7)$</th>
<th>$E(x,0.7)$</th>
<th>$\hat{y}(x,0.7)$</th>
<th>$\hat{E}(x,0.7)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.4000000</td>
<td>1.4000000</td>
<td>0.0000000</td>
<td>2.6000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>0.1</td>
<td>1.3224508</td>
<td>1.3223751</td>
<td>0.0000757</td>
<td>2.6177966</td>
<td>0.000643</td>
</tr>
<tr>
<td>0.2</td>
<td>1.2119969</td>
<td>1.2120015</td>
<td>0.0000464</td>
<td>2.5509211</td>
<td>0.000070</td>
</tr>
<tr>
<td>0.3</td>
<td>1.0727444</td>
<td>1.0727385</td>
<td>0.0000592</td>
<td>1.9773348</td>
<td>0.0000038</td>
</tr>
<tr>
<td>0.4</td>
<td>0.9097521</td>
<td>0.9097470</td>
<td>0.0000514</td>
<td>2.1762138</td>
<td>0.0000289</td>
</tr>
<tr>
<td>0.5</td>
<td>0.7288354</td>
<td>0.7289057</td>
<td>0.0000705</td>
<td>1.8820808</td>
<td>0.0000989</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5363410</td>
<td>0.5363339</td>
<td>0.0000720</td>
<td>1.5303938</td>
<td>0.000023</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3389024</td>
<td>0.3389635</td>
<td>0.0000611</td>
<td>1.1341329</td>
<td>0.000478</td>
</tr>
<tr>
<td>0.8</td>
<td>0.1431861</td>
<td>0.1431835</td>
<td>0.0000260</td>
<td>0.7078908</td>
<td>0.000088</td>
</tr>
<tr>
<td>0.9</td>
<td>-0.044363</td>
<td>-0.044339</td>
<td>0.0002400</td>
<td>0.2673036</td>
<td>0.000026</td>
</tr>
<tr>
<td>1</td>
<td>-0.217762</td>
<td>-0.217759</td>
<td>0.0000300</td>
<td>-0.171559</td>
<td>0.000153</td>
</tr>
</tbody>
</table>

8. CONCLUSIONS

In this paper, we presented a hybrid approach based on modified fuzzy neural networks for solving fuzzy differential equations. We demonstrate, for the first time, the ability of modified fuzzy neural networks to approximate the solutions of FDEs. By comparing our results with the results obtained by using numerical methods, it can be observed that the proposed method yields more accurate approximations. Even better results may be possible if one uses more neurons or more training points. Moreover, after solving a FDE the solution is obtainable at any arbitrary point in the training interval (even between training points). The main reason for using modified fuzzy neural networks was their applicability in function approximation. Further research is in progress to apply and extend this method to solve fuzzy partial differential equations FPDEs.

REFERENCES


Source of support: Nil, Conflict of interest: None Declared

[Copy right © 2015. This is an Open Access article distributed under the terms of the International Journal of Mathematical Archive (IJMA), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.]