ISSN: 0067-2904

Efficient Algorithm for Solving Fuzzy Singularly Perturbed Volterra Integro-Differential Equation

Khalid Mindeel Mohammed Al-abrahemee
Department of Mathematics, College of Education, AL-Qadisiyah University, AL-Qadisiyah, Iraq

Received: 3/10/2022 Accepted: 15/4/2023 Published: 30/11/2023

Abstract:
In this paper, we design a fuzzy neural network to solve fuzzy singularly perturbed Volterra integro-differential equation by using a High Performance Training Algorithm such as the Levenberge-Marqaurdt (TrianLM) and the sigmoid function of the hidden units which is the hyperbolic tangent activation function. A fuzzy trial solution to fuzzy singularly perturbed Volterra integro-differential equation is written as a sum of two components. The first component meets the fuzzy requirements, however, it does not have any fuzzy adjustable parameters. The second component is a feed-forward fuzzy neural network with fuzzy adjustable parameters. The proposed method is compared with the analytical solutions. We find that the proposed method has excellent accuracy in findings, a lower error rate, and faster convergence than other typical methods.

Keywords: Fuzzy Singularly perturbed Volterra integro-differential equations, Singularly perturbed problems, Volterra integro-differential equations, Fuzzy neural network, Levenberge-Marqaurdt Algorithm.

خوارزمية فعالة لحل معادلة فولتيرا التكاملية التفاضلية المضطربه الشاذه

خالد منديل محمد الابراهيمي
قسم الرياضيات، كلية التربية، جامعة القادسية، العراق

الخلاصة
في هذا البحث ، قمنا بتصميم شبكة عصبية ضبابية لحل معادلة فولتيرا التفاضلية التكاملية المضطربة الشاذه . استخدامت خوارزمية تدريب عالية الأداء مثل (Levenberge– Marqaurdt (TrianLM) ودالة sigmoid للوحدات الخفيه هي دالة التنشيط الظل الزائدية الحل التجريبي الضبابي الخاص بـ معادلة فولتيرا التفاضلية التكاملية المضطربة الشاذه كتب على شكل مجموع مركبين. المركب الأول يحقق الشروط الضبابية ولكن لا يحتوي على أي معلمات ضبابية قابلة للتغيير. المركب الثاني عبارة عن شبكة عصبية ضبابية تقدمية مع معلمات ضبابية قابلة للتعديل. تمت مقارنة الطريقة المقترحة مع الحلول التحليلية. وجدنا أن الطريقة المقترحة لها دقة ممتازة في النتائج ، ومعدل خطأ أقل ، وتقارب اسرع من الطرق التقليدية.

1. Introduction:

We use in this article a constructive interference of the artificial intelligence and fuzzy logic that appear in the form of a fuzzy artificial neural network (FANN). An initial attempts to construct the artificial intelligence (AI) and neural network concept that is accomplished by

* Email: khalid.mohammed@qu.edu.iq

Werren McCulloach and Walter Pitts in 1943 [1]. In 1949, Donald Hebb developed a neuropsychological theory and its rule [2] that is known as Hebbian learning [3]. The first neurocomputer was created in 1950 by Marvin Minsky and Dean Edmonds [4]. We can consider 1956 as the year of artificial intelligence's birth, when McCarthy, Minsky, Rochester, and Shavron, AI pioneers conducted a summer symposium on AI at Dartmouth [5], [6].

Many physical and biological problems include the perturbed Volterra integro-differential and integral equation (see, e.g., [7], [8], [9]). In [10] , the authors provide a survey of singularly perturbed Volterra integral and integro-differential equations. When the perturbed parameter approaches zero for singly perturbed situations, the diameter of the boundary layer narrows.

Nonlinear FSPVIDE plays an important research material in science and engineering. In a fuzzy framework, these equations provide a usual system to simulate the doubt of energetic structures in several logical domains such as physics, geography, medicine, biology, applied mathematics, biophysics, quantum physics, medicine, bio-informatics, and gravity. It is frequently difficult to discover analytic solutions to these problems. Several mathematicians have investigated the numerical solutions to fuzzy equations in recent years [11], [12], [13], [14], [15], [16].

Scientists and engineers have recently conducted extensive research on the Adomian decomposition method (ADM) that is used to solve nonlinear differential and integral problems [11]. Adomian [17] invented the ADM for solving several types of functional equations which is the theme of considerable approximation and analytical study.

In this paper, we consider the numerical discretization of fuzzy singularly perturbed Volterra integro-differential equations (FSPVIDE) in one and two dimensions, respectively as follows:

$$\varepsilon\Psi''(x) = F(\Psi, \Psi', x) + \int_0^x K(x,t)\Psi(t)dt \quad , \quad x \in [a,b]$$

$$\varepsilon\Psi'_x(x,y) = F(x,y,\varepsilon,\Psi(x,y) + \int_0^x \int_0^y K(x,y,t,s)\Psi(t,s)dtds \, , \quad x,y \in [0,X] * [0,Y] \, .$$

With a fuzzy boundary condition, ε is called the perturbation parameter such that $0 < \varepsilon \ll 1$, F and K are given smooth functions on [0, X], $[0, y]$.

The purpose of this paper is to design a neural system to solve a type of perturbation problem in the integral equations numerically (FSPVIDE) and to compare it with the exact solutions.

The paper is structured as follows. In section 2, basic concepts of fuzzy sets are given and we explain some important definitions. In section 3, a fuzzy integro-differential equation is discussed, we also explain the general form of the first order fuzzy Volterra integro-differential equation. Fuzzy Neural Network is given in section 4. In section 5, we describe the fuzzy neural network architecture that is used in this paper through the structure of FNN. In section 6, we solve the Fuzzy Singularly Perturbed Volterra integro-differential equation.

An illustration of the method is done in section 8. In section 2, Numerical Results are presented to illustrate the numerical results of the method by taking two examples. Finally, we show the most important results of the paper through conclusions in section 9.

2. Basic concepts of fuzzy sets:

In this section, we provide several fundamental concepts related to fuzzy theory.

Definition 1 [16]: If U is a collection of substances, then a fuzzy set $\mathcal{A}$in U is a set of ordered pairs:

$\mathcal{A} = \{(U, \xi_{\mathcal{A}}(u)) : u \in U, 0 \leq \xi_{\mathcal{A}}(u) \leq 1\}$.

$\xi_{\mathcal{A}}(u)$ is known as the membership function or grade of membership of u in $\mathcal{A}$, it is also known as the degree of compatibility or degree of truth, which maps U to the membership space $\xi_{\mathcal{A}}(u)$. A is nonfuzzy when $\xi_{\mathcal{A}}(u)$ contains only the two points 0 and 1 and $\xi_{\mathcal{A}}(u)$ is the same as a nonfuzzy set's characteristic function.

Definition 2 [17] : The support of a fuzzy set $\mathcal{A}$, S($\mathcal{A}$) is the crisp set of all $u \in U$ as well as $\xi_{\mathcal{A}}(u) > 0$ and denoted S($\mathcal{A}$).

Definition 3 [18]: The crisp set of elements from the fuzzy set $\mathcal{A}$ at least to the degree τ is called the τ-level set:

$\mathcal{A}_\tau = \{u \in U : \xi_{\mathcal{A}}(u) \geq \tau\}$.

$\mathcal{A}_\tau^{`} = \{u \in U : \xi_{\mathcal{A}}(u) > \tau\}$ is called the strong τ-level set or the strong τ-cut.

Definition 4 [18] **:** A fuzzy set $\mathcal{A}$ is convex if

$\mathcal{A}(\lambda u_1 + (1-\lambda)u_2) \geq min\{\xi_{\mathcal{A}}(u_1), \xi_{\mathcal{A}}(u_2)\}, u_1, u_2 \in U, \lambda \in [0,1]$.

Alternatively, a fuzzy set is convex if all its τ -level sets are convex.

Definition 5 [19]**:** A fuzzy number κ is entirely determined by an ordered pair of functions ($\kappa^l(\tau)$,$\kappa^u(\tau)$), $0 \leq \tau \leq 1$, which satisfies the following:

1) $\kappa^l(\tau)$ is a bounded left continuous non-decreasing function on [0,1].

2) $\kappa^u(\tau)$ is a bounded left continuous function that does not increase. [0,1].

3) $\kappa^l(\tau) \leq \kappa^u(\tau)$, $0 \leq \tau \leq 1$. The crispy number a is just signified by: $\kappa^l(\tau) = \kappa^u(\tau) = a$, $0 \leq \tau \leq 1$. The set of all the fuzzy numbers is denoted by E^1 .

Remark (1), [19]:

For subjective $= (\kappa^l, \kappa^u)$, $\upsilon = (\upsilon^l, \upsilon^u)$ and $C \in$ R, the addition and multiplication by C can be defined as follows:

1) $(\kappa + \upsilon)^l(\tau) = \kappa^l(\tau) + \upsilon^l(\tau)$.

2) $(\kappa + \upsilon)^u(\tau) = \kappa^u(\tau) + \upsilon^u(\tau)$.

3) $(C\kappa)^l(\tau) = C\kappa^l(\tau)$, $(C\kappa)^u(\tau) = C\kappa^u(\tau)$, if C ≥ 0 for all $\tau \in [0,1]$.

4) $(C\kappa)^l(\tau) = C\kappa^u(\tau)$, $(C\kappa)^u(\tau) = C\kappa^l(\tau)$, if C < 0 for all $\tau \in [0,1]$.

Remark (2), [20]:

The distance function between arbitrary two fuzzy numbers $\kappa = (\kappa^l, \kappa^u)$ and $\upsilon = (\upsilon^l, \upsilon^u)$ is given as follows:

$$D(\kappa, \upsilon) = \left[\int_0^1 (\kappa^l(\tau) - \upsilon^l(\tau))^2 d\tau + \int_0^1 \kappa^u(\tau) - \upsilon^u(\tau) d\tau\right]^{\frac{1}{2}}.$$

Definition 6, [19] : The function $\varphi: R \longrightarrow E^1$ is called a fuzzy function. Also, we call every function that is defined from $\mathcal{A}_1 \subseteq E^1$ into $\mathcal{A}_2 \subseteq E^1$ by a fuzzy function.

Definition 7, [19]**:** The fuzzy function φ: R $\longrightarrow$ E^1 is said to be continuous if: For an arbitrary $u_1 \in$ R and $\epsilon > 0$ there exists an $\alpha > 0$ such that: |u - u_1| < $\alpha \Rightarrow D$ (φ (u), $\varphi(u_1)$) < ϵ, where D is the distance function between two fuzzy numbers.

Definition 8, [19]**:** Let I be the real interval. The τ-level set of the fuzzy function Ψ: I $\rightarrow E_1$ can be denoted by: $[\Psi(u)]^{\tau} = [\Psi_1^{\tau}(u), \Psi_2^{\tau}(u)]$, $x \in I$, $\tau \in [0,1]$
The Seikkala derivative $\Psi'(u)$ of the fuzzy function $\Psi(x)$ is defined by: $[\Psi'(u)]^{\tau}=[(\Psi_1^{\tau})'(u), (\Psi_2^{\tau})'(u)]$ $x \in$ I, $\tau \in [0,1]$

3.Fuzzy integro-differential equations

The first order fuzzy Volterra integro-differential equation is given by

$$\varepsilon\Psi'(x) = F(\Psi, x) + \int_0^x K(x,t)\Psi(t)dt \quad ,$$

with boundary conditions $\Psi(a) = A$, $\Psi(b) = B$,where A and B are fuzzy number in E^1 , K is the kernel function over the rectangular region$x, t \in [a, b]$ and F is a specified function of x . If Ψ is a fuzzy function, $F(x)$ is a given fuzzy function and Ψ' is the fuzzy derivative of Ψ, this equation may only possess a fuzzy solution. Let $\Psi(x) = [[\Psi^l(\mathrm{x},\tau)], [\Psi^u(\mathrm{x},\tau)]]$ be a fuzzy solution of the first order of FSPVIDE , therefore, by Definition 5 and Definition8 we have the equivalent system

$\varepsilon[\Psi'(x)]_{\tau}^{l} = [F(\Psi, x)]_{\tau}^{l}, + \int_0^x [K(x,t)\Psi(t)]_{\tau}^{l}, dt \quad , [\Psi']_{\tau}^{l}(\boldsymbol{a}) = [A]_{\tau}^{l}, [\Psi']_{\tau}^{u}(\boldsymbol{a}) = [A]_{\tau}^{u},$

$\varepsilon[\Psi'(x)]_{\tau}^{u} = [F(\Psi, x)]_{\tau}^{u} + \int_0^x [K(x,t)\Psi(t)]_{\tau}^{u} dt \quad , [\Psi']_{\tau}^{l}(\boldsymbol{b}) = [B]_{\tau}^{l}, [\Psi']_{\tau}^{u}(\boldsymbol{b}) = [B]_{\tau}^{u},$

for $\tau \in [0,1]$. Suppose $K(x,t)$ is continuous in $[0,1]$ and changes its sign in finite points for fix *t*.

4.Fuzzy Neural Network(FNN)** [21]

An FNN is a learning machine which is also known as a neuro-fuzzy system that describes the parameters of a fuzzy system by using neural network approximation techniques.
There are some things in common between the neural networks and neuro-fuzzy systems. They are used to solve a problem, for example, pattern recognition, regression, or density estimation, if there is no mathematical model of the problem. They only have specific disadvantages and advantages that are almost entirely eliminated by accomplishing both notions.

Neural networks can only be used if the problem is stated by a large enough number of observed examples. These observations are used to train the black box if there is no required prior knowledge of the issue. On the other hand, it is difficult to extract comprehensible rules from the structure of the neural network.

A fuzzy system, on the other hand, requires language rules rather than acquiring examples as prior knowledge. In addition, the input and output variables must be linguistically characterized. If the knowledge is partial, incorrect, or contradicting, the fuzzy system must be tweaked. Because there is no formal approach, tweaking is done heuristically. This is frequently time-consuming and error-prone.

If all of the variable parameters (weights and biases) are fuzzy numbers, the FNN is said to be completely FNN; otherwise, it is said to be partially FNN. Figure 1 depicts FNN with an input layer, a single hidden layer, and an output layer as a basic structural architecture for the sake of this study.

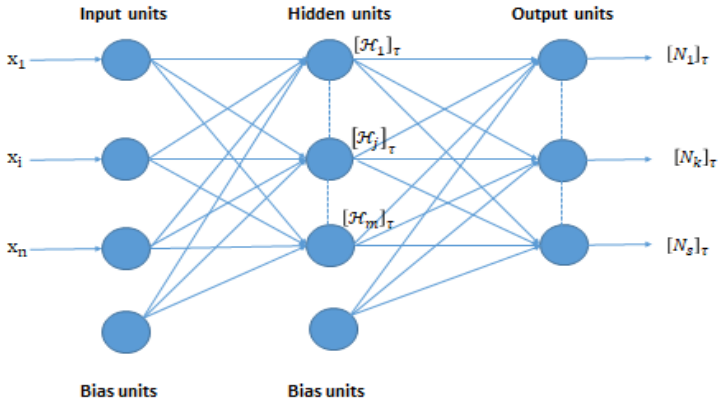

Figure 1: The FNN

5. Structure of FNN

To explain the structure of FNNs for a design consisting of three layers with *n* of input units, *m* of hidden units, and *s* of output boards. Thus, the dimension of the fuzzy neural network is (*n x m x s*). Target vector, connection weights and biases are fuzzy numbers and the input vector is a real number.

The model's architecture demonstrates how FNN changes the n inputs $(x_1, x_2, \dots x_i, \dots x_n)$ into $([O_{ut_1}]_\tau, [O_{ut_2}]_\tau, \dots, [O_{ut_k}]_\tau, \dots, [O_{ut_s}]_\tau)$ which outputs are fuzzy throughout the *m* neurons $([H_{id_1}]_\tau, [H_{id_2}]_\tau, \dots, [H_{id_j}]_\tau, \dots, [H_{id_m}]_\tau)$ which outputs are fuzzy such that $[\mathscr{b}_j]_\tau$ and $[\mathscr{g}_k]_\tau$ are the fuzzy biases for the $[H_{id_j}]_\tau$, $[O_{ut_k}]_\tau$ respectively, $[\mathscr{w}_{ji}]_\tau$ are the fuzzy weight connecting crisp neuron x_i to fuzzy neuron $[H_{id_j}]_\tau$, and $[\mathbb{v}_{kj}]_\tau$ are the fuzzy weight connecting $[H_{id_j}]_\tau$ to $[O_{ut_k}]_\tau$. And Θ the hyperbolic tangent transfer function: $\Theta(x) = \tanh(x)$, and $\Theta'(x) = 1 - \Theta^2(x)$.

Input unit

$x = x_i, \quad i = 1,2,3, \dots, n$

Hidden unit:

$$[H_{id_j}]_\tau = \left[[H_{id_j}]_\tau^l, [H_{id_j}]_\tau^u\right] = \Theta\left([ne\mathcal{T}_j]_\tau\right) = \left[\left(\Theta\left[N_{et_j}\right]_\tau^l\right), \Theta\left(\left[N_{et_j}\right]_\tau^u\right)\right], j = 1,2,..,m$$

$$\left[N_{et_j}\right]_\tau^l = \sum_{i=1}^n x_i [\mathscr{w}_{ji}]_\tau^l + [\mathscr{b}_j]_\tau^l$$

$$\left[N_{et_j}\right]_\tau^u = \sum_{i=1}^n x_i [\mathscr{w}_{ji}]_\tau^u + [\mathscr{b}_j]_\tau^u$$

Output unit :

$$[O_{ut_k}]_\tau = \left[[O_{ut_k}]_\tau^l, [O_{ut_k}]_\tau^u\right] = \quad \Theta([ne\mathcal{T}_k]_\tau) = \left[\left(\Theta[N_{et_k}]_\tau^l\right), \Theta\left([N_{et_k}]_\tau^u\right)\right], \; k = 1,2,3, \dots, s$$

Where

$$[N_{et_k}]_\tau^l = \left(\sum_{j\in a} [\mathbb{v}_{kj}]_\tau^l [H_{id_j}]_\tau^l + \sum_{j\in b} [\mathbb{v}_{kj}]_\tau^l [H_{id_j}]_\tau^u\right) + [\mathscr{g}_k]_\tau^l$$

$$[N_{et_k}]_\tau^u = \left(\sum_{j\in c}[\mathbb{v}_{kj}]_\tau^u [H_{id_j}]_\tau^u + \sum_{j\in d}[\mathbb{v}_{kj}]_\tau^u [H_{id_j}]_\tau^l\right) + [\mathcal{g}_k]_\tau^u$$

$$for \quad [H_{id_j}]_\tau^u \geq [H_{id_j}]_\tau^l \geq 0 \qquad \text{,where}$$

$$a = \{j : [\mathbb{v}_{kj}]_\tau^l \geq 0\}, b = \{j : [\mathbb{v}_{kj}]_\tau^l < 0\}\ ,\ c = \{j : [\mathbb{v}_{kj}]_\tau^u \geq 0\},\ d = \{j : [\mathbb{v}_{kj}]_\tau^u < 0\}$$

6.Solving Fuzzy Singularly Perturbed Volterra integro-differential equation.

6.1. Solving FSPVIDE(One dimension).

To solve any FSPVIDE for one dimension with boundary condition (BC) by FNN, we reflect a three-layered neural networks with one unit entry *x*, one hidden layer with *m* hidden units (neurons), and one linear output unit, resulting in an FNN dimension of$(1 \times m \times 1)$. As it is shown in Figure 2. In this paper, we take the number of hidden layer neurons as five, in other words, *m*=5. The input - output each unit of our neural network can be rewritten for $\tau -$ level set as follows :

Input unit :

$x=x$

Hidden unit:

$$[H_{id_j}]_\tau = \left[[H_{id_j}]_\tau^l, [H_{id_j}]_\tau^u\right] = \left[\left(\Theta\left[N_{et_j}\right]_\tau^l\right), \Theta\left(\left[N_{et_j}\right]_\tau^u\right)\right]$$

Where

$$\left[N_{et_j}\right]_\tau^l = \mathrm{x}[w_j]_\tau^l + [\mathcal{b}_j]_\tau^l\ , \quad \left[N_{et_j}\right]_\tau^u = \mathrm{x}[w_j]_\tau^u + [\mathcal{b}_j]_\tau^u$$

Output unit :

$$[O_{ut}]_\tau = [[O_{ut}]_\tau^l, [O_{ut}]_\tau^u]$$

Where

$$[O_{ut}]_\tau^l = \left(\sum_{j\in a}[\mathbb{v}_j]_\tau^l [H_{id}\mathcal{H}_j]_\tau^l + \sum_{j\in b}[\mathbb{v}_j]_\tau^l [H_{id_j}]_\tau^u\right)$$

$$[O_{ut}]_\tau^u = \left(\sum_{j\in c}[\mathbb{v}_j]_\tau^u [H_{id_j}]_\tau^u + \sum_{j\in d}[\mathbb{v}_j]_\tau^u [H_{id_j}]_\tau^l\right)$$

$$for \quad [H_{id_j}]_\tau^u \geq [H_{id_j}]_\tau^l \geq 0 \qquad \text{,where}$$

$$a = \{j : [\mathbb{v}_j]_\tau^l \geq 0\}\ ,\ b = \{j : [\mathbb{v}_j]_\tau^l < 0\}\ ,\ c = \{j : [\mathbb{v}_j]_\tau^u \geq 0\}\ ,\ d = \{j : [\mathbb{v}_j]_\tau^u < 0\}$$

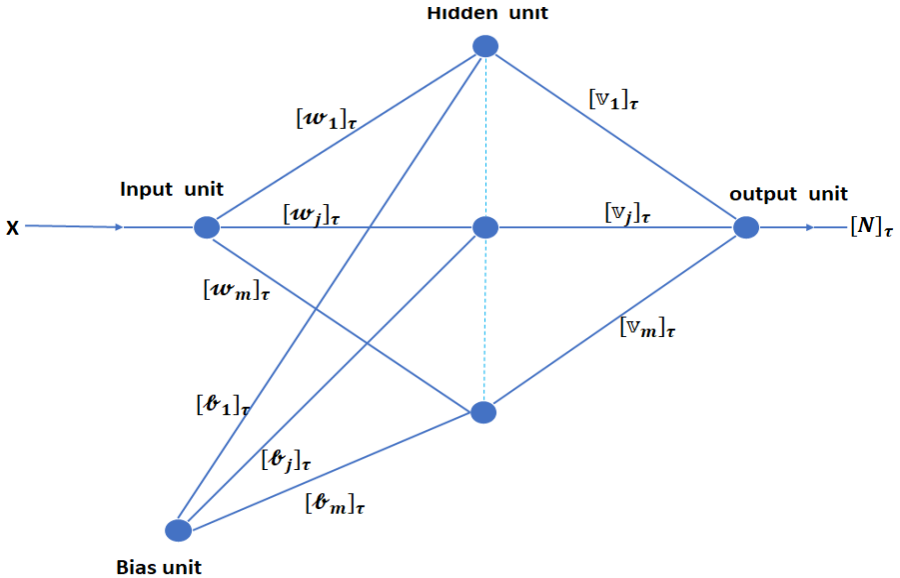

Figure 2: The (1×m×1)Feed forward FNN

6.2. Solution of two-dimensional FSVIDE

To solve two dimensions of FSPVIDE using FNN, we reflect the 3 layers of the neural network with two unit entries x and y, one hidden layer consisting of m unit and one unit output $O_{ut}(x,y)$. Here, the dimension of FFNN (2×m×1) is shown in Figure 3. In this paper, we take the number of hidden layer neurons as seven, in other words, m=7. Also, the input - output each unit of our neural network can be rewritten for $\tau-$ level set as follows:

Input unit :

x=x ,y=y

Hidden unit :

$$\left[H_{id_j}\right]_\tau = \left[\left[H_{id_j}\right]_\tau^l, \left[H_{id_j}\right]_\tau^u\right] = \left[\Theta\left(\left[N_{et_j}\right]_\tau^l\right), \Theta\left(\left[N_{et_j}\right]_\tau^u\right)\right]$$

$$\left[N_{et_j}\right]_\tau^l = x\left[w_{j1}\right]_\tau^l + y\left[w_{j2}\right]_\tau^l + \left[\theta_j\right]_\tau^l$$

$$\left[N_{et_j}\right]_\tau^u = x\left[w_{j1}\right]_\tau^u + y\left[w_{j2}\right]_\tau^u + \left[\theta_j\right]_\tau^u$$

Output unit :

$$[O_{ut}]_\tau = [[O_{ut}]_\tau^l, [O_{ut}]_\tau^u]$$

$$[O_{ut}]_\tau^l = \sum_{j\in a}^m \left[\mathbb{v}_j\right]_\tau^l \left[H_{id_j}\right]_\tau^l + \sum_{j\in b}^m \left[\mathbb{v}_j\right]_\tau^l \left[H_{id_j}\right]_\tau^u$$

$$[O_{ut}]_\tau^u = \sum_{j\in c}^m \left[\mathbb{v}_j\right]_\tau^u [H_{id}]_\tau^u + \sum_{j\in d}^m \left[\mathbb{v}_j\right]_\tau^u \left[H_{id_j}\right]_\tau^l$$

$$for\ \left[H_{id_j}\right]_\tau^u \geq \left[H_{id_j}\right]_\tau^l \geq 0\ , where\ :$$

$$a = \left\{j : \left[\mathbb{v}_j\right]_\tau^l \geq 0\right\}\ ,\ b = \left\{j : \left[\mathbb{v}_j\right]_\tau^l < 0\right\}\ ,\ c = \left\{j : \left[\mathbb{v}_j\right]_\tau^u \geq 0\right\}\ ,\ d = \left\{j : \left[\mathbb{v}_j\right]_\tau^u < 0\right\}$$

$$a \cup b = \{1,2,3,\ldots m\}\ and\ \ c \cup d = \{1,2,3,\ldots m\}$$

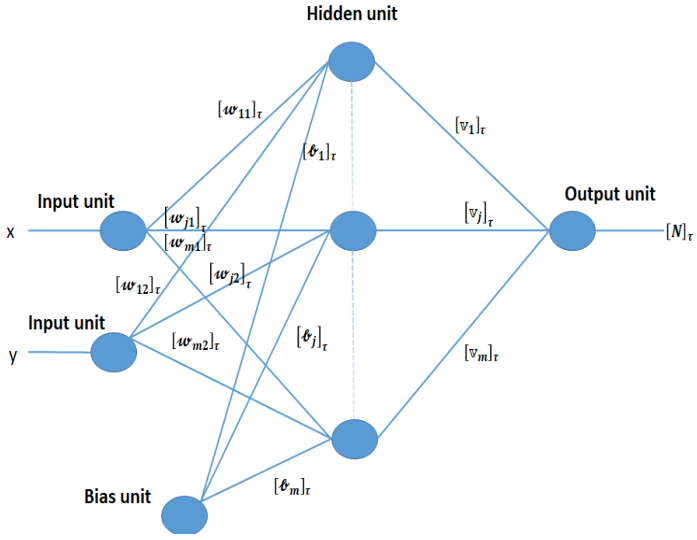

Figure 3: (2×m×1)Feed forward FFNN

7. Illustration of the method

Consider the one-dimensional FSPVIDE:

$$\varepsilon\Psi''(x) = F(\Psi,\Psi',x) + \int_0^x K(x,t)\Psi(t)dt \quad , \quad x \in [a,b] \qquad (1)$$

with boundary condition $\Psi(a) = A\ , \Psi(b) = B$, where A and B are fuzzy number in E^1 with $\tau-$ level sets $[A]_\tau = [[A]_\tau^l, [A]_\tau^u]\ and\ [B]_\tau = [[B]_\tau^l, [B]_\tau^u]$ and ε is a small parameter

satisfying $0 \ll \epsilon < 1$ called perturbation parameter, F and K are given smooth functions on $[0, X]$.

For this problem, the fuzzy trial solution can be written as follows:

$$[\Psi_t(\mathrm{x},p)]_\tau = \frac{b[A]_\tau - a[B]_\tau}{b-a} + \frac{[B]_\tau - [A]_\tau}{b-a}\mathrm{x} + (\mathrm{x}-a)(\mathrm{x}-b)[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau \quad (2)$$

where $O_{ut}(\mathrm{x},\mathrm{p},\varepsilon)$ is the output of feed forward the problem is transformed into a discretized from

$$min \sum_{x_i \in D} F(\Psi, \Psi', x_i, \varepsilon) + \int_0^{x_i} K(x_i, t, \varepsilon)\Psi(t)dt \quad , \quad x \in [a,b]$$

The amount of error that must be minimized is given as follows:

$$\mathbb{E}(p) = \sum_{i=1}^{g}(\mathbb{E}_{i\tau}^{l}(p) + \mathbb{E}_{i\tau}^{u}(p)) \quad (3)$$

Where

$$\mathbb{E}_{i\tau}^{l}(p,\varepsilon) = \left[\varepsilon\left[\frac{d^2\Psi_t(\mathrm{x}_i,p)}{dx^2}\right]_\tau^l - \left[\sum_{x_i\in D} F(\Psi_t(x_i), \Psi_t{}'(x_i), x_i, \varepsilon) + \int_0^{x_i} K(x_i,t,\varepsilon)\Psi_t(t)dt\right]_\tau^l\right]^2 \quad (4)$$

$$\mathbb{E}_{i\tau}^{u}(p,\varepsilon) = \left[\varepsilon\left[\frac{d^2\Psi_t(\mathrm{x}_i,p)}{dx^2}\right]_\tau^u - \left[\sum_{x_i\in D} F(\Psi_t(x_i), \Psi_t{}'(x_i), x_i, \varepsilon) + \int_0^{x_i} K(x_i,t,\varepsilon)\Psi_t(t)dt\right]_\tau^u\right]^2 \quad (5)$$

where $\{\mathrm{x}_i\}_{i=1}^{g}$ are distinct points [a,b], $\mathbb{E}_{i\tau}^{l}$ and $\mathbb{E}_{i\tau}^{u}$ are the least squared errors for the τ - level sets' lower and upper boundaries, respectively.

To drive the minimized error function of problem (1):

From (2), we can find :

$$[\Psi_t(\mathrm{x},p)]_\tau^l = \frac{b[A]_\tau^l - a[B]_\tau^l}{b-a} + \frac{[B]_\tau^l - [A]_\tau^l}{b-a}\mathrm{x} + (\mathrm{x}-a)(\mathrm{x}-b)[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^l \ . \quad (6)$$

$$[\Psi_t(\mathrm{x},p)]_\tau^u = \frac{b[A]_\tau^u - a[B]_\tau^l}{b-a} + \frac{[B]_\tau^u - [A]_\tau^u}{b-a}\mathrm{x} + (\mathrm{x}-a)(\mathrm{x}-b)[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^u. \quad (7)$$

Then we get :

$$\frac{d[\Psi_t(\mathrm{x},p)]_\tau^l}{d\mathrm{x}} = \frac{[B]_\tau^l - [A]_\tau^l}{b-a} + (\mathrm{x}-a)(\mathrm{x}-b)\frac{d[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^l}{d\mathrm{x}} + \big(2\mathrm{x}-(a+b)\big)[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^l. \quad (8)$$

$$\frac{d[\Psi_t(\mathrm{x},p)]_\tau^u}{d\mathrm{x}} = \frac{[B]_\tau^u - [A]_\tau^u}{b-a} + (\mathrm{x}-\mathrm{a})(\mathrm{x}-\mathrm{b})\frac{d[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^u}{d\mathrm{x}} + \big(2\mathrm{x}-(a+b)\big)[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^u. \quad (9)$$

We have :

$$\left[\frac{d^2\Psi_t(\mathrm{x})}{d\mathrm{x}^2}\right]_\tau^l = (\mathrm{x}-\mathrm{a})(\mathrm{x}-b)\frac{d^2[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^l}{d\mathrm{x}^2} + 2\big(2\mathrm{x}-(a+b)\big)\frac{d[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^l}{d\mathrm{x}} + 2[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^l. \quad (10)$$

$$\left[\frac{d^2\Psi_t(\mathrm{x})}{d\mathrm{x}^2}\right]_\tau^u = (\mathrm{x}-\mathrm{a})(\mathrm{x}-\mathrm{b})\frac{d^2[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^u}{d\mathrm{x}^2} + 2(2\mathrm{x}-(a+b))\frac{d[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^u}{d\mathrm{x}} + 2[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau^u \ . \quad (11)$$

Then(4) and (5)can be rewritten as :

$$\mathbb{E}_{i\tau}^{l}(p,\varepsilon) =$$

$$\Big[(\mathrm{x}_i-\mathrm{a})(\mathrm{x}_i-\mathrm{b})\frac{d^2[O_{ut}(\mathrm{x}_i,p,\varepsilon)]_\tau^l}{d\mathrm{x}^2} + 2\big(2\mathrm{x}_i-(a+b)\big)\frac{d[N(\mathrm{x}_i,p,\varepsilon)]_\tau^l}{d\mathrm{x}} + 2[O_{ut}(\mathrm{x}_i,p,\varepsilon)]_\tau^l -$$

$$\frac{1}{\varepsilon}F\Big(\mathrm{x}_i, \frac{b[A]_\tau^l - a[B]_\tau^l}{b-a} + \frac{[B]_\tau^l - [A]_\tau^l}{b-a}\mathrm{x}_i + (\mathrm{x}_i-a)(\mathrm{x}_i-b)[O_{ut}(\mathrm{x}_i,p,\varepsilon)]_\tau^l, \frac{[B]_\tau^l - [A]_\tau^l}{b-a} + (\mathrm{x}_i-a)(\mathrm{x}_i-$$

$$b)\frac{d[O_{ut}(\mathrm{x}_i,p,\varepsilon)]_\tau^l}{d\mathrm{x}} + \big(2\mathrm{x}_i-(a+b)\big)[O_{ut}(\mathrm{x}_i,p,\varepsilon)]_\tau^l + \int_0^{x_i} K(x_i,t,\varepsilon)\Psi_t(t)dt\Big)\Big]^2. \quad (12)$$

$$\mathbb{E}_{i\tau}^{u}(p,\varepsilon) = \Big[(x_i-a)(x_i-b)\frac{d^2[O_{ut}(x_i,p,\varepsilon)]_\tau^u}{dx^2} + 2\big(2x_i-(a+b)\big)\frac{d[O_{ut}(x_i,p,\varepsilon)]_\tau^u}{dx} +$$

$$2[O_{ut}(x_i,p,\varepsilon)]_\tau^u - \frac{1}{\varepsilon}F(x_i, \frac{b[A]_\tau^u - a[B]_\tau^u}{b-a} + \frac{[B]_\tau^u - [A]_\tau^u}{b-a}x_i +$$

$$(x_i-a)(x_i-b)[O_{ut}(x_i,p,\varepsilon)]_\tau^u\,,\frac{[B]_\tau^u-[A]_\tau^u}{b-a}+(x_i-a)(x_i-b)\frac{d[O_{ut}(x_i,p,\varepsilon)]_\tau^u}{dx}+\big(2x_i-(a+b)\big)[O_{ut}(x_i,p,\varepsilon)]_\tau^u+\int_0^{x_i}K(x_i,t,\varepsilon)\Psi_t(t)dt\,\Big)\Big]^2 \quad (13)$$

To illustrate the solution to the two-dimensional FSPVIDE, we consider the first order FSPVIDE

$$\varepsilon\Psi'_x(x,y)=F(x,y,\varepsilon,\Psi(\mathrm{x},\mathrm{y})+\int_0^x\int_0^y K(x,y,t,s)\Psi(t,s)dtds\,,\ x,y\in[0,x].[0,y]\,. \quad (14)$$

where ε is a small parameter that satisfies $0<\epsilon\ll 1$ and called perturbation parameter, *F* and *K* are given smooth functions on [0, *X*]. [0, *Y*], they are given by the Dirichiet fuzzy BCs for *x,y* ∈ [0,1]

$\Psi(0,y)=\mathcal{F}_0(y),\Psi(1,y)=\mathcal{F}_1(y),\Psi(\mathrm{x},0)=\mathcal{G}_0(\mathrm{x})\ \ and\ \ \Psi(\mathrm{x},1)=\mathcal{G}_1(\mathrm{x})$

$\mathcal{F}_0(y)\,,\mathcal{F}_1(y)\,,\mathcal{G}_0(\mathrm{x})$ and $\mathcal{G}_1(\mathrm{x})$ are fuzzy numbers or fuzzy functions with τ −level set:

$[\mathcal{F}_0(y)]_\tau=[\mathcal{F}_0^l(y),\mathcal{F}_0^u(y)]\,,[\mathcal{F}_1(y)]_\tau=[\mathcal{F}_1^l(y),\mathcal{F}_1^u(y)]$

$[\mathcal{G}_0(\mathrm{x})]_\tau=[\mathcal{G}_0^l(\mathrm{x}),\mathcal{G}_0^u(\mathrm{x})]\,,[\mathcal{G}_1(\mathrm{x})]_\tau=[\mathcal{G}_1^l(\mathrm{x}),\mathcal{G}_1^u(\mathrm{x})]$

The fuzzy trial solution

$$[\Psi_t(\mathrm{x},y)]_\tau=[[\Psi_t^l(\mathrm{x},y)]_\tau,[\Psi_t^u(\mathrm{x},y)]_\tau] \quad (15)$$

$$[\Psi_t(\mathrm{x},y)]_\tau^l=[A(\mathrm{x},y)]_\tau^l+\mathrm{x}y(1-\mathrm{x})(1-y)[O_{ut}(\mathrm{x},y,\varepsilon)]_\tau^l \quad (16)$$

$$[\Psi_t(\mathrm{x},y)]_\tau^u=[A(\mathrm{x},y)]_\tau^u+\mathrm{x}y(1-\mathrm{x})(1-y)[O_{ut}(\mathrm{x},y,\varepsilon)]_\tau^u \quad (17)$$

where$[A(\mathrm{x},y)]_\tau^l$ and$[A(\mathrm{x},y)]_\tau^u$ are chosen so as to satisfy the following fuzzy BCs

$$[A(\mathrm{x},y)]_\tau^l=(1-\mathrm{x})\mathcal{F}_0^l(y)+\mathrm{x}\mathcal{F}_1^l(y)+(1-y)\Big[\mathcal{G}_0^l(\mathrm{x})-[(1-\mathrm{x})\mathcal{G}_0^l(0)+\mathrm{x}\mathcal{G}_0^l(1)]\Big]+y\Big[\mathcal{G}_1^l(\mathrm{x})-[(1-\mathrm{x})\mathcal{G}_1^l(0)+\mathrm{x}\mathcal{G}_1^l(1)]\Big] \quad (18)$$

$$[A(\mathrm{x},y)]_\tau^u=(1-\mathrm{x})\mathcal{F}_0^u(y)+\mathrm{x}\mathcal{F}_1^u(y)+(1-y)\big[\mathcal{G}_0^u(\mathrm{x})-[(1-\mathrm{x})\mathcal{G}_0^u(0)+\mathrm{x}\mathcal{G}_0^u(1)]\big]+y\big[\mathcal{G}_1^u(\mathrm{x})-[(1-\mathrm{x})\mathcal{G}_1^u(0)+\mathrm{x}\mathcal{G}_1^u(1)]\big] \quad (19)$$

The minimized error function will be :

$$\mathbb{E}(p)=\sum_{i=1}^{g}(\mathbb{E}_{i\tau}^l(p)+\mathbb{E}_{i\tau}^u(p)),\ p \text{ is parimeter} \quad (20)$$

$$\mathbb{E}_{i\tau}^l(p,\varepsilon)=\left[\varepsilon\left[\frac{d\Psi_t(\mathrm{x}_i,\mathrm{y}_ip)}{d\mathrm{x}}\right]_\tau^l-\left[\sum_{x_{i,y_i}\in D}F(\Psi_t(x_i,\mathrm{y}_i),x_i,\mathrm{y}_i\varepsilon)+\int_0^{x_i}\int_0^{y_i}K(x_i,\mathrm{y}_i,t,\varepsilon)\Psi_t(t,s)dtds\,\right]_\tau^l\right]^2 \quad (21)$$

$$\mathbb{E}_{i\tau}^u(p,\varepsilon)=\left[\varepsilon\left[\frac{d\Psi_t(\mathrm{x}_i,\mathrm{y}_ip)}{d\mathrm{x}}\right]_\tau^u-\left[\sum_{x_{i,y_i}\in D}F(\Psi_t(x_i,\mathrm{y}_i),x_i,\mathrm{y}_i\varepsilon)+\int_0^{x_i}\int_0^{y_i}K(x_i,\mathrm{y}_i,t,\varepsilon)\Psi_t(t,s)dtds\,\right]_\tau^u\right]^2 \quad (22)$$

8. Numerical Results

In this section, we give illustrations of programs built in MATLAB 7.11 that exemplify the behavior and characteristics of the suggested design.

An FNN is employed with one input unit for solving one dimensional FSPVIDE and two input units for solving two dimensions of the FSPVIDE, one hidden layer with five units for solving one dimension and seven hidden units for solving two dimensions, and one linear output unit. The hyperbolic tangent activation function is the sigmoid activation function of each buried unit.

To calculate the accuracy of the method, we use the absolute value between the approximate and exact values such that $[\Psi_{\mathrm{a}}(\mathrm{x})]_\tau$ is the analytic solution.

$$[\mathbb{E}(\mathrm{x})]_\tau^l=|[\Psi_t(\mathrm{x})]_\tau^l-[\Psi_a(\mathrm{x})]_\tau^l|\,. \quad (23)$$

$$[\mathbb{E}(\mathrm{x})]_\tau^u=|[\Psi_t(\mathrm{x})]_\tau^u-[\Psi_a(\mathrm{x})]_\tau^u|. \quad (24)$$

We give figures exhibiting the associated error $[\mathbb{E}(x)]_\tau$ together at the little places (training points) that are used for training and at several added points (test points) within the domain of each equation to illustrate the properties of the solution supplied by the suggested design.

Example 1: Consider the FSPVIDE

$\varepsilon\Psi''(x) = C\frac{\varepsilon}{24x}(36-5x^4)+\frac{1}{2}\int_0^x(x^2+s^2)\Psi(s,x)ds$, with fuzzy B.C $\Psi(0) = [0,0], \Psi(1) = [\tau^5+2\tau, 6-3\tau^3]$ and $C = [(\tau^5+2\tau)x^3, (6-3\tau^3)x^3$, $0 \le s \le x \le 1$ $0 < \varepsilon << 1$,$\tau \in [0,1]$.

The exact solution is given as follows [22]:

$[\Psi(x)]_\tau^l = (\tau^5+2\tau)x^3$, $[\Psi(x)]_\tau^u = (6-3\tau^3)x^3$

Then the fuzzy trial solution for this example is :

$$[\Psi_t(\mathrm{x})]_\tau = [\tau^5+2\tau, 6-3\tau^3]\mathrm{x} + \mathrm{x}(\mathrm{x}-1)[O_{ut}(\mathrm{x},p,\varepsilon)]_\tau$$

where $O_{ut}(\mathrm{x},\mathrm{p},\varepsilon)$ the output of the feed forward . From the fuzzy trial solution, we have

$[\Psi_t(\mathrm{x})]_\tau^l = -\tau^5+2\tau\mathrm{x}+\mathrm{x}(\mathrm{x}-1)[O_{ut}(\mathrm{x},p,\varepsilon))]_\tau^l$

$[\Psi_t(\mathrm{x})]_\tau^u = 6-3\tau^3\mathrm{x}+\mathrm{x}(\mathrm{x}-1)[O_{ut}(\mathrm{x},p,\varepsilon))]_\tau^u$

The error function which must be minimized is given as :

$\mathbb{E}(p) = \sum_{i=1}^{g}(\mathbb{E}_{i\tau}^l(p)+\mathbb{E}_{i\tau}^u(p))$

Where

$$\mathbb{E}_{i\tau}^l(p,\varepsilon) = \left[\varepsilon\left[\frac{d^2\Psi_t(\mathrm{x}_i,p)}{dx^2}\right]_\tau^l - \left[\sum_{x_i\in D}(\tau^5+2\tau){x_i}^3\frac{\varepsilon}{12x_i}(36-5{x_i}^4)+\frac{1}{2}\int_0^{x_i}(x^2+s^2)\Psi_t(s,x)ds\right]_\tau^l\right]^2$$

$$\mathbb{E}_{i\tau}^u(p,\varepsilon) = \left[\varepsilon\left[\frac{d^2\Psi_t(\mathrm{x}_i,p)}{dx^2}\right]_\tau^u - \left[\sum_{x_i\in D}(6-3\tau^3){x_i}^3\frac{\varepsilon}{12x}(36-5{x_i}^4)+\frac{1}{2}\int_0^{x_i}(x^2+s^2)\Psi_t(s,x)ds\right]_\tau^u\right]^2$$

The error function that must be minimized for this state is simple to evaluate:

$\mathbb{E} = \sum_{i=1}^{11}(\mathbb{E}_{i\tau}^l+\mathbb{E}_{i\tau}^u)$.

The following tables and figures show the approximate solution and its comparison with the exact solution for a set of training points for the proposed network. The errors in the tables show the accuracy and speed of convergence of the proposed method.

Table 1: Exact and FNN solutions of example 1, $\varepsilon = 10^{-6}$, $\tau = 0.7$

input	Exact solution $\Psi_a(x)$		Solution of FFNN $\Psi_t(x)$	
x	$[\Psi_a(x)]_\tau^\ell$	$[\Psi_a(x)]_\tau^u$	$[\Psi_t(x)]_\tau^\ell$	$[\Psi_t(x)]_\tau^u$
0	0.000000000000	0.000000000000	0.000000000000	**0.000000000000**
0.1	0.001568070000	0.004971000000	0.001568050000	**0.004975430010**
0.2	0.012544560000	0.039768000000	0.012544580000	**0.039763006400**
0.3	0.042337890000	0.134217000000	0.042337679000	**0.134214000460**
0.4	0.100356480000	0.318144000000	0.100356460000	**0.318146000000**
0.5	0.196008750000	0.621375000000	0.196008759654	**0.621339000000**
0.6	0.338703120000	1.073736000000	0.338703129800	**1.073732764900**
0.7	0.537848010000	1.705053000000	0.537848010001	**1.705053344210**
0.8	0.802851840000	2.545152000000	0.802851843400	**2.545165008500**
0.9	1.143123030000	3.623859000000	1.143123038700	**3.623852000000**
1	1.568070000000	4.971000000000	1.568070000000	**4.971000000000**

Table 2: Accurateness of solutions of example 1, $\varepsilon = 10^{-6}, \tau = 0.7$

The error $[\mathbf{E}(\mathbf{x})]_\tau = \lvert[\Psi_a(\boldsymbol{\varkappa})]_\tau - \Psi_t(\boldsymbol{\varkappa})]_\tau\rvert$	
$[\mathbf{E}(\boldsymbol{\varkappa})]_\tau^{\ell}$	$[\mathbf{E}(\boldsymbol{\varkappa})]_\tau^{u}$
0	**0**
2E-08	**4.43001E-06**
2E-08	**4.9936E-06**
2.11E-07	**2.99954E-06**
2E-08	**2E-06**
9.654E-09	**3.6E-05**
9.8E-09	**3.2351E-06**
5.40235E-13	**3.4421E-07**
3.4E-09	**1.30085E-05**
8.7E-09	**7E-06**
0	**0**
MSE=4.56339E-14	**MSE=1.56452E-09**

Table 3: Exact and FNN solution of example 1, $\varepsilon = 10^{-6}$, $x = 0.5$

input	Exact solution $\Psi_a(x)$		Solution of FFNN $\Psi_t(\varkappa)$	
τ	$[\Psi_a(\boldsymbol{\varkappa})]_\tau^{\ell}$	$[\Psi_a(\boldsymbol{\varkappa})]_\tau^{u}$	$[\Psi_t(\boldsymbol{\varkappa})]_\tau^{\ell}$	$[\Psi_t(\boldsymbol{\varkappa})]_\tau^{u}$
0	0.000000000000	0.750000000000	0.000000000000	**0.750000000000**
0.1	0.025001250000	0.749625000000	0.025001244300	**0.749623200000**
0.2	0.050040000000	0.747000000000	0.050906654900	**0.747654000000**
0.3	0.075303750000	0.739875000000	0.075303743000	**0.739873000000**
0.4	0.101280000000	0.726000000000	0.101230000000	**0.726760000000**
0.5	0.128906250000	0.703125000000	0.128906303000	**0.703122000000**
0.6	0.159720000000	0.669000000000	0.159727500000	**0.669400000000**
0.7	0.196008750000	0.621375000000	0.196008320000	**0.621373000000**
0.8	0.240960000000	0.558000000000	0.240970080000	**0.558500000000**
0.9	0.298811250000	0.476625000000	0.298811290000	**0.476624000000**
1	0.375000000000	0.375000000000	0.375000000000	**0.375000000000**

Table 4: Accurateness of solutions of example 3.2, $\varepsilon = 10^{-6}$, $x = 0.5$

The error $[\mathbf{E}(\mathbf{x})]_\tau = \lvert[\Psi_a(\boldsymbol{\varkappa})]_\tau - \Psi_t(\boldsymbol{\varkappa})]_\tau\rvert$	
$[\mathbf{E}(\boldsymbol{\varkappa})]_\tau^{\ell}$	$[\mathbf{E}(\boldsymbol{\varkappa})]_\tau^{u}$
0	**0**
5.7E-09	**1.8E-06**
0.000866655	**0.000654**
7E-09	**2E-06**
5E-05	**0.00076**
5.3E-08	**3E-06**

7.5E-06	**0.0004**
4.3E-07	**2E-06**
1.008E-05	**0.0005**
4E-08	**1E-06**
0	**0**
MSE=7.53749E-07	**MSE=1.41533E-06**

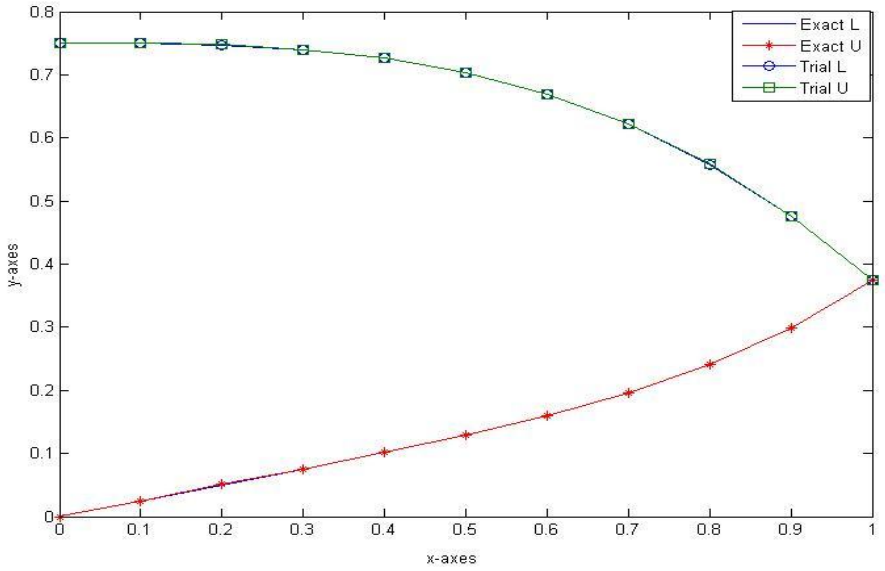

Figure 4: Exact and neural solution of example 1, with ε =10^{-6}, $\tau = 0.7$

Example 2: In this problem, we consider the following FSPVIDE
$\varepsilon\Psi'_x(x,y) - \int_0^x \int_0^y (t+s)\Psi(t,s)dtds = \frac{(y+1)}{\varepsilon} - \frac{x^3y^3}{6} - \frac{x^3y}{3} - \frac{x^2y^3}{6} - \frac{t^2y^2}{2}$, $x, y \in [0,x].[0,y]$, $0 \le s \le x \le 1$ $0 < \varepsilon << 1$,$\tau \in [0,1]$, with the Dirichiet fuzzy BCs

$$[\Psi(0,y)]_\tau = [0,0]_\tau, [\Psi(1,y)]_\tau = [(\tau-1,1-\tau)(y+1)]_\tau, [\Psi(\mathrm{x},0)]_\tau = [(\tau-1,1-\tau)x]_\tau \text{ and } [\Psi(\mathrm{x},1)]_\tau = [(\tau-1,1-\tau)2x]_\tau$$

which has the following exact solution [22]:

$$\Psi(x,y) = [\tau-1,1-\tau](xy+x)$$

The fuzzy trial solution

$$[\Psi_t(\mathrm{x},y)]_\tau = [[\Psi_t^l(\mathrm{x},y)]_\tau, [\Psi_t^u(\mathrm{x},y)]_\tau]$$

Then we get

$$[\boldsymbol{\Psi_t}(\mathbf{x},\boldsymbol{y})]_{\boldsymbol{\tau}}^{\boldsymbol{l}} = (\mathbf{xy}+\boldsymbol{x})(\boldsymbol{\tau}-\mathbf{1}) + (\mathbf{1}-\boldsymbol{y})\big[\boldsymbol{x}(\boldsymbol{\tau}-\mathbf{1}) - [\mathbf{x}(\boldsymbol{\tau}-\mathbf{1})]\big] + \boldsymbol{y}\big[\mathbf{2}\boldsymbol{x}(\boldsymbol{\tau}-\mathbf{1}) + [\mathbf{2}\boldsymbol{x}(\boldsymbol{\tau}-\mathbf{1})]\big] + \mathbf{xy}(\mathbf{1}-\mathbf{x})(\mathbf{1}-\boldsymbol{y})[\boldsymbol{O_{ut}}(\mathbf{x},\boldsymbol{y},\boldsymbol{\varepsilon})]_{\boldsymbol{\tau}}^{\boldsymbol{l}}$$

$$[\Psi_t(\mathrm{x},y)]_\tau^u = (\mathrm{xy}+\mathrm{x})(1-\tau) + (1-y)\big[x(1-\tau) - [x^2(1-\tau))]\big] + y\big[2x(1-\tau) - [\mathrm{x}(1-\tau)]\big] + \mathrm{x}y(1-\mathrm{x})(1-y)[O_{ut}(\mathrm{x},y,\varepsilon)]_\tau^u$$

The minimized error function will be :

$\mathbb{E}(p) = \sum_{i=1}^{g}(\mathbb{E}_{i\tau}^l(p) + \mathbb{E}_{i\tau}^u(p))$, p is parimeter.

The following tables and figures show the approximate solution and its comparison with the analytical solution for a set of training points for the proposed network, and the error tables show the accuracy and speed of convergence of the proposed method.

Table 5: Exact and FNN solution of example 2, $\varepsilon = 10^{-4}$ $\tau = 0.5$

input		Exact solution $\Psi_a(\varkappa)$		Solution of FNN $\Psi_t(\varkappa)$	
x	**y**	$[\Psi_a(\varkappa)]_\tau^\ell$	$[\Psi_a(\varkappa)]_\tau^u$	$[\Psi(\varkappa)]_\tau^\ell$	$[\Psi_t(\varkappa)]_\tau^u$
0	0	0.000000000000	0.000000000000	0.000000000000	**0.000000000000**
0.1	0.1	-0.055000000000	0.075000000000	-0.055221870000	**0.075004328760**
0.2	0.2	-0.120000000000	0.150000000000	-0.120004321000	**0.150006594300**
0.3	0.3	-0.195000000000	0.225000000000	-0.195005432100	**0.225005400970**
0.4	0.4	-0.280000000000	0.300000000000	-0.280000054300	**0.300000003200**
0.5	0.5	-0.375000000000	0.375000000000	-0.375000005410	**0.375000000065**
0.6	0.6	-0.480000000000	0.450000000000	-0.480003219000	**0.450000512900**
0.7	0.7	-0.595000000000	0.525000000000	-0.595000432100	**0.525000000054**
0.8	0.8	-0.720000000000	0.600000000000	-0.720005499860	**0.600000006543**
0.9	0.9	-0.855000000000	0.675000000000	-0.855000432980	**0.675000000000**
1	1	-1.000000000000	0.750000000000	-1.000000000000	**0.750000000000**

Table 6: Accurateness of solutions of example 2, $\varepsilon = 10^{-4}$ $\tau = 0.5$

The error $[\mathbf{E}(\mathbf{x})]_\tau = \lvert[\Psi_a(\varkappa)]_\tau - \Psi(\varkappa)]_\tau\rvert$	
$[\mathbf{E}(\varkappa)]_\tau^\ell$	$[\mathbf{E}(\varkappa)]_\tau^u$
0	**0**
0.00022187	**4.32876E-06**
4.321E-06	**6.5943E-06**
5.4321E-06	**5.40097E-06**
5.43E-08	**3.2E-09**
5.4096E-09	**6.5E-11**
3.219E-06	**5.129E-07**
4.321E-07	**5.40001E-11**
5.49986E-06	**6.543E-09**
4.3298E-07	**0**
0	**0**
MSE= 4.56428E-09	**MSE= 7.46219E-11**

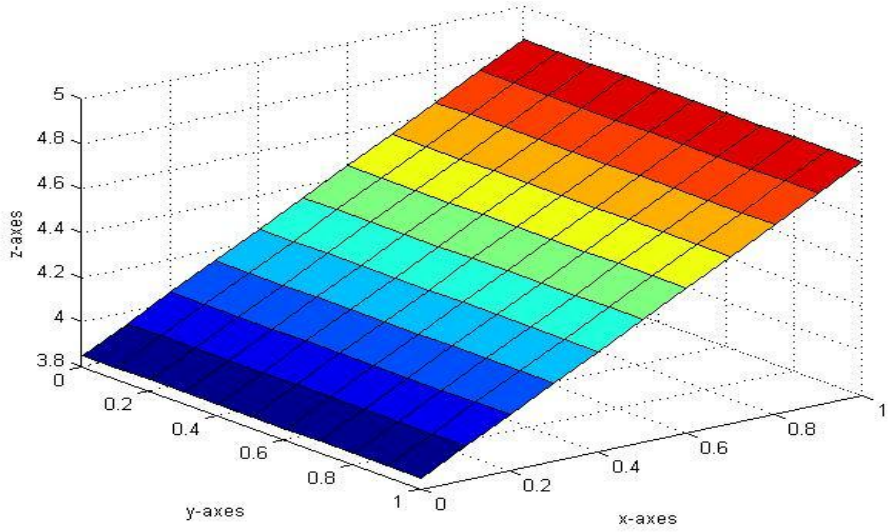

Figure 5: Exact and neural solution of example 2, $\varepsilon = 10^{-4}$ $\tau = 0.5$

9. Conclusions

In this paper, we used a type of AI, which is within deep learning, by designing an artificial neural network to find the approximate solution for certain kinds of fuzzy integrative equations, which are the fuzzy singularly perturbed Volterra integro-differential equations with one and two variables, this type of equation is of great importance in many applications. It is often used in the transmission of radiant energy and ripple or oscillation issues.

In this design, three layers, the input and output layer and the hidden layer are used, in which the number of nodes is either five or seven according to the number of variables with an activation function of type sigmoid which is the hyperbolic tangent activation function. The results of this paper are clear through the application of this technique to some examples with one or two variables through the speed of convergence and the reduction of the error rate after comparing them to the exact solutions.

It is possible that there are some future works for this paper by changing the structure of the neural network or changing the activation function, or it is possible to apply this technique to the type of equations in three dimensions.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," vol. 5, no. *Bull. Math. Biophys*, pp. 115-133, 1943.

[2] D. O. Hebb and D. Hebb, *The Organization of Behavior, 65th ed. Wiley , New York*, 1949.

[3] P. Milner, *A brief history of the hebbian learning rule. Canadian Psychology, Psychologie*, 2003.

[4] M. Mosleh, "Fuzzy neural network for solving a system of fuzzy dfferential equations" *Applied Soft Computing*, vol. 13, pp. 3597-3607, 2013.

[5] A. B. Badiru and J. Cheung, *Fuzzy Engineering Expert Systems with Neural Network Applications, 11th ed. John Wiley & Sons, Sons*, 2002.

[6] J. Moor, *The Dartmouth College Artificial Intelligence Conference*, 27487th ed. AI Magazine, 2006.

[7] J.S., Olmstead, W.E. Angell, "Singularly perturbed Volterra integral equations," *SIAM J. Appl. Math.*, vol. 47, no. 1, pp. 1–14, 1987.

[8] Bijura A.M., "Rigorous results on the asymptotic solutions of singularly perturbed nonlinear Volterra integral equations," *J. Integral Equ. Appl*, vol. 14, no. 2, pp. 119–149, 2002.

[9] Jordan G.S., "A nonlinear singularly perturbed Volterra integrodifferential equation of nonconvolution type," *Proc.R. Soc. Edinb. A* 80, pp. 235–247, 1978.
[10] J.-P Kauthen, "A survey of singularly perturbed Volterra equations," *Appl. Numer. Math*, vol. 24, pp. 95–114, 1997.
[11] S. S. Behzadi, "Solving Fuzzy Nonlinear Volterra-Fredholm Integral Equations by Using Homotopy Analysis and Adomian Decomposition Methods," *International of Fuzzy Set Valued Analysis*, vol. 35, p. 13 pages, 2011.
[12] H. Roman-Flores Y. Chalco-Cano, "On New Solutions of Fuzzy Differential Equations. Chaos," *Solitons and Fractals*, vol. 38, pp. 112-119, 2008.
[13] A. Alidema. . , e, 793: A. Georgieva, "Convergence of homotopy perturbation method for solving of two-dimensional fuzzy Volterra functional integral equations," *Advanced Computing in Industrial Mathematics ,Studies in Computational Intelligence*, vol. 793, pp. 129–145, 2019.
[14] S. Hristova. A. Georgieva, "Homotopy Analysis Method to Solve Two-Dimensional Nonlinear Volterra-Fredholm Fuzzy Integral Equations," *Fractal Fract*., vol. 4, 2020.
[15] I. Naydenova. A. Georgieva, "Application of homotopy analysis method for solving of two-dimensional linear Volterra fuzzy integral equations," in *AIP Conference Proceedings*, 2019, pp. 2159, 030012.
[16] L. Hooshangian., "Nonlinear Fuzzy Volterra Integro-diferential Equation of N-th Order: Analytic Solution and Existence and Uniqueness of Solution," *Int. J. Industrial Mathematics*, vol. 11, no. 1, 2019.
[17] G. Adomian., "Solving Frontier Problems of Physics," *Kluver Academic Publishers,Boston* , vol. 12, 1994.
[18] Hans-Jiirgen, *Fuzzy set theory--and its applieations, 4th ed. Zimmermann: Library of Congress Cataloging-in-Publication Data Zimmermann*, 1934.
[19] D. and Kruse, R. Nauck, "Neuro-Fuzzy Classification with NEFCLASS," in *Operations Research Proceedings, Berlin*, 1996, pp. pp. 294-299.
[20] Otadi M Mosleh M., "Simulation and Evaluation of Fuzzy Differential Equations by Fuzzy Neural Network," *Applied Soft Computing* , vol. 12, pp. 2817-2827, 2012.
[21] J. J. and Hayashi, Y. Buckley, "Neural networks for fuzzy systems," *Fuzzy Sets and Systems* , vol. 71, pp. pp. 265-276, 1995.
[22] M. Matinfar, M. Ghanbari, and R. Nuraei, "Numerical solution of linear fuzzy Volterra integro-differential equations by variational iteration method," *Intell. Fuzzy Syst*, vol. 24, pp. 575–586, 2013.