RESEARCH PAPER
Future Research in AI & IoT, 2025, 1(2)
Homepage: https://frai.journals.umz.ac.ir/
DOI:

An Application of Improved Neural Networks to Solve Differetial Equation based Z-Process

Somayeh Ezadi[1*], Mohammad Ali Fariborzi Araghi[2], Masoud Baghfalaki[3], Nader Biranvand[4]

[1]*Department of Mathematics, N.T.C., Islamic Azad University, Tehran, Iran,*
[2]*Department of Mathematics, C.T.C, Islamic Azad University, Tehran, Iran,*
[3]*Department of Mathematics, Ker.C., Islamic Azad University, Kermanshah, Iran,*
[4]*Department of Mathematics, Faculty of Basic Sciences Imam Ali University, Tehran, Iran.*

Article Info
Received -----
Accepted -----
Available online -----

Keywords:
Fuzzy Numbers;
Z-Numbers;
First-Order Differential Equation;
Improved Neural Network.

Abstract:
In this study, we employ an enhanced neural network architecture to develop a novel approach for solving differential equations involving Z-number-based initial value estimation. The proposed methodology operates by evaluating Z-numbers through the function $[x_T(t)]^Z = \big(A_T(t), B_T(t)\big)$, which $A_T(t)$ serves as a constraint function, while $B_T(t)$ quantifying the reliability measure of $A_T(t)$. To formulate the differential equation within the Z-number framework (ZDE), the function $[x_T(t)]^Z$ is expressed in the decomposed form $[x_T(t)]^Z = ((\underline{A}_T(t), \overline{A}_T(t)), B_T(t))$. The generalized neural network structure comprises three distinct layers. Input Layer: Consisting of input variables, first-layer weights, and network biases. The number of weights is determined by the number of governing equations in the primary problem, ensuring alignment with the input dimensionality. Hidden Layer: Composed of neurons equipped with nonlinear activation functions. Output Layer: Producing the final output via linear activation functions and associated weights. Notably, the enhanced neural network processes real-valued inputs, whereas its weights and outputs adhere to Z-valuation. To train this modified neural network, we define the objective function as the sum of squared errors (SSE), which is subsequently minimized using an optimization algorithm to determine the optimal network weights. The resulting solution generated by this method exhibits convergence to the true analytical solution. To validate the efficacy and applicability of the proposed approach in approximating exact solutions, we provide two illustrative numerical examples.

***Corresponding Author:** somayeh.ezadi@yahoo.com
Supplementary information: Supplementary information for this article is available at https://frai.journals.umz.ac.ir/
Please cite this paper as:

1. Introduction

The solution of fuzzy differential equations with fuzzy initial values represents an important research area in fuzzy mathematics and engineering sciences, as evidenced by numerous studies [1,3,4,9,10,15]. A persistent challenge in this field has been the inherent uncertainty in information reliability, necessitating new analytical approaches. Zadeh's Z-number concept [17] offers a promising solution, demonstrating superior capability in representing human knowledge compared to conventional fuzzy numbers [14]. While significant research exists on fuzzy numbers [2,5,6,7,12,13,16,18], applications to differential equations using Z-numbers remain limited [14]. This paper presents a novel numerical approach for solving Z-number-based differential equations employing Multi-layer Perceptron (MLP) networks. The MLP-based method offers distinct computational advantages due to its network architecture. The structure of this paper is as follows:

In Section 2, the fundamental and essential concepts have been stated. In Section 3, the estimate of ZDE has been

ISSN

stated using a Z-number neural network. In Section 4, a numerical example is presented. In Section 5, the conclusion is provided.

2. The fundamental and essential concepts

This section introduces the essential definitions and fundamental theorems needed to formulate the proposed model.

Definition 2.1. Z-number

A Z-number [17] is an ordered pair $Z = (A, B)$ characterizing uncertain information about a random variable X, where:

1. Constraint Component (A):

-A fuzzy set defining a possibilistic restriction on X

- Membership function $\mu_A: R \to [0,1]$ quantifies allowable values

2. Reliability Component (B):

-A fuzzy set representing confidence in A

- Membership function $\mu_B: [0,1] \to [0,1]$ grades probability measures

Mathematical Representation:
For a Z-valuation " (X, A, B)", the semantics follow:

$$P(\mathrm{X}\ is\ A) = \int_R \mu_A(u) P_X(u)\ du\ is\ B$$

Where $P_X(u)$ is the (generally unknown) probability density of X. The integral computes the probability of the fuzzy event $\{ X \text{ is } A \}$.B fuzzifies this probability measure.

Definition 2.2. (See [13]).

Let Z^* denote the space of all Z-numbers. For an arbitrary Z-number $Z = (A, B) \in Z^*$, its parametric form is expressed as an ordered pair of characteristic functions:

$$Z = \left(\left(\underline{A}(r), \overline{A}(r) \right), B \right), r \epsilon [0,1],$$

Its components satisfy the following requirements:

1. $\underline{A}(r) \leq \overline{A}(r)\ \ for\ r \epsilon [0,1]$,

2. $\underline{A}(r)$ and $\underline{B}(r)$ are bounded, left-continuous functions, and monotonically non-decreasing on the interval $[0,1]$.

3. . $\overline{A}(r)$ and $\overline{B}(r)$ are bounded, left-continuous functions, and monotonically non-increasing on $[0,1]$.

4. $B = f(\underline{A}(r), \overline{A}(r))$.

Definition 2.3. (See [13]). Let $Z = (A, B)$ be a Z-number. The $Z - numbers$ is said to be normal if the height of its reliability component B is exactly 1 (i.e., $h(B) = 1$).

2.4. MLP Network and it training

An MLP represents a type of feedforward artificial neural network architecture. This network structure contains a minimum of three node layers, where all nodes except the input layer employ nonlinear activation functions. Unlike simple linear perceptrons, MLPs can handle non-linearly separable data due to their layered structure and nonlinear activation capabilities. For training, MLPs typically employ supervised learning through the backpropagation algorithm. Among various learning methods available for multi-layer networks, backpropagation has become the most widely adopted approach, operating on the principle of error correction learning.
The backpropagation process requires computation of sensitivity gradients across different network layers, which necessitates calculating derivatives of the neurons' activation functions. Consequently, only differentiable activation functions can be utilized in this framework. A commonly used example is the sigmoid function, expressed mathematically as $S(n) = \frac{1}{1+e^{-n}}$, which produces outputs constrained to the interval [0, 1].

Theorem 2.5. An MLP network featuring a single hidden layer with sigmoid activation (specifically the hyperbolic tangent function) and linear transformations in the output layer possesses universal approximation capabilities. This architecture can approximate any square-integrable function to arbitrary precision (as demonstrated in [8]).

2.6. BFGs Technique
The neural network training process involves computing a predefined error function by comparing the network's output values with the correct targets, then propagating this error back through the network. Through iterative adjustments using this error information, the algorithm modifies each connection's weight to gradually reduce the error value. After sufficient training cycles, the network typically converges to a state with minimal computational error. This weight adjustment process essentially solves an unconstrained optimization problem that can be addressed through various minimization techniques including, the steepest descent method, conjugate gradient method, or Quasi-Newton methods. While the Newton method is an important nonlinear optimization algorithm, its main limitation lies in requiring the computation of the second derivative matrix (Hessian matrix). Quasi-Newton methods, developed through the seminal work of Davidon (1959) and Fletcher-Powell (1963), address this limitation by constructing successive approximations to the Hessian matrix, thereby avoiding its direct computation. In this implementation, we specifically utilize the Quasi-Newton BFGS method as referenced in [11], which provides an efficient approach to estimating the Hessian matrix for optimization purposes.

3. The definition of a derivative-based $Z - numbers$.

Let Z^* be the set of all $Z - numbers$. The general initial value problem for $Z - numbers$ differential equations takes the form:

$$\begin{cases}[x'(t)]^Z = f(t,[x(t)]^Z) \\ [x(t_0)]^Z = x_0 \in Z^* \end{cases} \quad (3.1)$$

With the temporal variable t belonging to the closed interval $[t_0, T]$ (a subset of positive reals), with $f: [t_0; T] \times Z^* \to Z^*$ being a continuous mapping, and x_0 denotes a Z-number in Z^*, and $x(\mathrm{t}) = (A_x, B_x)$.
Suppose, x' is a function with Z-valuation, and we define it as follows

$$[x'(t)]^Z = (A_{x'}, B_{x'}) \quad (3.2)$$

In this formulation, $A_{x'}$ represents the constraint component while $A_x(t)$ characterizes the reliability measure of these constraints. For the ordered pair $(A_{x'}, B_{x'})$, under the assumption that the Hukuhara difference exists between $A_x(t+h)$ and $A_x(t)$, we define:

$$A_{x'} = \lim_{h\to 0} \frac{A_x(t+h) -_h A_x(t)}{h}, \qquad B_{x'} = p(x'(\mathrm{t})\ is\ A_{x'}) \quad (3.3)$$

Where $p(x'(\mathrm{t})\ is\ A_{x'})$ represents the probability distribution function associated with the derivative process. Equation (3.1) can be rewritten in the form of the parameter defined in [17]:

$$\begin{cases}[x'(t)]^Z = (A_{x'}, B_{x'}) \\ [x(t_0)]^Z = \left(A_{x_0}, B_{x_0}\right), \end{cases} \quad (3.4)$$

In equation (3.2), $A_{x'}$ represents a fuzzy-valued function while $B_{x'}$ is a real-valued function. Here, A_{x_0} possesses a specific fuzzy quantity and Bx0Bx0 a real quantity. The functions $A_{x'}$ and A_{x_0} serve as constraint operators, whereas $B_{x'}$ and B_{x_0} quantify the reliability measures for $[x'(t)]$ and $[x(t_0)]^Z$ respectively. The fuzzy constraint function $A_{x'}$ is formally defined as:

$$A_{x'} = f\left(t, \tilde{x}(t)\right) \quad (3.5)$$

With the following parametric form

$$[A_{x'}]^r = \begin{cases} \overline{A}_{x'}(t) = \overline{f}(t,x) = G\left(t, \underline{x}, \overline{x}\right), & \overline{x}(t_0) = \overline{x}_0 \\ \underline{A}_{x'}(t) = \underline{f}(t,x) = F\left(t, \underline{x}, \overline{x}\right), & \underline{x}(t_0) = \underline{x}_0 \end{cases} \quad (3.6)$$

Where $[.]^\alpha$ is the symbol of r-cut, and

$$\begin{cases} G\left(t, \underline{x}, \overline{x}\right) = max\{f(t,u) | u \in [\underline{x}, \overline{x}]\}, \\ F\left(t, \underline{x}, \overline{x}\right) = min\{f(t,u) | u \in [\underline{x}, \overline{x}]\}, \end{cases} \quad (3.7)$$

Theorem 3.1. The Z-numbers initial value problem (ZIVP) in relation (3.1) has a unique Z-process solution.

Proof. According to [17], relations (3.1) and (3.4) are mathematically equivalent. In (3.4), $A_{x'}$ represents a fuzzy function defined in (3.5). For each α -cut, the relation (3.5) and relation (3.6) are equivalent. Furthermore, Theorem 3.2 in [15] guarantees the existence of a unique solution to the fuzzy initial value problem in (3.6). An analogous argument applies to $B_{x'}$. Consequently, the uniqueness of the solution to ZIVP (3.1) follows directly.

Assumption, $x_T(t)$ is an approximate solution for $Z(t)$ of Relation (3.5). It is defined as follows:

$$x_T(t) = \left(A_T(t), B_T(t)\right) \quad (3.8)$$

Where

$$A_T(t) = \left(\underline{A}_T(t), \overline{A}_T(t)\right), \quad (3.9)$$

$$B_T(t) = \begin{cases} 1 & if \quad A_T(t_0) = A(t_0) \\ 1 - \left(e^{-\lambda Z}\right)|_{A_T} & if \quad A_T(t_0) \neq A(t_0), \ \lambda \in [0,1]. \end{cases} \quad (3.10)$$

And

$$\begin{cases} \underline{A}_T(t) = f_1(\underline{A}_T(t_0), \underline{N}_A(t,p)) \\ \overline{A}_T(t) = f_2(\overline{A}_T(t_0), \overline{N}_A(t,p)) \end{cases} \quad (3.11)$$

Where $f_i, \ \forall i = 1,2$ is an arbitrary function based on the neural network. In this work, we define f as follows

$$f\left(A_T(t_0), N_A(t,p)\right) = A_T(t_0) - tN_A(t,p),$$

Where N_A is the artificial neural network and its formula is shown in relation (3.17).

$$B_T(t) = p\left(X_T(t)\ is\ A_T(t)\right) \quad (3.12)$$

B_T is a probability distribution function based on A_T, which can vary depending on the type of problem and the initial and boundary conditions. The criterion is determined according to the type of problem and the initial condition. Now we have to be derived

$$x'_T(t) = \left(A'_T(t), \hat{B}_T(t)\right) \quad (3.13)$$

Where

$$A'_T(t) = \left(\underline{A'}_T(t), \overline{A'}_T(t)\right),$$

So that:

$$: \begin{cases} A'_T(t) = t\frac{\partial f_1}{\partial t}(\underline{A}_T(t_0), t\underline{N}_A(t,p)) \\ A'_T(t) = t\frac{\partial f_2}{\partial t}(\overline{A}_T(t_0), t\overline{N}_A(t,p)) \end{cases} \quad (3.14)$$

and

$$\hat{B}_T(t) = p\left(X'_T(t)\ is\ A'_T(t)\right) \quad (3.15)$$

$$\hat{B}_T(t) = \begin{cases} 1 & if \quad A'_T(t_0) = A'^{(t_0)} \\ 1 - \left(e^{-\lambda y}\right)|_{A'_T} & if \quad A'_T(t_0) \neq A'(t_0) \end{cases} \quad (3.16)$$

Consider a MLP architecture consisting of:

- A single hidden layer containing H neurons with hyperbolic tangent activation functions

- A linear output unit

Under this configuration, we establish the following:

$$N_A = \sum_{i=1}^{H} V_i\, S(Q_i), \qquad Q_i = w_i t + b_i \tag{3.17}$$

Where w_i is a weight parameter from input layer to the ith hidden layer, V_i is an ith weight parameter from hidden layer and Q_i signify the bias term associated with the ii-th hidden unit. The output of the ii-th hidden unit is processed through an arbitrary hyperbolic tangent activation function, denoted as SS. In neural networks, activation functions are employed to constrain the output of neurons, typically confining their values to intervals such as [0,1] or [-1.1]. In this framework, we utilize both a linear activation function and a hyperbolic tangent activation function, though, in principle, alternatives such as a sigmoid function (or hyperbolic tangent) may also be applied. The hyperbolic tangent function maps its output to the interval [-1.1] and is differentiable, making it particularly suitable for training multi-layer networks via error backpropagation algorithms. We now consider a MLP architecture comprising a single hidden layer with sigmoidal activation units and a linear output unit. To facilitate subsequent analysis, we derive the first-order derivatives as follows:

$$S(Q_i) = \frac{1}{1 + e^{-Q_i}} \tag{3.18}$$

$$S'(Q_i) = (1 - S(Q_i))\, S(Q_i) \tag{3.19}$$

The derivatives of N with respect to input x_i is:

$$\frac{\partial N_A}{\partial t} = \sum_{i=1}^{H} V_i\, w_i S'(Q_i) \tag{3.20}$$

Consequently, the total error function to be minimized with respect to all trainable parameters of the neural network is given by:

$$E(p) = \sum_{i=1}^{m} \left([x'(t)]^Z - x'_T(t) \right)^2 \tag{3.21}$$

For relation (3.21), we have

$$\min_{p} \sum_{i=1}^{m} e^{\left(x'_{A_1}(t_i,p) - x'_{TA_1}(t_i,p)\right)^2} - e^{\left(x'_{A_2}(t_i,p) - x'_{TA_2}(t_i,p)\right)^2} \tag{3.22}$$

The objective function in (3.22) can be optimized via the BFGS quasi-Newton technique [12].

4. Numerical Examples

In this section, we present two numerical examples to demonstrate the behavior and key characteristics of the proposed method. The computational experiments were performed in MATLAB R2012a, utilizing the unconstrained optimization solver fminunc to minimize the objective function defined in Equation (3.15). The initial weight parameters were initialized using a random sampling scheme.

Example 1. We examine a first-order differential equation with Z-number initial conditions (ZIVP) of the form:

$$\begin{cases} X' = X(t), & t \in [0,1] \\ X(0) = \left((0.75 + 0.25r, 1.125 - 0.125r), (1)\right) \end{cases} \tag{4.1}$$

The exact solution (for $t = 1$) of the ZIVP is:

$$X(1,r) = \Big(\big((0.75 + 0.25r)e, (1.125 - 0.125r)e\big), 1 - \left(e^{-\lambda Z}\right)|_A \Big), r \in [0,1] \tag{4.2}$$

The trial solution is given as

$$X_T(t) = \left(\left(\underline{A}_T(t), \overline{A}_T(t) \right), B_T(t) \right) \tag{4.3}$$

Where

$$\begin{cases} \underline{A}_T(t) = 0.75 + 0.25r + tN_{A_1}(t,p) \\ \overline{A}_T(t) = 1.125 - 0.125r + tN_{A_2}(t,p) \end{cases} \tag{4.4}$$

$$B_T = \begin{cases} 1 & if \quad A_T(t_0) = A(t_0) \\ 1 - \left(e^{-\lambda y}\right)|_{A_T} & if \quad A_T(t_0) \neq A(t_0) \end{cases} \tag{4.5}$$

Where $t \in [0,1]$. The error function was trained using a network architecture comprising $H = 5$ sigmoidal activation units in the hidden layer, evaluated across $m = 50$ uniformly distributed discretization points within the interval [0, 1]. The exact solutions for X are presented in Table 4.1, while the optimized weight and bias parameters are tabulated in Table 4.2. A comparative analysis between the radial basis function (RBF) approach [18] and our proposed methodology is provided in Table 3. Figure 1 illustrates both the analytical solution and the trial function at the specific temporal value $t = 1$.

Table 4.1. The exact X solutions for Example 1 ($\lambda = 1$).

r	$\underline{A}(t)$	$\overline{A}(t)$	$B(t)$
0	2.038711	3.058067	0.916781991
0.1	2.106668	3.024088	0.926959521
0.2	2.174625	2.990110	0.936631141
0.3	2.242582	2.956131	0.945835810
0.4	2.310539	2.922152	0.954609983
0.5	2.378496	2.888174	0.962987909
0.6	2.446453	2.854195	0.971001959
0.7	2.514410	2.820217	0.978682374
0.8	2.582367	2.786238	0.986057857
0.9	2.650324	2.752260	0.993155222
1	2.718281	2.718281	1.000000000

Table 4.2. The numerically determined optimal weights w, v and biases b that minimize the objective function for example 1.

i	1	2	3	4	5
$\underline{v}_{iA}$	-0.15411	2.860106	0.423156	0.72548	0.934447
$\underline{w}_{1iA}$	1.62064	2.069084	0.410291	0.887180	1.591845
$\underline{w}_{2iA}$	0.329611	-1.790976	0.482489	0.245760	-0.763779
$\underline{b}_{iA}$	0.202012	-2.395825	0.606070	0.496051	-0.258443
$\overline{v}_{iA}$	1.340129	0.067314	0.352386	1.548856	1.1187202
$\overline{w}_{1iA}$	2.026426	0.534770	0.974960	1.161197	0.6929401

$\overline{w}_{2iA}$	-1.769286	0.244643	-0.062266	-1.41148	0.3465096
$\overline{b}_{iA}$	-1.59365	0.331701	0.363209	-0.97593	-0.280726

Table 4.3. Comparison of the exact X and X_T approximated solutions for $\lambda = 1$.

	RBF method [18]			Proposed method		
r	$\underline{A}_T(t)$	$\overline{A}_T(t)$	$B_T(t)$	$\underline{A}_T(t)$	$\overline{A}_T(t)$	$B_T(t)$
0	2.00	3.04	0.874	2.038778	3.058075	0.916790338
0.1	2.07	3.02	0.885	2.106675	3.024085	0.926960518
0.2	2.14	2.99	0.899	2.174666	2.990107	0.936635951
0.3	2.21	2.95	0.911	2.242599	2.956165	0.945835846
0.4	2.28	2.92	0.925	2.310402	2.922136	0.954597251
0.5	2.35	2.88	0.937	2.378514	2.888173	0.962989633
0.6	2.42	2.84	0.950	2.446465	2.854227	0.971001155
0.7	2.50	2.81	0.962	2.514394	2.820183	0.978683106
0.8	2.57	2.78	0.975	2.582387	2.786251	0.986058567
0.9	2.64	2.75	0.987	2.650339	2.752261	0.993156217
1	2.71	2.71	1	2.718295	2.718263	0.999997888

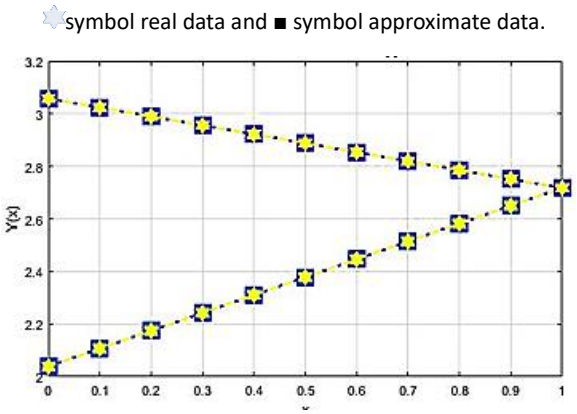

Fig. 1. The graphical representation of Example 1's solution shows the exact solution superimposed with the numerical approximation, with evaluation points selected both within the interval [0, 1] and in its exterior region.

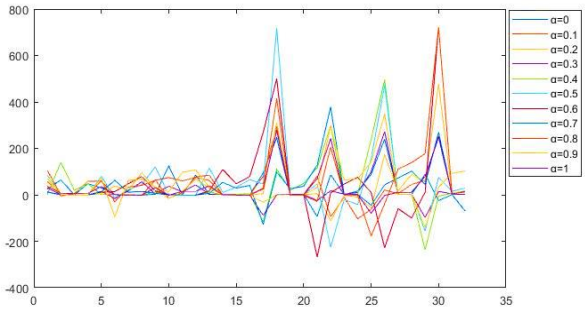

Fig. 2. The convergence of the neural network weights for each r-cut.

Example 2. Consider the following first-order ZIVP:

$$\begin{cases} X' = 3t^2X(t), & t \in [0,1] \\ X(0) = \left(\left(0.5\sqrt{r}, 0.2\sqrt{1-r}+0.5\right), (1)\right) \end{cases} \quad (4.6)$$

The exact solution (for $t = 1$) of the ZIVP is:

$$X(1,r) = \left(\left(0.5\sqrt{r}e, \left(0.2\sqrt{1-r}+0.5\right)e\right), 1 - \left(e^{-\lambda Z}\right)|_A\right), \quad r \in [0,1] \quad (4.7)$$

The trial solution is given as

$$X_T(t) = \left(\left(\underline{A}_T(t), \overline{A}_T(t)\right), B_T(t)\right)$$

$$\begin{cases} \underline{A}_T(t) = 0.5\sqrt{r} + tN_{A_1}(t,p) \\ \overline{A}_T(t) = 0.2\sqrt{1-r} + 0.5 + tN_{A_2}(t,p) \end{cases} \quad (4.8)$$

$$B_T = \begin{cases} 1 & if \quad A_T(t_0) = A(t_0) \\ 1 - \left(e^{-\lambda Z}\right)|_{A_T} & if \quad A_T(t_0) \neq A(t_0) \end{cases} \quad (4.9)$$

Where $t \in [0,1]$. This error function for the H = 5 Sigmoid function units in the hidden layer and for m = 50 equally spaced points inside the interval [0, 1] is trained. The exact X solutions can be seen in Table 4.4. The optimal value of the weights and biases are shown in Table 4.5. Table 6 provides a quantitative comparison of the results obtained from the RBF method [18] and our proposed approach. The analytical solution alongside the trial function are graphically depicted in Figure 3 for the specific case when $t = 1$.

Table 4.4. The exact X solutions for Example 2 ($\lambda = 1$).

r	$\underline{A}(t)$	$\overline{A}(t)$	$B(t)$
0	0.00000	1.90279	0.149151904
0.1	0.42979	1.87489	0.502726121
0.2	0.60782	1.84540	0.613425457
0.3	0.74443	1.81399	0.687997482
0.4	0.85959	1.78025	0.745260379
0.5	0.96105	1.74356	0.792405599
0.6	1.05278	1.70297	0.833175486
0.7	1.13713	1.65691	0.869989197
0.8	1.21565	1.60227	0.904921316
0.9	1.28939	1.53106	0.940867512
1	1.35914	1.35914	1

Table 4.5. The numerically determined optimal weights w, v and biases b that minimize the objective function for Example 1.

i	1	2	3	4	5
$\underline{v}_{iA}$	1.4862	-0.1474	-0.7942	0.9962	0.1256
$\underline{w}_{1i}$	1.6433	1.53310	0.7359	0.4520	0.7035
$\underline{w}_{2i}$	-1.0322	-0.0688	0.4702	0.5763	0.6266
$\underline{b}_{iA}$	-1.0013	-0.1832	0.0519	1.0962	0.1924
$\overline{v}_{iA}$	0.7624	-1.0011	1.3368	-0.3780	0.7421
$\overline{w}_{1i}$	0.0390	1.2951	1.3953	1.1182	1.1241
$\overline{w}_{2i}$	1.3646	0.4911	-0.4115	0.2215	-0.2635
$\overline{b}_{iA}$	1.6567	0.3276	-0.5685	0.4276	0.1246

Table 4.6. The exact solution X and its approximate counterpart X_T are systematically compared for the specific case when λ takes unit value.

	RBF method [18]			Proposed method		
r	$\underline{A}_T(t)$	$\overline{A}_T(t)$	$B_T(t)$	$\underline{A}_T(t)$	$\overline{A}_T(t)$	$B_T(t)$
0	-0.000000002	1.87	0.687	0.00000	1.90279	0.14919892
0.1	0.41	1.85	0.769	0.42979	1.87489	0.502631957
0.2	0.58	1.82	0.804	0.60782	1.84540	0.613424751
0.3	0.72	1.79	0.833	0.74443	1.81399	0.687994362
0.4	0.83	1.76	0.856	0.85959	1.78025	0.745258693
0.5	0.93	1.72	0.878	0.96105	1.74356	0.792463446
0.6	1.02	1.68	0.899	1.05278	1.70297	0.833178975
0.7	1.11	1.64	0.919	1.13713	1.65691	0.869996219
0.8	1.19	1.58	0.940	1.21565	1.60227	0.904923532
0.9	1.27	1.51	0.963	1.28939	1.53106	0.940756364
1	1.34	1.34	1	1.35914	1.35914	0.999660675

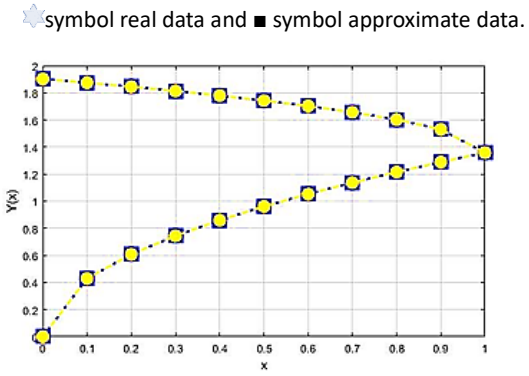

Fig. 3. The graphical representation of Example 2's solution shows the exact solution superimposed with the numerical approximation, with evaluation points selected both within the interval [0, 1] and in its exterior region.

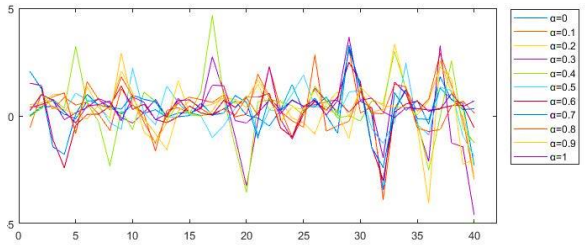

Fig. 4. The convergence of the neural network weights for each α-cut.

The results of the proposed method in both of the above examples show that the approximate value obtained by the MLP method has much better accuracy compared to the RBF method [18].

5. Conclusion

Many physical phenomena across mathematical sciences, physics, mechanics, and scientific computations are typically modeled using differential equations, with a significant portion operating in uncertain environments. Solving these equations analytically within finite or semi-infinite domains proves challenging, often lacking both sufficient accuracy and computational efficiency. Consequently, numerical approaches become valuable for addressing such problems. This paper presents a novel approach using Z-number-based initial value differential equations (ZDEs) and develops a corresponding solution model employing MLP neural networks within a nonlinear optimization framework. We pioneer the application of neural networks for approximating solutions to ZDEs, specifically implementing sigmoid or hyperbolic tangent activation functions in the MLP's hidden layer to achieve high-precision results. A key benefit of our method lies in its flexibility to adjust both the number of hidden layers and training points according to problem-specific requirements, enabling enhanced solution accuracy. The fundamental motivation for employing neural networks stems from their well-established capability in function approximation tasks.

References

1. Allahviranloo, T., Ahmady, N., Ahmady, E.: Numerical solution of fuzzy differential equations by redictorcorrector method, Information Sciences 177, 1633-1647 (2007).

2. Allahviranloo,T and Ezadi, S., Z-Advanced numbers processes, Information Sciences, 480, 130-143 (2019).

3. Barzegar Kelishami, H., Fariborzi Araghi, M. A., Amirfakhrian, M.: Applying the fuzzy CESTAC method to find the optimal shape parameter in solving fuzzy differential equations via RBF-meshless methods. Soft Comput. 24 (20) 15655-15670 (2020).

4. Ezadi, S., Parandin, N., GHomashi, A.: Numerical Solution of Fuzzy Differential Equations Based on Semi-Taylor by Using Neural Network, J. Basic. Appl. Sci. Res., 3(1s)477-482 (2013).

5. Ezadi, S., Allahviranloo, T.: New multi-layer method for Z-number ranking using Hyperbolic Tangent function and convex combination, Intelligent Automation Soft Computing., 1-7 (2017).

6. Ezadi, S., Allahviranloo, T.: Numerical solution of linear regression based on Z-numbers by improved neural network, IntellIgent AutomAtIon and Soft ComputIng, 1-11 (2017).

7. Ezadi, S., Allahviranloo, T.: Two new methods for ranking of Z-numbers based on sigmoid function and sign method, International Journal of Intelligent Systems., 1-12 (2018).

8. Hornick, K., Stinchcombe, M.: White H. Multilayer feedforward networks are universal approximators, Neural Networks, 2, 359-366 (1989).

9. Luenberger, D.G.: Linear and Nonlinear Programming, second ed., Addison Wesley, (1984).

10. Lee, H., kang, I.S.: Neural algorithms for solving differential equations, journal of computational physics 91, 110-131 (1990).

11. Liu, C., and Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming, 45(3) 503-528 (1989).

12. Mohamad, Shaharani, D. S. A. and Kamis, N. H. A.: Z-number based decision making procedure with ranking fuzzy numbers method, AIP Conference Proceedings., 1635, 160–166 (2014).

13. Pirmuhammadi, S., Allahviranloo, T., Keshavarz, M.: The parametric form of Z-number and its application in Z-number initial value Problem, (2017).

14. Qalehe, L. Afshar Kermani, M., Allahviranloo, T.: Solving First-Order Differential Equations of Z-numbers initial value using Radial Basic Function, International Journal of Differential Equations, 1-11 (2020).

15. Seikkala, S.: On the fuzzy initial value problem, Fuzzy Sets and Systems 24, 319-330 (1987).

16. Yager, R.R.: On Z-Valuations Using Zadeh's Z-Numbers, International journal of intelligent systems, 27, 259–278 (2012).

17. Zadeh, L A.: A Note on Z-numbers, Information Sciences 181, 2923–2932 (2011).

18. J. B. Zajia, J. A. Morente-Molinera, I. A. Díaz, Decision Making by Applying Z-Numbers, Doctoral Symposium on Information and Communication Technologies, (2022), 32-43.