ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 10 (2017) 470–480

Using ANNs Approach for Solving Fractional Order Volterra Integro-differential Equations

Ahmad Jafarian [1] , Fariba Rostami [1] , Alireza K. Golmankhaneh [2] *, Dumitru Baleanu [3,4]

[1] Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran

E-mail: jafarian5594@yahoo.com

[2] Young Researchers and Elite Club, Urmia Branch, Islamic Azad university, Urmia, Iran

E-mail: a.khalili@iaurmia.ac.ir

[3] Department of Mathematics Çankaya University, 06530 Ankara, Turkey

[4] Institute of Space Sciences, P.O.BOX, MG-23, R 76900, Magurele-Bucharest, Romania

E-mail: dumitru@cankaya.edu.tr

Received 14 July 2016

Accepted 17 November 2016

Abstract

Indeed, interesting properties of artificial neural networks approach made this non-parametric model a powerful tool in solving various complicated mathematical problems. The current research attempts to produce an approximate polynomial solution for special type of fractional order Volterra integro-differential equations. The present technique combines the neural networks approach with the power series method to introduce an efficient iterative technique. To do this, a multi-layer feed-forward neural architecture is depicted for constructing a power series of arbitrary degree. Combining the initial conditions with the resulted series gives us a suitable trial solution. Substituting this solution instead of the unknown function and employing the least mean square rule, converts the origin problem to an approximated unconstrained optimization problem. Subsequently, the resulting nonlinear minimization problem is solved iteratively using the neural networks approach. For this aim, a suitable three-layer feed-forward neural architecture is formed and trained using a back-propagation supervised learning algorithm which is based on the gradient descent rule. In other words, discretizing the differential domain with a classical rule produces some training rules. By importing these to designed architecture as input signals, the indicated learning algorithm can minimize the defined criterion function to achieve the solution series coefficients. Ultimately, the analysis is accompanied by two numerical examples to illustrate the ability of the method. Also, some comparisons are made between the present iterative approach and another traditional technique. The obtained results reveal that our method is very effective, and in these examples leads to the better approximations.

Keywords: Fractional equation; Power-series method; Artificial neural networks approach; Criterion function; Back-propagation learning algorithm.

* Corresponding Author.

1. Introduction

Fractional calculus is old subject but it recently has found numerous applications in various fields of science and engineering [1,2,3,4,5,6]. Fractional derivatives are powerful and efficient tools to describe the physical systems that have long-term memory [7,8]. Especially, in modeling the complex dynamic systems it can be seen that fractional differential equations naturally arise once power-type non-local interacting systems with power-law memory [5,9,10]. Differential equations of the fractional order have gained applications in modeling non-conservative systems [11,4]. A large amount of research works has been devoted to tackle a wide variety of fractional equations, particularly generalized odd dimension mechanics is suggested in Refs [12,13,14,15].
The analytical and approximate analytical methods are used to achieved the solutions of differential equations [16]. The fractional deferential equations are solved using generalized analytical, approximate analytical methods and numerical methods[17,6]. The iterative methods is utilized widely and found the solution of the equations in the area of the initial and boundary value problems. We already have a number of famous computational techniques dealing with the solution of high order fractional differential equations, such as weighted essentially non-oscillatory scheme [18], Chebyshev collocation method [19,20], Green functions method [21] and block-by-block approach [22]. Moreover, as an excellent modeling tool, fractional integro-differential equations issue have attracted much attention recently [23,24].
The artificial neural networks (ANNs) approach has attracted much consideration to its advantages, such as learning, adaptivity, fault-tolerance, etc [25,26,27,28]. Recently, a modification of ANNs approach has been developed to the treatment of related boundary problem in two-dimensional case . On the other hand, power series method has received considerable attention in dealing with complicated math problems. This method provides the solution function as a series polynomial in which its coefficients can be determined via an appropriate standard method. Here, we intend to extend the application of neural networks approach and power series method as an iterative minimization strategy for the numerical solution of the mentioned fractional order initial value problem. It is worthy of note that, representation of solution in terms of a suitable truncated series expansion is the fundamental issue in approximation theory. After discretizing the problem domain into elements and making substitutions, the mentioned equation is transformed to minimizing problem by using the least mean square (LMS) error function. Now, the generalized delta learning rule is used to minimize the mentioned error function on the unknown space. After determining the unknowns, the solution can be calculated via a convergent series polynomial [29].
The brief outline of this paper consists of the following steps:

Current research begins the procedure in section 2, by reviewing basic definitions and fundamental issues of neural networks and fractional calculus. Here, the given initial value problem is also investigated via the offered combination numerical method. Two numerical examples are given in section 3 to show that our method yields accurate approximates of the mentioned problem. Since, the mentioned problem usually has no exact solution in the fractional order, the obtained numerical results can not be compared with exact ones. So to have the better understanding of the method, we help the error caused by the proposed method, and we show that by growing the number of iterations, this error is reduced. Finally, conclusion is described in section 4.

2. Description of the Method

As indicated, unique capabilities of artificial neural networks have persuaded researchers to serve these as powerful tools in solving complex real world problems. The main aim of this section is to combine the ANNs approach and a modification of polynomial series technique as an efficient iterative method to approximate numerical solution of a linear Fredholm integro-differential equations of fractional order. In order to better clarify all the fundamental mathematical features of the method pre-

sented in this research, first some preliminary definitions and necessary notations in neural networks are given, which will be utilized in the following parts.

2.1. Basic Idea of ANNs

First, let us have a brief part of introduction on typical neural networks. The most striking question about neural networks is why they have been applied. To gain insight into this question, we go through the paper and also we will explore the reasons why they are useful for solving certain types of tasks and it is believe that algorithmic and math problems are solved by computers which are as a specific piece of technology. It is sometimes seen that there are likely a lot of failed mathematical algorithm. There are a couple of problems that are not effortless to be expressed into an algorithm such as facial recognition and language processing. However, these tasks are trivial to humans. One important and pioneering feature of artificial neural networks is that their design enables them to process data in a similar way to our own biological brains which is done by drawing inspiration from the form of our own nervous system functions. This qualifies them more sophisticated at solving problems like facial recognition, which our biological brain is capable of doing it easily. We also do a good job understanding of operating the designed neural network and processing data. An Artificial neuron is conceived as a model of biological elements (neurons). There are many References on neural nets field [25,26].

Figure 1, illustrates a widely used structure consisting of a external input and one hidden layer of the identity activation type which are completely interconnected to a single linear output neuron. It is worth mentioning that there can be any number of limited edition series of nodes per layer and there is exactly one hidden layer to pass through before reaching the output layer. It is needed to say that if an appropriate number of nodes and layers could be chosen during optimizing, neural network would be implemented appreciatively. In the diagram shown, since the signals are passed through the layers of the neural network in a single direction, this diagram is called a feed-forward back-propagation network. Here, output unit can travel into the opposite direction to input signals. In this paper, we introduced an additional adaptive multi-layer neural network and its significant qualities. In the following, the designed architecture will be considered which is capable of approximating solution of the mentioned nonlinear math problem, to any desired degree of accuracy. As the model is running, training patterns from each input signal are weighted and then introduced into the hidden layer, where they are summed and passed through given activation functions. By computing the resulting values are weighted and summed to be the networks output. Assume that the parameters v_i, w_i (for $i = 1,..,n$) and b denote the corresponding connection weights and bias term, respectively. According to the above, the relations in each neuron of proposed network structure can be expressed as follows:

• *Input unit:*

$$o^1 = x. \tag{1}$$

• *Hidden units:*

$$o_i^2 = f(net_i), \tag{2}$$

$$net_i = x.v_i,\ i = 1,...,n.$$

• *Output unit:*

$$N(x) = \sum_{i=1}^{n} (o_i^2.w_i) + b. \tag{3}$$

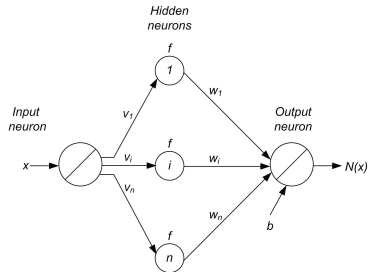

Fig. 1. Block diagram of the represented neural architecture.

The most important issue is that is that the designed neural network will not be able to perform

the given target by itself. Therefore, we must adopt a procedure that given the network parameters to be adjusted in order to achieve the ultimate goal. There are many effective ways which help us to achieve this goal. One of the famous methods is the back-propagation learning algorithm. The mentioned learning algorithm was fully introduced in 1986 by Rumelhart and Mc-Clelland [30]. The concept of forward propagation is that the given artificial neurons are located in network layers, and then generate its signal output to forward direction. Back-propagation which is a supervised algorithm means that the errors back must be propagated to adjust the weighs and bias parameters. For this aim, the calculated output is compared with the desired one, and then network error can be calculated via a suitable rule easily. In this algorithm it is assumed that the network parameters have been randomly quantified with small real valued numbers. Now, at each learning step the network output is calculated and it looks for the minimum of the error function. To the better approximation, a combination of introduced neural network mathematical modeling technique with a modification of power series method will be described in the following parts of paper.

2.2. *Implementation of the method*

Increasing use of fractional problems modeled by integro-differential equations (IDEs), has led to the development of new research on these mathematical problems. As previously indicated, many different algorithms are available to the numerical solutions of differential equations of fractional order. Due to the power series method's computation productivity and less storage requirements, it seems to be the most commonly used one in modeling fractional problems. In the work mentioned here, we focus on the numerical solution of mentioned initial value problem via a combination of neural nets approach and power-series method.

Let us recall notations and some general concepts regarding fractional calculus that are significant in this study [1,2,3,4,5,6].

Definition. If $u(x)$ is continuously differentiable function on interval $[a,b]$ up to order k. Then the Caputo derivative $D^{\alpha}_{a,x}$ and fractional integral operator $I^{\alpha}_{a,x}$ of order $\alpha > 0$ are defined by [1,2,3,4,5,6]:

$$D^{\alpha}_{a,x}[u(x)] = \begin{cases} \frac{d^k u(x)}{dx^k}, \\ \frac{1}{\Gamma(k-\alpha)} \int_a^x \frac{u^{(k)}(\tau)}{(x-\tau)^{\alpha-k+1}} d\tau, \ x > a, \end{cases} \tag{4}$$

where $\alpha = k \in N$ and $0 \leqslant k-1 < \alpha < k$.

$$I^{\alpha}_{a,x}[u(x)] = \frac{1}{\Gamma(\alpha)} \int_a^x \frac{u(\tau)}{(x-\tau)^{1-\alpha}} d\tau, \tag{5}$$

respectively. Where $\Gamma(.)$ denotes the Gamma function. Recall that for the Caputo sense, derivative of a constant is zero, and the following useful properties hold:

$$D^{\alpha}_{0,x}[x^k] = \begin{cases} 0, & k \in Z^+, \ k < \lceil \alpha \rceil, \\ \frac{\Gamma(k+1)}{\Gamma(k+1-\alpha)} x^{k-\alpha}, \ x > c, & k \in Z^+, \ k \geqslant \lceil \alpha \rceil, \end{cases} \tag{6}$$

$$I^{\alpha}_{0,x}[t^k] = \frac{\Gamma(k+1)}{\Gamma(k+1+\alpha)} x^{k+\alpha}, \ k \in Z^+. \tag{7}$$

In the above relations, notation $\lceil \alpha \rceil$ is denoted the smallest integer greater than or equal to constant α [1,2,3,4,5,6]. Now, we intend to solve the following linear Volterra integro-differential equation of fractional order:

$$P(x) D^{\alpha_1}_{0,x}[u(x)] + Q(x) I^{\alpha_2}_{0,x}[x^{\lambda_1} t^{\lambda_2} u(t)] = R(x), \tag{8}$$

$$0 < \alpha_1, \alpha_2 \leqslant 1, \tag{9}$$

subject to initial condition

$$u(0) = \beta,$$

where P, Q and R are given real-valued analytic functions on open set $(0,T)$. This mentioned equation is one of the simplest forms of fractional integro-differential equations, which is important in modelling real world phenomena. Sufficient and necessary conditions and theorems for existence and uniqueness solutions of present differential problem

(8) can be found in [31,32]. In general, the particular solution u is not easy to be found, therefore the unknown to be approximated using a suitable alternate method.

2.2.1. *Discretization of the problem*

Often, finding an analytical technique really is pretty impossible to determine the exact solutions of applied mathematical problems. The power series solution method has been traditionally developed by several authors, as an approximate but theoretically acceptable method to solve most variety of differential equations. In other words, capacity of power series to introduce any analytical function with algebraic series is coming up with the idea of developing approximate solution. The method considered here aims to produce a specific solution with a series of unknown coefficients. Ones, the power series polynomial is substituted in given differential problem, a system of equations is provided in linear or non-linear case. Now, ANNs approach is employed to obtain the initial or boundary value based unknown coefficients. This procedure will lead us to approximate unknown functions for the solution coefficients. Note that existence, particularity and smoothness of the solution are well defined.
The proposed combination method first consists of substituting solution in the decomposition form given by:

$$u(x) = \sum_{j=0}^{\infty} a_j x^j. \tag{10}$$

It should be considered that the proposed test function (10), must be satisfied in the corresponding initial conditions. In order to subdue this imperfection, the series solution is modified by:

$$u_{trial}(x) = \beta + \sum_{j=1}^{\infty} a_j x^j. \tag{11}$$

It is reasonable to consider that the designed feed-forward neural net is fully associated with the first $(n+1)$ terms of series (11). The offered architecture satisfies by construction the given initial condition, and also contains unknown coefficient in which to be determined. Assuming that $u_{trial}(x)$ symbolizes the trial solution with adjustable parameters a_j, the problem is transformed from the original construction to an unconstrained one by direct substitution (11) in (8). Thus, we have:

$$\begin{aligned} &P(x) D_{0,x}^{\alpha_1}[\beta + \sum_{j=1}^{\infty} a_j x^j] \\ &+ Q(x) I_{0,x}^{\alpha_2}[x^{\lambda_1}(\beta t^{\lambda_2} + \sum_{j=1}^{\infty} a_j t^{j+\lambda_2})] = R(x), \end{aligned} \tag{12}$$

$$\begin{aligned} &\Rightarrow P(x) \sum_{j=1}^{\infty} \zeta_j x^{j-\alpha_1} \\ &+ x^{\lambda_1} Q(x)[\phi x^{\lambda_2+\alpha_2} + \sum_{j=1}^{\infty} \xi_j x^{j+\lambda_2+\alpha_2}] = R(x), \\ &0 < \alpha_1, \alpha_2 \leqslant 1, \end{aligned} \tag{13}$$

where

$$\begin{aligned} \zeta_j &= \frac{\Gamma(j+1)}{\Gamma(j+1-\alpha_1)} a_j,\ \phi = \frac{\beta\Gamma(\lambda_2+1)}{\Gamma(\lambda_2+1+\alpha_2)}, \\ \xi_j &= \frac{\Gamma(j+\lambda_2+1)}{\Gamma(j+\lambda_2+\alpha_2+1)} a_j. \end{aligned} \tag{14}$$

Now, a set of acceptable node points must be defined for the discretization of resulting equation. Let an uniform partition on domain $[0,T]$ with the node points $x_r = \frac{rT}{n'}$ (for $r = 0,...,n'$) (for positive integer n'). After truncating the power series (11) with $n+1$ power series terms, and then placing the collocation point x_r into the Eq. (12), the origin problem will be transformed into the following form:

$$\begin{aligned} &P(x_r) \sum_{j=1}^{\infty} \zeta_j x_r^{j-\alpha_1} \\ &+ x_r^{\lambda_1} Q(x_r)[\phi x_r^{\lambda_2+\alpha_2} + \sum_{j=1}^{\infty} \xi_j x_r^{j+\lambda_2+\alpha_2}] = R(x_r). \end{aligned} \tag{15}$$

In the following part, we will review the iterative technique which help us to solve the resultant system for unknown coefficients.

2.2.2. *Proposed error function*

The conventional way in artificial neural networks approach is to start with an initial neural architecture with untrained network parameters (weights and biases), presentation training patterns to calculate the corresponding neural outputs and then employing an error correlation rule to update the parameters at every stage of repetition. The error correlation rule is a suitable combination of a given criterion function of the weights and bias parameters and an efficient learning algorithm that minimizes the criterion function for the set of connection weights and biases. In other words, the network parameters are adjusted to reduce the network error. The least mean square (LMS) output error which is a quadratic error correcting rule, represents the most commonly used criterion function [33]. Here, one starts with the indicated criterion function as follows:

$$E_r = \frac{1}{2}(P(x_r)\sum_{j=1}^{n}\zeta_j x_r^{j-\alpha_1} + x_r^{\lambda_1}Q(x_r)[\phi x_r^{\lambda_2+\alpha_2}] + \sum_{j=1}^{n}\xi_j x_r^{j+\lambda_2+\alpha_2} - R(x_r))^2, \quad (16)$$

where for simplifying, the above mathematical symbols are defined with:

$$\zeta_j = \frac{\Gamma(j+1)}{\Gamma(j+1-\alpha_1)}a_j,\ \xi_j = \frac{\Gamma(j+1)}{\Gamma(j+1+\alpha_2)}a_j,$$

$$v_i = x_r^{i-1},\ w_i = a_i \text{ and } b = \beta,\ r = 0,...,n'.$$

Then the total error is obtained by summing the predetermined error function over all the collocation points, as:

$$E = \sum_{r=0}^{n'} E_r.$$

Eventually, an attempt is made to the adjustment of network parameters through an efficient learning algorithm. In other words, we intend to employ the training rules for minimizing the network error by adaptively updating the network parameters. In order to achieve this goal, one possible error correction rule as the steepest gradient descent based back-propagation algorithm must be essentially used to optimize the weight terms. Further details in this respect can be found in [34].

2.2.3. *Proposed learning algorithm*

Learning in neural networks is choosing the appropriate network parameters where the total network error is minimized by using each training pattern. Now, the well-known back-propagation algorithm lies in and calculates to detect how much the network error depends on the input signals, network output and connection weights. To learn, first the input signals are quantified by arbitrary initial guesses, and then neural network will calculate the output for each training set. Then, the defined LMS error function is employed to train the proposed network. So, we use an optimization technique which in turn requires the computation of the gradient of the error with respect to the net parameters. finally, the supervised back-propagation learning algorithm is used for adjusting the parameters such that the network error to be minimized with respect to the input signals. The performance of this algorithm is well summarized in the following paragraph.
Now, the initial network parameters a_j (for $j = 1,...,n$) are chosen randomly to begin the learning process. The weight change for any hidden layer parameter a_j can thus be written as:

$$a_j(t+1) = a_j(t) + \Delta a_j(t),\ \ j = 1,...,n, \quad (17)$$

$$\Delta a_j(t) = -\eta.\frac{\partial E_r}{\partial a_j} + \gamma.\Delta a_j(t-1),\ r = 0,...,n',$$

where η and γ are the small valued constant learning rate and the momentum term constant, respectively. In the above relation, the symbol t in $a_j(t)$ returns to the index of iteration number and the subscript j in a_j is the label of the training weight parameter. In addition, $a_j(t+1)$ and $a_j(t)$ symbolize the updated and current weight values, respectively. It is

interesting to note here, the learning rate η utilizes a scaling factor in order to establish an adjustment to the old weight. Given the variation in the training set is not large, the neural network may learn as fast as possible, if the factor has been set with a large value. Generally, the idea to find out the better learning is to set the factor to a small value initially and then increase it. On the other hand, one of the most impressive factors in artificial neural networks is the momentum constant. The momentum factor γ basically provides some changes to the weights to persist for a number of adjustment cycles. Using the chain rule for derivatives, the above partial derivative can be expanded as:

$$\frac{\partial E_r}{\partial a_j} = (P(x_r)\sum_{j=1}^{n}\zeta_j x_r^{j-\alpha_1} + x_r^{\lambda_1}Q(x_r))[\phi x_r^{\lambda_2+\alpha_2} + \sum_{j=1}^{n}\xi_j x_r^{j+\lambda_2+\alpha_2} - R(x_r)]\times \tag{18}$$

$$(P(x_r)\frac{\partial \zeta_j}{\partial a_j}x_r^{j-\alpha_1} + x_r^{\lambda_1}Q(x_r)[\frac{\partial \xi_j}{\partial a_j}x_r^{j+\lambda_2+\alpha_2}),$$

where

$$\frac{\partial \zeta_j}{\partial a_r} = \frac{\Gamma(j+1)}{\Gamma(j+1-\alpha_1)}, \frac{\partial \xi_j}{\partial a_r} = \frac{\Gamma(j+1)}{\Gamma(j+1+\alpha_2)}, \quad r = 0,...,n'; j = 1,...,n. \tag{19}$$

It is clear that existing a mathematical software with high quality will be necessary to omit wasting time and enhance the accuracy of complicated calculations. In this study, the presented numerical examples have been tested with mathematical programing software Matlab *v*7.10.

3. Numerical Simulation and Discussion

To illustrate the ability of the method outlined in the previous section, it was applied to the two fractional order initial value integro-differential equations. Also, in order to confirm whether the mentioned iterative technique leads to higher accuracy, the test cases are compared in simulation with the samples provided in [35]. Below, we use the formal specifications $\eta = 0.03$ and $\gamma = 0.05$.

Example 3.1. Consider the following fractional order linear Volterra integro-differential equation:

$$\Gamma(\frac{13}{4})\sqrt[4]{x^3}\, D_{0,x}^{0.75}[u(x)] + \Gamma(\frac{9}{2})\sqrt[2]{x}\, I_{0,x}^{0.5}[u(t)] = 6(x^4+x^3) + \Gamma(\frac{9}{2})x,\ 0 \leqslant x \leqslant 1, \tag{20}$$

with initial condition $u(0) = 1$ and exact solution $u(x) = x^3 + 1$. The mein aim is to approximate the solution function by using the offered power series (11). Here, the regular domain-partitioning technique is used to construct training patterns on interval domain $[0,1]$. For the sake of simplicity, all of the used math symbols are listed below:

Variable	Description
n	The order of power-series polynomial
n'	The number of collocation points
t	The index of iterations
x_r	The training locations
γ	Momentum constant
r	The training index
η	Learning rate
E	Total error

First, the initial output-layer connection weights a_i (for $i = 1,...,n$) are quantified with small random values, which have been chosen randomly on interval $[0,1]$. Then the training patterns are used to successively adjust the connection weights until a suitable solution is found. To illustrate the more efficiency and to have the highest view of this method, the indicated root mean square error is shown in Table 1. It can be seen that by increasing the learning steps, total error decreases. The error function has been plotted in Figure 2 for $t = 300$ and $n' = 3$.

The exact and approximate solutions are plotted in Figures 3–6 for different number of network parameters. Also, the absolute error between approximate and exact solutions are plotted in Figures 7–10 for different number of network parameters.

Moreover, the relationship between the increasing of output neurons, training nodes and the performance analyzing of introduced ANN is given in Figure 11.

Table 1. Measured total network errors for different number of network parameters.

t	$n=3$, $n'=10$	$n=3$, $n'=15$	$n=3$, $n'=20$
500	8.7512×10^{-8}	7.0881×10^{-8}	5.9311×10^{-8}
1000	6.2704×10^{-8}	2.3801×10^{-8}	2.1466×10^{-8}
2500	3.4298×10^{-8}	2.1701×10^{-8}	1.9033×10^{-8}
5000	2.6809×10^{-8}	1.8378×10^{-8}	1.2571×10^{-8}
t	$n=5$, $n'=10$	$n=5$, $n'=15$	$n=5$, $n'=20$
500	5.4431×10^{-8}	4.8669×10^{-8}	4.0364×10^{-8}
1000	3.9018×10^{-8}	3.7442×10^{-8}	2.3472×10^{-8}
2500	3.2436×10^{-8}	3.3325×10^{-8}	2.0022×10^{-8}
5000	2.6972×10^{-8}	2.0172×10^{-8}	8.7112×10^{-9}
t	$n=7$, $n'=10$	$n=7$, $n'=15$	$n=7$, $n'=20$
500	4.2369×10^{-8}	3.5371×10^{-8}	3.1022×10^{-8}
1000	2.4365×10^{-8}	2.1914×10^{-8}	1.8439×10^{-8}
2500	1.7230×10^{-8}	1.1920×10^{-8}	8.6610×10^{-9}
5000	1.1301×10^{-8}	9.0748×10^{-9}	7.2301×10^{-9}
t	$n=9$, $n'=10$	$n=9$, $n'=15$	$n=9$, $n'=20$
500	3.4383×10^{-8}	2.1633×10^{-8}	1.9175×10^{-8}
1000	6.4571×10^{-9}	3.2133×10^{-9}	2.2344×10^{-9}
2500	4.1745×10^{-9}	2.8889×10^{-9}	1.4580×10^{-9}
5000	3.8311×10^{-9}	2.0716×10^{-9}	1.2172×10^{-9}

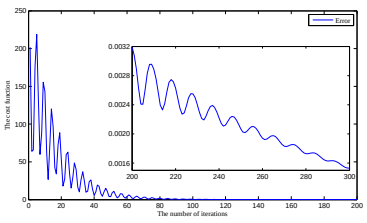

Fig. 2. Criterion function on the number of iterations.

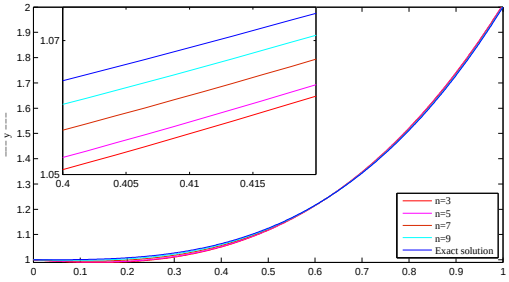

Fig. 3. Exact and approximate solutions for t=500.

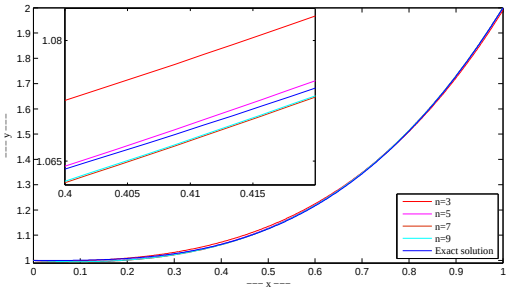

Fig. 4. Exact and approximate solutions for t=1000.

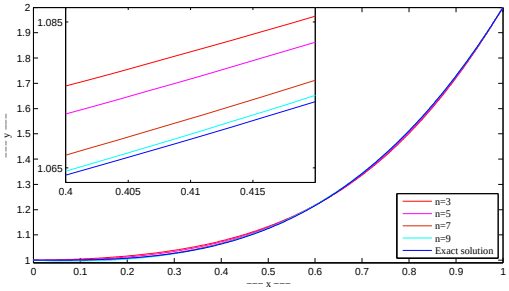

Fig. 5. Exact and approximate solutions for t=2500.

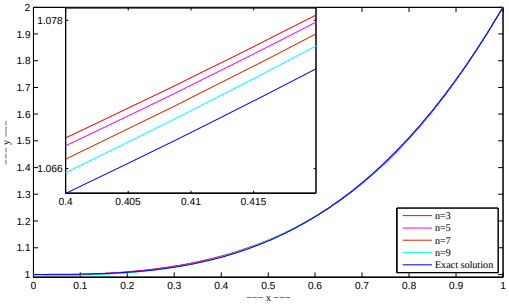

Fig. 6. Exact and approximate solutions for t=5000.

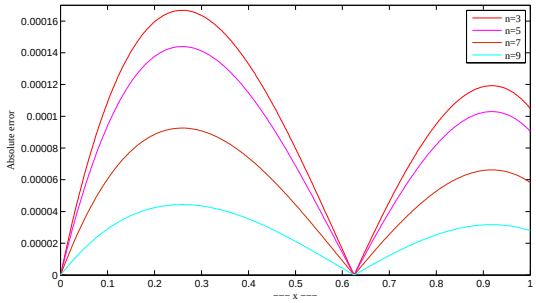

Fig. 7. $|u(x) - u_{trial}(x)|$ for t=500.

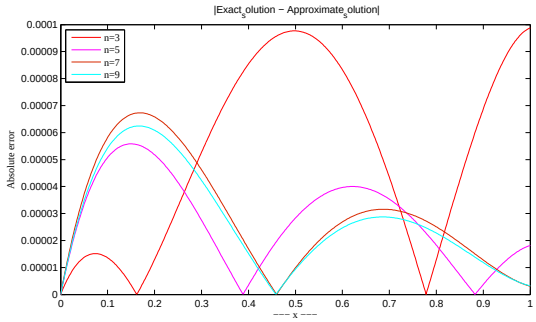

Fig. 8. $|u(x) - u_{trial}(x)|$ for t=1000.

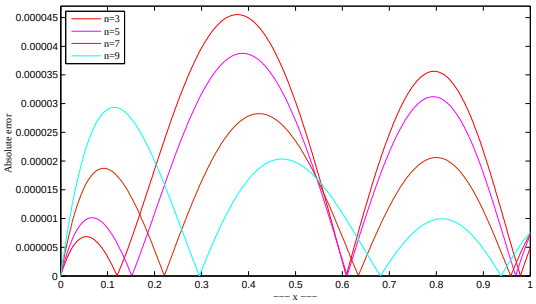

Fig. 9. $|u(x) - u_{trial}(x)|$ for t=2500.

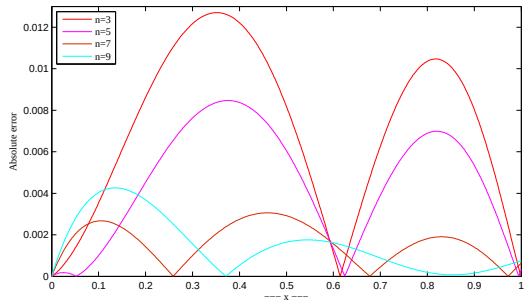

Fig. 10. $|u(x) - u_{trial}(x)|$ for t=5000.

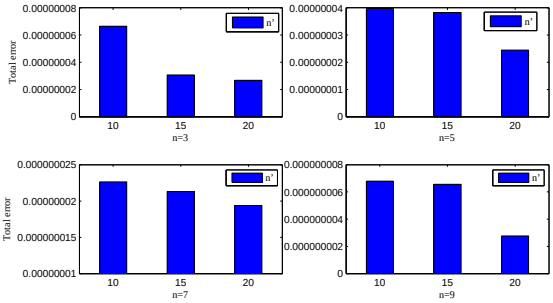

Fig. 11. The performance of proposed network architecture over different neural elements for $t = 5000$.

Example 3.2. As the second example, consider the following initial value fractional problem:

$$D_{0,x}^{0.5}[u(x)] - I_{0,x}^{1}[xtu(t)] = R(x),\ u(0) = 1,\ 0 \leqslant x \leqslant 1,$$

where

$$R(x) = -\frac{\Gamma(\frac{5}{2})}{\Gamma(2)}x - \frac{x^3}{2} + (\frac{\Gamma(5)}{\Gamma(\frac{9}{2})} + \frac{2}{7})x^{\frac{7}{2}} - \frac{x^9}{6},$$

with the exact solution $u(x) = 1 - x^{\frac{3}{2}} + x^4$. Here, we intend to compare the numerical results with

Table 2. Obtained numerical results for $t = 5000$.

n'	*ANN*		
	$n = 3$	$n = 5$	$n = 7$
20	0.9813×10^{-8}	0.7101×10^{-09}	0.4738×10^{-10}
40	0.9225×10^{-8}	0.6967×10^{-09}	0.4141×10^{-10}
80	0.5140×10^{-8}	0.5838×10^{-09}	0.3944×10^{-10}
160	0.2780×10^{-8}	0.4417×10^{-09}	0.1919×10^{-10}
320	0.2184×10^{-8}	0.8660×10^{-10}	0.1133×10^{-10}

n'	*ANN*	*HC*
	$n = 9$	
20	0.6238×10^{-11}	0.2878×10^{-5}
40	0.4638×10^{-11}	0.8731×10^{-6}
80	0.3170×10^{-11}	0.3952×10^{-6}
160	0.8719×10^{-12}	0.1706×10^{-6}
320	0.4815×10^{-12}	0.8344×10^{-7}

the ones obtained from the hybrid collocation (HC) method [35]. The maximum absolute errors ($\|.\|_\infty$) between approximate and exact solutions are numerically compared in Table 2. Similarly, the computer simulations are presented in Figures 12–13.

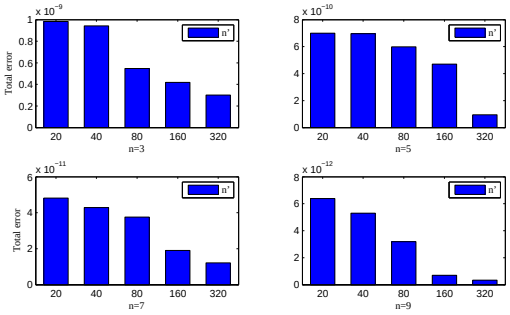

Fig. 12. The performance of proposed network architecture over different neural elements for $t = 5000$.

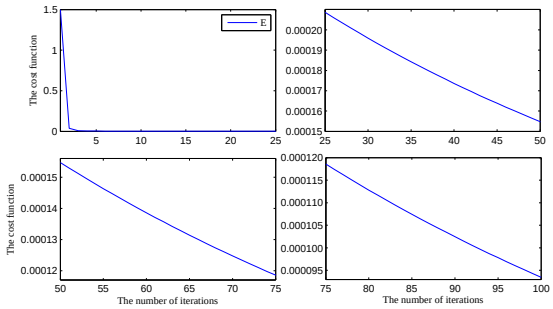

Fig. 13. The cost curve for $n = 3$, $n' = 320$ and $t = 100$.

4. Conclusion

Through the use of power series method and neural networks approach, a combination iterative technique has been derived for finding solution of an initial value linear Volterra integro-differential problem involving fractional order. The process begins with the assumption that the unknown function can be expressed as a power series polynomial. In this regard, the mentioned fractional problem is reduced to solve a linear algebraic equations system. Then, a modification of ANNs approach is employed to the comparative study of achieved system. The suggested technique was applied to study two test cases of the problem. The obtained results showed that the employed neural architecture is powerful mathematical tool for solving mentioned type integro-differential equation. Also, we compared the obtained results with both their exact solutions and those of another numerical method. The given numerical examples support our claim that the employed neural architecture gives rapidly convergent approximation without any restrictive assumptions. As areas for future research, extension of this method can be extended to cover nonlinear fractional integro-differential equations.

References

1. A.A. Kilbas , H.H. Srivastava ,J.J. Trujillo , *Theory and Applications of Fractional Differential Equations*, (Elsevier, The Netherlands, 2006).
2. I. Podlubny , *Fractional differential equations*, (Academic Press, New York, 1999).
3. V. V. Uchaikin, *Fractional derivatives for physicists and engineers*, (Springer, Berlin, 2013).
4. C. Cattani, H.M. Srivastava, Xiao-J., Yang, (Eds), *Fractional dynamics*, (Walter de Gruyter GmbH Co. KG, 2016).
5. T.M. Atanackovic , S. Pilipovic, B. Stankovic, D. Zorica, *Fractional Calculus with Applications in Mechanics: From the Cell to the Ecosystem* (Wiley-ISTE, 2014).
6. D. Baleanu, K. Diethelm, E. Scalas, J.J. Trujillo, *Fractional calculus models and numerical methods, in: Series on Complexity, Nonlinearity and Chaos*, (World Scientific, Boston, 2012).
7. Hilfer, Rudolf, ed. *Applications of fractional calculus in physics*. (World Scientific, 2000).
8. D. Baleanu, A.K. Golmankhaneh, A.K. Golmankhaneh, R.R. Nigmatullin, Newtonian law with memory, *Nonlinear Dynam.*, **60(1-2)**, (2010) 81-86 .
9. A. K. Golmankhaneh, *Investigations in Dynamics: With Focus on Fractional Dynamics*, (Lap Lambert, Academic Publishing, Germany, 2012).
10. A.K. Golmankhaneh, Relativistic scalar fields for non-conservative systems, *Phys. Scripta*, 014008, 2009 (T136).
11. D. Baleanu, A.K. Golmankhaneh, R. Nigmatullin, A.K. Golmankhaneh, Fractional newtonian mechanics, *Open Phys.*, 8(1), (2010) 120-125.
12. A.K. Golmankhaneh, A. K. Golmankhaneh, D. Baleanu, M. C. Baleanu, Fractional odd-dimensional mechanics, *Adv. Diffr. Equ-ny,* 2011(1) (2011) 1-12.
13. A. K. Golmankhaneh, R. Arefi, D. Baleanu. Synchro-

nization in a nonidentical fractional order of a proposed modified *system, J. Vib. Control* **21(6)**, (2015) 1154-1161.
14. V. Lakshmikantham, S. Leela, J. Vasundhara Devi, *Theory of fractional dynamic systems*, (Cambridge University Press, Cambridge, 2009).
15. D. Baleanu, S. I. Muslih, E. M. Rabei, A. K. Golmankhaneh, A. K. Golmankhaneh, On fractional Hamiltonian systems possessing first-class constraints within Caputo derivatives, *Rom. Rep. Phys* **63(1)**, (2011) 3-8.
16. A. Jafarian, P. Ghaderi, A. K. Golmankhaneh, D. Baleanu, Analytical treatment of system of abel integral equations by homotopy analysis method, *Rom. Rep. Phys* **66(3)**, (2014) 603-611.
17. A. K. Golmankhaneh, A. K. Golmankhaneh, D. Baleanu, On nonlinear fractional Klein-Gordon equation, *Signal Proc.*, **91(3)**, (2011) 446-451.
18. W. Deng, Sh. Du, Y. Wu, High order finite difference WENO schemes for fractional differential equations, *Appl. Math. Lett.*, **26** (2013) 362-366.
19. M.M. Khader, T.S. El-Danaf, A.S. Hendy, A computational matrix method for solving systems of high order fractional differential equations, *Appl. Math. Model.*, **37**, (2013) 4035-4050.
20. J. Dascioglu, H. Yaslan, The solution of high-order nonlinear ordinary differential Eqs, *Appl. Math. Comput.*, **217**, (2011) 56585666.
21. X. Zhang, L. Liu, Y. Wu, The eigenvalue problem for a singular higher order fractional differential equation involving fractional derivatives, *Appl. Math. Comput.*, **218**, (2012) 8526-8536.
22. J. Cao, C. Xu, A high order schema for the numerical solution of the fractional ordinary differential equations, *J. Comput. Phys.*, **238**, (2013) 154-168.
23. V. Erturk, A. Yildirim, S. Momani, Y. Khan, The differential transform method and Pad approximants for a fractional population growth model, *Int. J. Numer. Method H.* , **22**, (2012) 791-802.
24. B. Mohammadi-Alasti, G. Rezazadeh, A. Borgheei, S. Minaei, R. Habibifar, On the mechanical behaviour of a functionally graded micro-beam subjected to a thermal moment and nonlinear electrostatic pressure, *Compos. Struct.,* **93**, (2011) 1516-1525.
25. D. Graupe, P*rinciples of artificial neural networks (2nd Edition),* (World Scientific Publishing, 2007).
26. M. Hanss, *Applied Fuzzy Arithmetic: An introduction with engineering applications,* (Springer-Verlag, Berlin, 2005).
27. G. Wang, J. Liang, Exponential stability analysis for delayed stochastic Cohen-Grossberg neural network, *Int. J. Comput. Int. Sys.* **3(1)**, (2010) 96-102
28. X. Shen, X.Z. Gao, R. Bie, Artificial Immune Networks: Models and Applications,*Int. J. Comput. Int. Sys.*, **1(2)**, (2008) 168-176.
29. A. Jafarian, R. Alizadeh, A new combinative method to the two-dimensional Bratu problem, *Neural Comput. Appl*, In press.
30. R. Fuller, *Neural fuzzy systems*, (Abo Akademi University press, 2005).
31. B. Bandyopadhyay, S. Kamal, *Stabilization and control of fractional order systems: A sliding mode approach*, (vol. 317, Springer, 2015).
32. M. Muslim, Existence and approximation of solutions to fractional differential equations, *Math. Comput. Modelling,* **49**, (2009) 1164-1172.
33. R. Rojas, *Neural networks: a systematic introduction*, (Springer-Verlag, Berlin, 1996).
34. M.H. Hassoun, *Fundamentals of artificial neural networks*, (MIT Press, 1995).
35. X. Ma, Ch. Huang, Numerical solution of fractional integro-differential equations by a hybrid collocation method, *Appl. Math. Comput.*, **219**, (2013) 6750-6760.