Neural, Parallel, and Scientific Computations 24 (2016) 369-380

AN APPLICATION OF ANNS ON POWER SERIES METHOD FOR SOLVING FRACTIONAL FREDHOLM TYPE INTEGRO-DIFFERENTIAL EQUATIONS

AHMAD JAFARIAN AND SAFA MEASOOMY NIA

Department of Mathematics,Urmia Branch
Islamic Azad University, Urmia, Iran
Department of Mathematics, Science and Research Branch, Islamic Azad University, Urmia, Iran

ABSTRACT. For the last decade, several authors demonstrated the performance of artificial neural network models over other traditional testing methods. The current research, aimed to present a global optimization technique based on combination of neural networks approach and power series method for the numerical solution of a fractional Fredholm type integro-differential equation involving the Caputo derivative. The mentioned problem to be solved approximately for the unknown series coefficients via a three-layer feed-forward neural architecture. In other words, an accurate truncated power series representation of the solution function is achieved when a suitable learning algorithm is used for the suggested neural architecture. As applications of the present iterative approach, some kinds of integro-differential equations are investigated. The achieved simulations are compared with the results obtained by some existing algorithms.

Key Words Fractional Fredholm type integro-differential equation, Generalized power series expansion, ANNs approach, Caputo fractional derivative, Approximate solution.

1. Introduction

Fractional Fredholm integro-differential equations are extensively appeared in mathematical modeling of real life problems. On the other hand, the analytical solutions of these kinds of equations can not be found easily, thus there has been growing interest in the proposition of alternative numerical methods. The most commonly used ones are, fixed point method [19, 1], Adomian decomposition method [13, 17], upper and lower solutions method [12], Chebyshev wavelet method [20], Variational iteration and Homotopy perturbation methods [14], Taylor expansion method [5]. Among these methods, the series expansion technique is more attractive and gives a closed form solution for a linear fractional integro-differential equation.

In this study, a combinative iterative procedure will be proposed for the solution of fractional Fredholm type integro-differential equations. The present technique uses a modification of the power series method for transforming the origin problem to a

Received April 3, 2016

non-linear algebraic equations system that can be solved with an usual method. There are a lot of optimization algorithms attainable to solve the resulting system. Among diversity of the theoretical studies, we employe an implementation of one-hidden-layer feed-forward neural net architecture to complete this procedure. In our recent works, different architecture of artificial neural networks (ANNs) have been widely applied to approximate solutions of various types of integral equations (see [6, 7, 8, 9, 10]). To begin the optimization process, a training set of collocation points is built by discretization of the differential interval into arbitrary length subintervals. Assuming that the equation has an unique solution, the unknown series coefficients are quantified randomly to obtain a primary approximate solution. The gradient descent based back-propagation learning algorithm is then used to achieve the unknown coefficients which were considered as network parameters. The patterns are used for training the network incrementally one by one to reduce the output error. At each learning stage, the network parameters are updated and the global network error is reduced to within the indicated tolerance. The organization of this paper proceeds as follows. In the following section, we first review some basic concepts and definitions of neural networks and then extend in detail the ANNs approach for solving the mentioned type integro-differential equation. Some numerical examples with comparison to some offered algorithms are given in section 4. Obtained simulative experimental results show that the approach has the potentiality to become an effective method. The final section contains conclusions and directions for future research.

2. **Illustration of the method**

In this section, we will mainly concentrate on the solution of both linear and nonlinear fractional Fredholm type integro-differential equations (F-FIDEs). As we know, all of the methods discussed in the last section are also valid, but we are interested in the proposition of a general scheme in which can be easily implemented for various classes of fractional equations. For this aim, the approximate solution is represented by means of a modification of the so called power series method, whose coefficients are estimated by training an appropriate neural network architecture. This work yields a truncated power series representation of the solution function, which usually is enough for obtaining approximation to it. We start by presenting some commonly used fundamental concepts.

2.1. **A brief introduction to ANNs.**

In this part we deal with essential concepts in which will be used further on. It is not an exaggeration to say that research in the field of artificial neural networks has been growing attention in the last ten years than ever before. First of all, we make a cursory review to neural network surrogate

modeling for the numerical solution of a given problem. For more information in this issue, refer to [3, 4].

Here, we focus on a simple neural network architecture in which can be easily transformed to get an efficient iterative scheme for estimating solution of the mentioned fractional order Fredholm integro-differential equation. The proposed three-layer feed-forward neural network framework shown in Figure 1, contains one external input, n hidden neurons and an output signal. This architecture shows how a power series expansion can be performed as a neural net. Let v and w denote the $1 \times n$ weight vectors, and also b is bias term. In the present architecture, the input signal which is represented by the mathematical symbol x, when multiplied by connection weight v_i, gives a weighted input. In this case, this product is fed through the activation function f to generate a result. Net output $N(x)$ is computed by multiplying the output of neuron i in the hidden layer with the weight parameter w_i and then adding to the bias signal b. The input-output relation of each unit in the proposed neural architecture can be summarized as follows:

- *Input unit:*

$$o^1 = x. \tag{2.1}$$

- *Hidden units:*

$$o_i^2 = f(net_i), \tag{2.2}$$

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

- *Output unit:*

$$N(x) = \sum_{i=1}^{n} (o_i^2 w_i) + b. \tag{2.3}$$

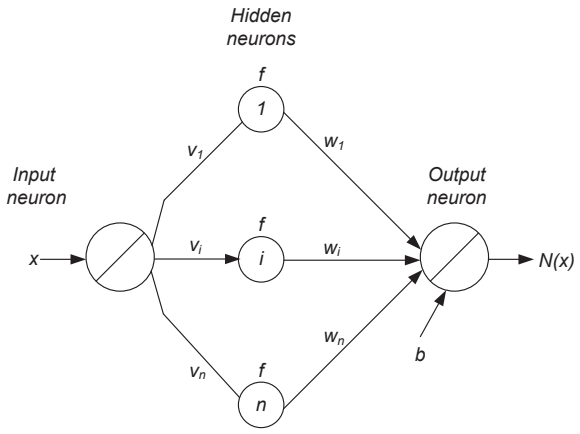

Fig. 1. *Illustration of the represented neural architecture.*

2.2. **Solving the F-FIDE problem.** In this study, we restrict our attention to the nonlinear initial value fractional integro-differential equation of the form:

$$ {}_cD_x^{\alpha}[u(x)] = \phi_1(x,u(x)) + \lambda \int_0^1 \xi(x,t)\phi_2(u(t))dt, \quad 0 \le x \le 1, \tag{2.4}$$

subject to the initial condition $u(0) = \beta$. Here, ${}_cD_x^{\alpha}$ denotes the Caputo differential operator of order $\alpha \in (0,1]$, λ is a real known parameter, $\phi_1, \phi_2 \in L^2([0,1])$ and $\xi \in L^2([0,1]^2)$ are given functions, and $u(x)$ is the unknown to be determined.

Definition 1. Let $u(x)$ is continuously differentiable function on finite interval $[a,b]$ up to order k. The Caputo fractional derivative operator ${}_cD_x^{\alpha}$ of order $\alpha > 0$ is defined by:

$$ {}_cD_x^{\alpha}[u(x)] = \begin{cases} \frac{d^k u(x)}{dx^k}, & \alpha = k \in N, \\ \frac{1}{\Gamma(k-\alpha)} \int_c^x \frac{u^{(k)}(\tau)}{(x-\tau)^{\alpha-k+1}} d\tau, \ x > c, & 0 \le k-1 < \alpha < k, \end{cases} \tag{2.5}$$

where $\Gamma(\cdot)$ denotes the Gamma function. Recall that for the Caputo sense, derivative of a constant is zero and the following useful property holds:

$$ {}_cD_x^{\alpha}[x^k] = \begin{cases} 0, & k \in N, \ k < \lceil \alpha \rceil \\ \frac{\Gamma(k+1)}{\Gamma(k+1-\alpha)} x^{k-\alpha}, \ x > c, & k \in N, k \ge \lceil \alpha \rceil \end{cases}, \tag{2.6}$$

where the ceiling function $\lceil \alpha \rceil$ denotes smallest integer greater than or equal to α. For more details and mathematical properties of fractional calculus, please refer to [18]. The reminder of this paper is organized into two segments: the solution function is represented by a truncated power series expansion, and using the purposed neural network configuration to determine the values of the series coefficients. Sufficient and necessary conditions for existence and uniqueness of solutions of the mentioned fractional problems are given in [2].

2.2.1. *Discretization of the problem.* The generalized power series expansion is a powerful tool which has been developed for the numerical solution of different kinds of fractional equations. This method decomposes the solution $u(x)$ into a rapidly convergent series of solution components. In our study, we will approximate the solution function $u(x)$ by following series:

$$ u(x) = \sum_{i=0}^{\infty} a_i x^{i\alpha}, \tag{2.7}$$

for fractional order $\alpha \in (0,1]$ (see, [11, 15]). Under initial condition, we take $a_0 = \beta$. Thus, Eq. (2.7) is written as:

$$ u(x) = \beta + \sum_{i=1}^{\infty} a_i x^{i\alpha}. \tag{2.8}$$

Now, we replace $u(x)$ with the introduced power series by substituting (2.8) into (2.6), and find its fractional derivative to find a relation among the coefficients, as:

(2.9)
$$\sum_{i=1}^{\infty} a_i \frac{\Gamma(i+1)}{\Gamma(i+1-\alpha)} x^{(i-1)\alpha} = \phi_1\left(x, \beta + \sum_{i=1}^{\infty} a_i x^{i\alpha}\right) + \lambda \int_0^1 \xi(x,t)\phi_2\left(\beta + \sum_{i=1}^{\infty} a_i t^{i\alpha}\right) dt.$$

For positive integer m, consider a partition of interval $[0,1]$ with the node points $x_j = \frac{j}{m}$, (for $j = 0, \ldots, m$). Now, let us put the collocation point x_j into the Eq. (2.9). After some simplifications and grouping, we obtain the following relation:

$$\sum_{i=1}^{\infty} a_i \frac{\Gamma(i+1)}{\Gamma(i+1-\alpha)} x_j^{(i-1)\alpha} = \phi_1\left(x_j, \beta + \sum_{i=1}^{\infty} a_i x_j^{i\alpha}\right) + \lambda \int_0^1 \xi(x_j,t)\phi_2\left(\beta + \sum_{i=1}^{\infty} a_i t^{i\alpha}\right) dt, \quad j = 0, \ldots, m. \tag{2.10}$$

Now, we look at the technique which will allow us to approximate desired values of the coefficients a_i (for $i \in N$).

2.2.2. *Proposed criterion function.* To achieve a particular goal, only the first $(n+1)$ terms of defined power series (2.8) are used. As a result, if we express the solution function as a truncated generalized power series, so it can be easily approximated by finding the free coefficients a_i, (for $i = 1, 2, \ldots, n$). This means, we intend to approximate solution $u(x)$ by the truncated series $u_n(x) = \beta + \sum_{i=1}^{n} a_i x^{i\alpha}$. Below, we use the notations $v_i = x^{i-1}$, $w_i = a_i$, $f(x) = x^{\alpha}$ and $b = \beta$. Whit these assumptions, it can easily be argued that the output of the designed neural network is equivalent to the mentioned truncated generalized power series.

In order to train this network over the space of connection weights, the parameter a_i must be changed in such a way that the network error is reduced. This error function evaluates the degree of estimate for any given set of network parameters. To implement this procedure, the well known commonly used mean squared error function is formulated after presentation of the j-th input pattern. Here, one starts with the criterion function:

$$E_j = \frac{1}{2}\left(\sum_{i=1}^{n} a_i \frac{\Gamma(i+1)}{\Gamma(i+1-\alpha)} x_j^{(i-1)\alpha} - \phi_1\left(x_j, \beta + \sum_{i=1}^{n} a_i x_j^{i\alpha}\right) - \lambda \int_0^1 \xi(x_j,t)\phi_2\left(\beta + \sum_{i=1}^{n} a_i t^{i\alpha}\right) dt\right)^2, \quad j = 0, \ldots, m. \tag{2.11}$$

Throughout next part, an attempt will be made to see how the defined error function can be minimized over the set of node points. For this aim, the weights are to be optimized via a gradient descent optimization process. This supplement, known as back error propagation in which will be considered in following. Further details in this respect can be found in [15].

2.2.3. *Proposed learning algorithm.* Before offering specific learning rule, it should be noted that learning in an artificial neural network implements a local search mechanism to obtain optimal weight values which decrease global network error. Now, we try to train the neural architecture by modifying the connecting weights according to the defined set points that can be formulated as an learning algorithm. Throughout this part, an attempt is made to construct an incremental learning process as a self learning mechanism for minimizing the predefined criterion function. Since, this function is analytical one, the gradient descent method will naturally lead to a capable learning rule. We illustrate the above idea by deriving a unsupervised back-propagation learning algorithm for adjusting the weight parameters such that the above target is minimized over the space of weight settings. The performance of this algorithm is well summarized in the following paragraph.

First, he initial weight parameters a_i (for $i = 1, \ldots, n$) are selected randomly to begin the procedure. Then, the set of node points are used to successively adjust the connection weights by moving a small step in direction in which correctly optimize the objective function. To complete the derivation of back-propagation for the output layer weights, we perform gradient descent rule on the criterion function (2.11). This standard algorithm works as follows:

$$a_i(r+1) = a_i(r) + \Delta a_i(r), \quad i = 1, \ldots, n, \tag{2.12}$$

$$\Delta a_i(r) = -\eta \cdot \frac{\partial E_j}{\partial a_i} + \gamma \cdot \Delta a_i(r-1), \tag{2.13}$$

where η and γ are the small constant learning rate and the momentum term constant, respectively. Due to its gradient descent nature, back propagation is very sensitive to the values of learning rate and momentum constant. If the choice of these initial quantities, then the learning convergence will be slow. Here, the index r in $a_i(r)$ refers to the iteration number and the subscript j in x_j is the label of the training node point. To complete the derivation of desired learning rule for the hidden layer weights, the above partial derivative is to be calculated using the Chain rule as:

$$\begin{aligned}
\frac{\partial E_j}{\partial a_i} = &\left(\sum_{i=1}^{n} a_i \frac{\Gamma(i+1)}{\Gamma(i+1-\alpha)} x_j^{(i-1)\alpha} - \phi_1 \left(x_j, \beta + \sum_{i=1}^{n} a_i x_j^{i\alpha} \right) \right. \\
&\left. - \lambda \int_0^1 \xi(x_j, t) \phi_2(\beta + \sum_{i=1}^{n} a_i t^{i\alpha}) dt \right) \\
&\times \left(\frac{\Gamma(i+1)}{\Gamma(i+1-\alpha)} x_j^{(i-1)\alpha} - x_j^{i\alpha} \phi_1' \left(x_j, \beta + \sum_{i=1}^{n} a_i x_j^{i\alpha} \right) \right. \\
&\left. - \lambda \int_0^1 t^{i\alpha} \xi(x_j, t) \phi_2'(\beta + \sum_{i=1}^{n} a_i t^{i\alpha}) dt \right).
\end{aligned}$$

Therefore, if a solution exists, it is approximated after sufficient training. Due to gradient descent nature of the method, back-propagation is very sensitive to the values of learning rate and momentum constant. If the choice of these initial quantities are inaccurate, then the learning convergence will be slow. Network learning via the whole training node points once is usually called cycling. Generally speaking, more than one cycle through the set points is needed to conclude an appropriate solution vector.

3. **Numerical examples**

In order to show the performance of the proposed neural networks approach in solving fractional integro-differential equation problems, two numerical examples are carried out in this section. A comparison is made between the proposed combinative algorithm and other methods presented in [16]. All calculations are achieved by using the mathematical software Matlab $v7.10$. Below, we use the specifications as following:

1) Learning rate: $\eta = 0.1$,
2) Momentum constant: $\gamma = 0.05$.

Also, for better comparison the root mean square error is employed as:

$$E_{mid} = \|u - u_n\| = \left(\frac{1}{m+1} \sum_{j=0}^{m} (u(x_j) - u_n(x_j))^2 \right)^{\frac{1}{2}}.$$

Example 3.1. Consider the following integro-differential equation:

$${}_cD_x^{\alpha}[u(x)] + 3u(x) = \phi_1(x, u(x)) + \frac{3\pi}{2} \int_0^1 xt^2 \sin(\pi u(t)) dt, \quad 0 \le x \le 1,$$

where

$$\phi_1(x, u(x)) = 3x^3 + \frac{\Gamma(4)}{\Gamma(\frac{7}{2})} x^{\frac{5}{2}} - x,$$

with the initial condition $u(0) = 0$ and the exact solution $u(x) = x^3$. As indicated, the solution function can be uniformly approximated on that interval by (2.8) to any degree of accuracy. The objective here is to adaptively update the hidden layer weights a_i (for $i = 1, \ldots, 10$), by attempting to optimize the associated criterion function over the set of $m+1$ training points $X = \{x_0, x_1, \ldots, x_m\}$. For this aim, the weights are first normally initialized to small random values and then the learning rule considered in previous section is employed. The root mean square errors for different number of iterations and node points are presented in Table 1. Figure 2 shows the cost functions on the different number of iterations. It is noticeable that by increasing the learning steps, the criterion function goes to zero. The approximate and exact solutions are plotted and compared in Figure 3, and also absolute error are plotted in Figure 4 for $r = 100$.

r	E_{mid}			
	$m = 10$	$m = 20$	$m = 30$	$m = 40$
100	0.0008745	0.0007344	0.0006383	0.0005617
200	0.0006194	0.0004625	0.0003832	0.0003463
300	0.0004083	0.0002940	0.0002334	0.0002172
400	0.0003140	0.0002077	0.0001835	0.0001702
500	0.0002801	0.0001756	0.0001587	0.0001427

Table 1. *Numerical results for Example 4.1.*

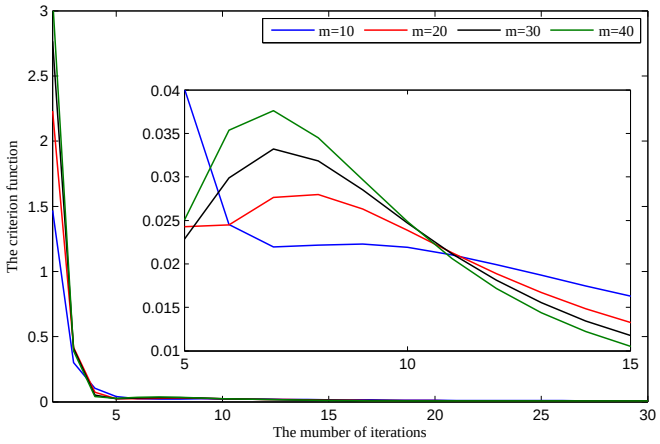

Fig. 2. *The cost curves with different step sizes for Example 4.1.*

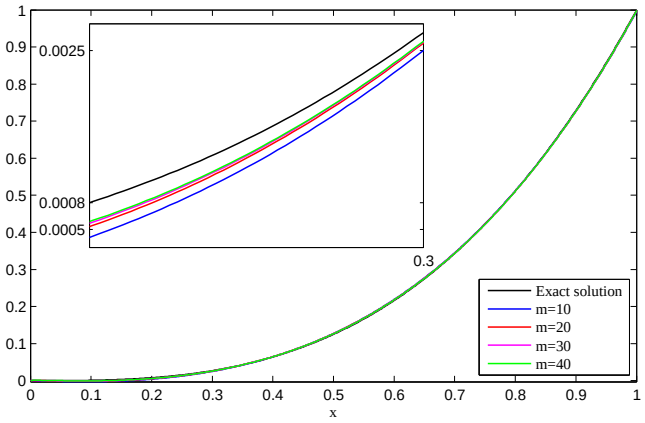

Fig. 3. *The exact and approximate solutions for Example 4.1.*

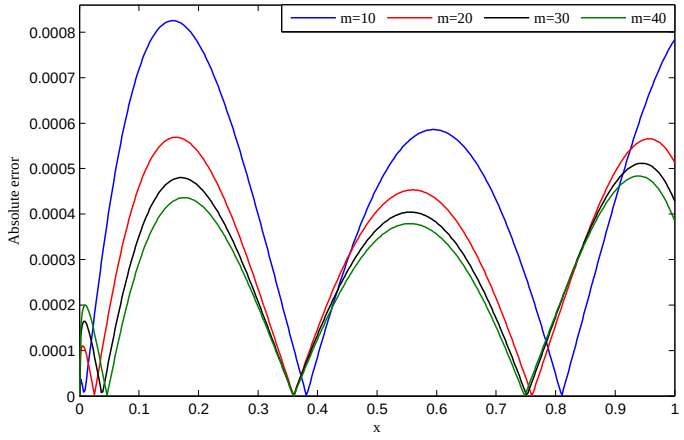

Fig. 4. *The absolute error between exact and approximate solutions for Example 4.1.*

Example 3.2. The following fractional equation is considered:

$$ {}_cD_x^{\alpha}[u(x)] = 1 - \frac{x}{4} + \int_0^1 xt[u(t)]^2 dt, \quad 0 \le x \le 1, $$

with initial condition $u(0) = 0$. Note that the exact solution of this problem for $\alpha = 1$, is $u(x) = x$. This equation is studied in [16] by using the rationalized Haar functions (RHF). The numerical results for $m = 10$ and $r = 1000$ with comparison are presented in Table 2. Based on the results, it can be concluded that our numerical simulations are in good agreement with the solutions reported in the literature. Therefore, it easily can be seen the approximate solutions for $\alpha = 0.25$, $\alpha = 0.5$ and $\alpha = 0.75$ are reliable. Figures 5 and 6 simulate the cost function and absolute error criterion for $\alpha = 1$, respectively. Since the exact solution of the above problem is available when $\alpha = 1$, so we are only able to show the criteria of mean absolute error for this value. As can be seen, the estimated solution is in high concurrence with the exact solution.

x	$\alpha = 0.25$		$\alpha = 0.5$	
	ANN	RHF	ANN	RHF
0.1	0.650960	0.650962	0.362259	0.362260
0.2	0.821677	0.821678	0.525360	0.525361
0.3	0.959522	0.959520	0.657181	0.657181
0.4	1.084528	1.084520	0.774341	0.774336
0.5	1.203268	1.203260	0.883181	0.883175
0.6	1.131835	1.131827	0.986274	0.986267
0.7	1.431320	1.431310	1.085718	1.085710
0.8	1.543181	1.543170	1.182468	1.182460
0.9	1.654432	1.654420	1.277279	1.277270
1.0	1.765433	1.765420	1.370730	1.370720

x	$\alpha = 0.75$		$\alpha = 1$	
	ANN	RHF	ANN	RHF
0.1	0.194153	0.194154	0.099997	0.099998
0.2	0.329896	0.329896	0.199994	0.199995
0.3	0.450541	0.450540	0.299990	0.299988
0.4	0.563053	0.563051	0.399979	0.399978
0.5	0.670476	0.670474	0.499968	0.499966
0.6	0.774076	0.774073	0.599954	0.599951
0.7	0.875052	0.875049	0.699936	0.699934
0.8	0.973944	0.973941	0.799912	0.799913
0.9	1.071224	1.071220	0.899893	0.899890
1.0	1.167254	1.167250	0.999867	0.999864

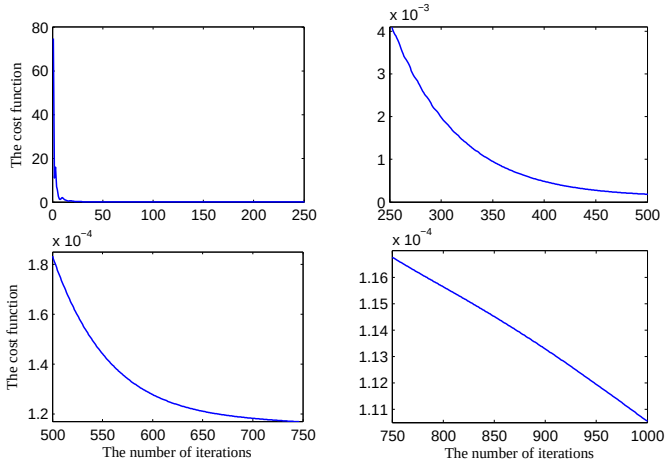

Fig. 5. *The cost curve for Example 4.2.*

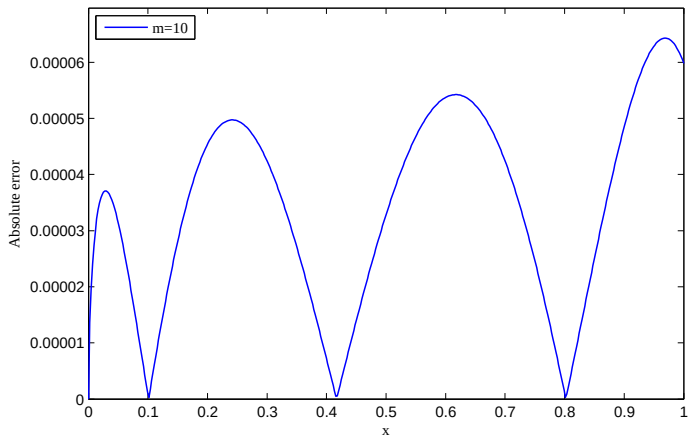

Fig. 6. *The absolute error between exact and approximate solutions for Example 4.2.*

4. Conclusion

Fractional equations have gained increasing importance due to their several applications in the field of real world problems. As indicated, these kind of equations are very difficult to handle analytically, so we have to usually get approximate solutions. This paper explained how a combination of the power series method and neural networks approach can be developed as an iterative technique for the numerical solution of a fractional Fredholm type integro-differential equation. The proposed methodology, employed a three-layer feed-forward neural network architecture for finding the power series coefficients of the solution function. From a practical point of view, tow numerical examples were investigated to demonstrate the applicability of the presented iterative method. Moreover, the obtained simulation results were compared with exact solutions and also with the solutions obtained by two another works. For the future works, we are going to develop our proposed method for solving high order integro-differential equations.

REFERENCES

[1] A. Anguraj, P. Karthikeyan, M. Rivero, J. J. Trujillo, On new existence results for fractional integro-differential equations with impulsive and integral conditions, 66(12) (2014) 2587–2594.

[2] B. Bandyopadhyay, S. Kamal, Stabilization and control of fractional order systems: A sliding mode approach, 317, 2015.

[3] D. Graupe, Principles of artificial neural networks (2nd Edition), World Scientific Publishing, 2007.

[4] M. Hanss, Applied Fuzzy Arithmetic: An introduction with engineering applications, Springer-Verlag, Berlin, 2005.

[5] L. Huang, X. F. Li, Y. L. Zhao, X. Y. Duan, Approximate solution of fractional integro-differential equations by Taylor expansion method, Comput. Math. Appl., 62(3) (2011) 1127–1134.

[6] A. Jafarian, S. Measoomy Nia, S. Abbasbandy, Artificial neural networks based modeling for solving linear Volterra integral equations system, Applied Soft Computing, 27 (2015) 391–395.

[7] A. Jafarian, S. Measoomy Nia, Artificial neural network approach to the fuzzy Abel integral equation problem, Journal of Intelligent and Fuzzy Systems, DOI:10.3233/IFS-130980.

[8] A. Jafarian, S. Measoomy Nia, New itrative method for solving linear Fredholm fuzzy integral equations of the second kined, International Journal of Industrial Mathematics, 5(3) (2013) 10 pages.

[9] A. Jafarian, S. Measoomy Nia, Feed-back neural network method for solving linear Volterra integral equations of the second kind, Int. J. Mathematical Modelling and Numerical Optimisation, 4(3) (2013) 225–237.

[10] A. Jafarian, S. Measoomy Nia, Utilizing feed-back neural network approach for solving linear Fredholm integral equations system, Applied Mathematical Modelling, 37(7) (2013) 5027–5038.

[11] G. Jumarie, Modified Riemann-Liouville derivative and fractional Taylor series of nondifferentiable functions further results, Comput. Math. Appl. 51 (2006).

[12] S. M. Momani, S. B. Hadid, Some comparison results for integro-fractional differential inequalities, J. Fract. Calc. 24 (2003) 37–44.

[13] S. Momani, M. Noor, Numerical methods for fourth order fractional integro-differential equations, Appl. Math. Comput., 182 (2006) 754–60.

[14] Y. Nawaz, Variational iteration method and homotopy perturbation method for fourth-order fractional integro-differential equations, Comput. Math. Appl., 61(8) (2011) 2330–2341.

[15] Z. Odibat, N. Shawagfeh, Generalized Taylor's formula, Appl. Math. Comput., 186(1) (2007) 286–293.

[16] Y. Ordokhani, N. Rahimi, Solving fractional nonlinear Fredholm integro-differential equations via hybrid of rationalized Haar functions, Journal of Information and Computing Science, 9(3) (2014) 169–180.

[17] S. S. Ray, Analytical solution for the space fractional diffusion equation by two-step Adomian decomposition method, Commun. Nonlinear. Sci. Numer Simulat., 14 (2009) 129–306.

[18] X. J. Yang, Advanced local fractional calculus and its applications, World Science Publisher, New York, USA, 2012.

[19] L. Zhanga, B. Ahmadb, G. Wanga, R. P. Agarwal, Nonlinear fractional integro-differential equations on unbounded domains in a Banach space, Journal of Computational and Applied Mathematics, 249 (2013) 51–56.

[20] L. Zhu, Q. Fan, Solving fractional nonlinear Fredholm integro-differential equations by the second kind Chebyshev wavelet, Commun. Nonlinear Sci. Numer. Simulat., 17 (2012) 2333–2341.