DOI: 10.25092/baunfbed.483922 *J. BAUN Inst. Sci. Technol., 20(3) Special Issue, 143-153, (2018)*

Simple recurrent neural networks for the numerical solutions of ODEs with Dirichlet boundary conditions

Korhan GÜNEL[1,*], Gülsüm İŞMAN[2], Merve KOCAKULA[2]

[1]*Aydin Adnan Menderes University, Faculty of Arts and Sciences, Department of Mathematics, Aydin.*
[2]*Aydin Adnan Menderes University, Graduate School of Natural Sciences, Department of Mathematics, Aydin.*

Geliş Tarihi (Recived Date): 16.08.2018
Kabul Tarihi (Accepted Date): 16.11.2018

Abstract

In this study, we consider Dirichlet Boundary Value Problems (DBVPs) for Ordinary Differential Equations (ODEs) to illustrate the general procedure of obtaining numerical solutions using simple Recurrent Neural Networks (RNNs). Different types of both linear and nonlinear activation functions are used in the neural network. The network is trained by Particle Swarm Optimization (PSO) method, and cross validation approach is performed to tune the arbitrary parameters of neural nets. The exact solutions and the obtained neural net solutions, regarding with the types of activation functions, are compared to determine the efficiency of using RNNs in solving the problem. In all cases, the exact solutions are confronted with those obtained from RNNs in the context of absolute errors and average mean squared errors (MSEs) with standard deviations.

Keywords: *Recurrent neural networks, particle swarm optimization, ordinary differential equations, Dirichlet boundary value problem.*

[*] Korhan GÜNEL, kgunel@adu.edu.tr, http://orcid.org/0000-0002-5260-1858
Gülsüm İŞMAN, gulsumisman@gmail.com, http://orcid.org/0000-0002-0563-5086
Merve KOCAKULA, kocakulamerve@gmail.com, http://orcid.org/0000-0001-9091-9050

Dirichlet sınır değer koşullarına sahip adi diferansiyel denklemlerin nümerik çözümleri için basit tekrarlayan sinir ağları

Özet

Bu çalışmada, basit tekrarlayan yapay sinir ağları (RNN'ler) kullanılarak nümerik çözümlerin elde edilmesine yönelik süreci genel olarak açıklamak adına, Adi Diferansiyel Denklemler (ODE) için Dirichlet Sınır Değer Problemleri (DBVP) ele alınmıştır. Yapay sinir ağında doğrusal ve doğrusal olmayan türlerde çeşitli aktivasyon fonksiyonları kullanılmıştır. Ağ, Parçacık Sürü Optimizasyonu (PSO) yöntemiyle eğitilmiştir ve ağın keyfi parametrelerinin ayarlanabilmesi için çarpaz doğrulama yaklaşımı kullanılmıştır. Problemin çözümünde RNN kullanımının etkinliğini belirlemek için, gerçek çözümler ile aktivasyon fonksiyonunun türüne bağlı olarak elde edilen sinir ağı çözümleri karşılaştırılmıştır. Tüm durumlarda gerçek çözümler ile RNN'den elde edilen sonuçlar, mutlak hatalar, ortalama karesel hataların ortalaması ve standart sapma bağlamında karşılaştırılmıştır.

Anahtar kelimeler: *Tekrarlayan sinir ağları, parçacık sürü optimizasyonu, adi diferansiyel denklemler, Dirichlet sınır değer problemi.*

1. Introduction

Description of the systems are substantial step for the solution of daily life problems, and the modelling of the complex systems is mostly come true by the means of stating differential equations. To obtain the numerical solutions of differential equations using traditional methods such as Shooting Method, Runge-Kutta based methods, Multi-step methods, and Finite Difference Method, firstly, the continuous domain is discretized by welcoming some cumulative errors. Furthermore, the numerical solutions are available only at discretization nodes in the problem domain.

Neural Networks (NNs) have been introduced as an alternative approach to overcome these bottlenecks [1-5]. NNs are superior to classical numerical methods by means of training with the discretization nodes and providing the approximate solutions at any point of continuous search space. NNs are mostly trained by derivative based optimization methods such as Gradient Descent, Scaled Conjugate Gradient and Levenberg-Marquardt optimizers. However, the training NNs can occur by dissimilar ways such as derivative free or heuristic optimization methods. With this direction, some of the works that can be considered milestone research in literature is summarized in the following.

The first study is presented by Lee and Kang (1990) including utilizing Hopfield Neural Network models for solving the finite difference equation [1]. Meade and Fernandez (1994) demonstrate that Feedforward Neural Network (FFNN) is able to solve linear ODEs [2]. Lagaris et al. (1998) solve initial and boundary value problems, which has a trial solution, including two parts, which satisfies conditions with ANNs solution [3].

Malek and Beidokthi present a hybrid method based on optimization techniques and ANNs so as to solve both first and high order ODEs [4]. Raja improve stochastic computational methods as ANNs and optimization methods such as Simulated Annealing (SA), Pattern Search (PS), Genetic Algorithms (GAs), Active-set Algorithm (ASA) and their hybrid methods in order to solve Boundary Value Problems (BVPs) of second order Pantography Functional Differential Equations (PFDEs) [5]. Raja et al. develop stochastic techniques for the solution of 2-dimensional Bratu problem with Feedforward Neural Networks [6]. For the network training, they utilize global technique as Particle Swarm Optimization (PSO) and to get faster convergence they use Sequential Quadratic Programming (SQP) and their hybrid approaches. Raja presents a study about the numerical treatment for the Troesch's problem [7]. For this purpose, he utilizes NNs optimized with optimization techniques as PSO, ASA and their hybrid methods. Raja et al. propose a computational intelligence method based on NNs and SQP for the solution of fractional order nonlinear Riccati Differential Equations [8].

Apart from of these studies, we put into practice one of the derivative-free population based global optimization method as Particle Swarm Optimization (PSO) to train a Recurrent Neural Network (RNN) for solving Dirichlet Boundary Problems of Ordinary Differential Equations, in this work. In contrast to studies in the literature, the main reason for using RNN in this study is that RNNs achieve more accurate results when making predictions with nonlinear chaotic time series as emphasied in the study of Brezak et al. (2012) and the study of Saini, Parkhe and Khadtare (2016) [9,10].

In the sequel, we proceed to summarize briefly the mathematical model of the problem utilized in this work for obtaining the Recurrent Neural Network solution of a Dirichlet Boundary Problem. The next section clarifies how to transform a DBVP to an optimization problem. The third section covers the limitations of this study and the experimental studies to compare the mentioned methods applied to some different types of second order ODEs with Dirichlet Boundary Conditions. The final section presents the findings of the study and some future works with the conclusion we reach.

2. Mathematical Modeling

In this section, we describe how to transform DBVPs for ODEs to an optimization problem along with the cost function depending on a Recurrent Neural Network solution. Let us consider the problem given in Eq. (1) in which the function f is continuous in $[a,b]$, so the ODE has a unique solution with the boundary conditions.

$$\begin{cases} y''(x) = f(x, y(x), y'(x)), \quad x \in (a,b) \\ y(a) = A, \\ y(b) = B \end{cases} \tag{1}$$

The trial function as formed in Eq. (2) can be used to solve the given problem in Eq. (1). One can easily seen that the trial function satisfies the boundary conditions of Eq. (1).

$$y_T(x_j, \vec{p}) = \frac{(x_j - b)}{a-b} A - \frac{(x_j - a)}{a-b} B + (x_j - a)(x_j - b)\mathrm{Net}(x_j, \vec{p}) \tag{2}$$

where the function Net depending on the arbitrary neural net parameters $\vec{p} = [\vec{\alpha}, \vec{\omega}, \vec{\Omega}, \vec{\beta}]$ as given in Eq. (3) and the input x_j denotes the simple RNN solution of Eq. (1). The inputs of RNN x_j for $j = 1, 2, \ldots, N$ are determined by discretization of the interval $[a, b]$ where N is the total number of discretization nodes. The dicretization nodes are only used to train the RNN whose architecture is given as in Fig. 1.

$$\text{Net}(x_j(t), \vec{p}) = \sum_{i=1}^{m} \alpha_i g(z_i) \tag{3}$$

where m is the total number of neurons in RNN depicted in Fig. 1, and the output of the i th neuron is $z_i(t) = \omega_i x_j(t) + \Omega_i z_j(t-1) + \beta_i$ at iteration t for $i = 1, 2, \ldots, m.$ In this study, the activation function g in Eq. (3) is selected among of hyperbolic tangent, Rectified Linear Unit (ReLU), Parametric ReLU (PreLU), Leaky ReLU, Exponential Linear Unit (ELU) and Self-Gated Activation Function (SWISH) functions used commonly in RNNs to avoid the vanishing gradient problem. The definition of the aforementioned activation functions are listed in Table 1.

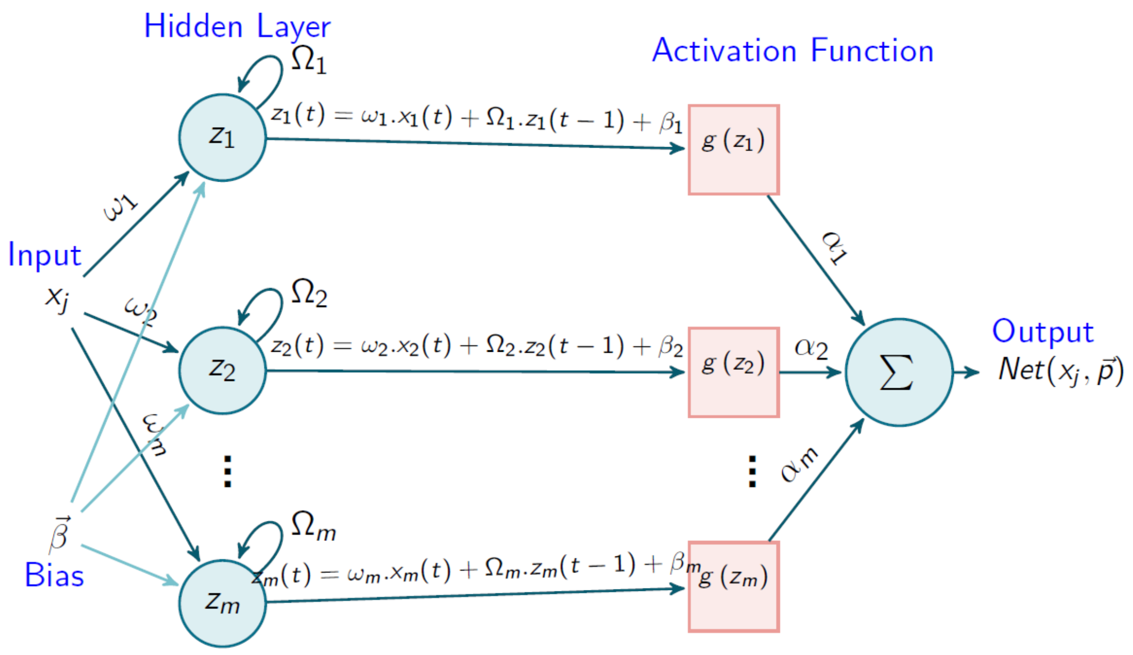

Figure 1. Architecture of simple RNN.

Table 1. The list of activation function used in this study.

Activation Function	**Definition**
tanh	$g(z)=\dfrac{e^{2z}-1}{e^{2z}+1}$
ReLU	$g(z)=\max(0,z)$
Leaky ReLU	$g(z)=\begin{cases} z, & \text{if } z>0 \\ 0.01z, & \text{otherwise} \end{cases}$
PReLU	$g(z)=\begin{cases} z, & \text{if } z>0 \\ az, & \text{otherwise} \end{cases}$ where $a\in(0,1)$
ELU	$g(z)=\begin{cases} z, & \text{if } x>0 \\ a\left(e^{z}-1\right), & \text{otherwise} \end{cases}$ where $a\geq 0$
SWISH	$g(z)=z.\sigma(z)=z.\dfrac{1}{1+e^{-z}}$

If unknown parameters of recurrent neural network given in Fig. 1 are specified as a cost function which is given in Eq. (4) our problem turns into an optimization problem.

$$E=\frac{1}{N}\sum_{j=1}^{N}\left\{\frac{\partial^2 y_T(x_j,\vec{p})}{\partial x_j^2}-f\left(x_j,y_T(x_j,\vec{p}),\frac{\partial y_T(x_j,\vec{p})}{\partial x_j}\right)\right\}^2 \tag{4}$$

First and second derivatives of the trial function in Eq. (4) are required to calculate the cost function, and they can be defined as given in Eq. (5) and Eq. (6) respectively.

$$\begin{aligned}\frac{\partial y_T(x_j(t),\vec{p})}{\partial x_j(t)}&=\frac{A-B}{a-b}+\left(2x_j(t)-a-b\right)\text{Net}(x_j(t),\vec{p})\\&+\left(x_j(t)-a\right)\left(x_j(t)-b\right)\frac{\partial\text{Net}(x_j(t),\vec{p})}{\partial x_j(t)}\end{aligned} \tag{5}$$

$$\begin{aligned}\frac{\partial^2 y_T(x_j(t),\vec{p})}{\partial x_j^2(t)}&=2\text{Net}(x_j(t),\vec{p})+2\left(2x_j(t)-a-b\right)\frac{\partial\text{Net}(x_j(t),\vec{p})}{\partial x_j(t)}\\&+\left(x_j(t)-a\right)\left(x_j(t)-b\right)\frac{\partial^2\text{Net}(x_j(t),\vec{p})}{\partial x_j^2(t)}\end{aligned} \tag{6}$$

where $\dfrac{\partial\text{Net}(x_j(t),\vec{p})}{\partial x_j(t)}=\alpha_i\dfrac{dg(z_i)}{dx_j(t)}$ and $\dfrac{\partial^2\text{Net}(x_j(t),\vec{p})}{\partial x_j^2(t)}=\alpha_i\dfrac{d^2g(z_i)}{dx_j^2(t)}$ at iteration t. The definition of the function g regarding with the selected activation function in RNN are listed in Table 1.

Finally, Dirichlet Boundary Problem for ODEs is transformed to an optimization problem as given in Eq. (7).

$$\text{Problem}: \underset{\vec{p}\in R^{4m}}{\arg\min}\{E\} \tag{7}$$

For solving the problem given in Eq. (7), derivative based optimization methods like Gradient Descent are often used. However, most of time, the mentioned optimization methods fall into the trap at local optimum without converging to global optimum. Moreover, the derivative based methods require to calculate the partial derivatives of the cost function $E = E\left(\vec{\alpha}, \vec{\omega}, \vec{\Omega}, \vec{\beta}\right)$ where $\vec{\alpha}, \vec{\omega}, \vec{\Omega}, \vec{\beta} \in R^m$ are the unknown parameters of the network such that m is the number of neurons in the neural network. Therefore, as the number of neurons in the neural network increases, the method requires more calculations. To handle the problem of falling into the local minima and to reduce the workload needed to solve the problem as given in Eq. (7), we used a variant of Particle Swarm Optimization (PSO) introduced by Kennedy and Eberhart (1995), in this study [11]. Because, when the number of neurons in RNN is increasing, the number of unknown parameters of the optimization problem are also increased, and it becomes more difficult to solve the problem. In addition, the cross-validation approach has been used to validate the method. With cross-validation process, one can check whether the proposed model converges to the solution, or not, every time with various population distribution initially.

3. Experiments

In this section, we present the numerical solutions, obtaining from RNNs, of both of linear and nonlinear types of Dirichlet Boundary Problems for ODEs. In all of the experiments, we use the step size as h = 0.02 at the training stage of the network. After training the RNN, the network is tested with inputs generated using half of the step size used in the training stage as $\frac{h}{2}$. The network includes 5 neurons with single hidden layer only, and the maximum number of epochs is selected as 3.000 for stopping criteria of training. The lower and upper bound values of each arbitrary parameter of the networks are -10 and 10 respectively.

The consideration by means of performance analysis is given via the Mean Squared Errors (MSEs). We use cross validation for parameter tuning for the arbitrary parameters of RNN. For this, each method is executed 10 times to obtain mean and standard deviations of MSEs obtained in the training and testing phase independently. We compare all of the obtained numerical solutions with exact solutions regarding with absolute errors using the RNN parameters produces the lowest cost value.

The proposed approach is coded in Python and implemented on a Windows 64 bit operating system with a 3.4Ghz Intel(R) Core (TM) i7-2600 CPU and 16 Gb 800Mhz DDR3 RAM.

Example 1. The second order homogenous linear differential equation with Dirichlet Boundary Condition in Eq. (8) has the exact solution as $y(x) = \sin(x) - \frac{1}{2}\cos(x)$.

$$\begin{cases} y''+y=0, & x \in \left[0, \dfrac{\pi}{2}\right] \\ y(0) = -\dfrac{1}{2}, \\ y\left(\dfrac{\pi}{2}\right) = 1 \end{cases} \tag{8}$$

According to the mean of MSEs shown in Table 2, when ReLU are used in RNN as activation function, the best numerical solutions are obtained. Therefore, the absolute errors are listed in Table 3 with ReLU activation function. The plot of numerical solution compared with exact solution of Eq. (8) is depicted in Fig. 2.(a). In addition, Fig. 2. (b) shows the changes of the cost value with respect to the iteration. The decreasing cost when the iteration increasing validates that the RNN trained by PSO solves the DBVPs for ODEs. It also shows the speed of convergence of RNN regarding with the activation functions, and both of Table 2 and Fig. 2. (b) also underline that the worst results are occured with hyberbolic tangent function.

Table 2. The obtained average of MSEs with standard deviations from the numerical solutions for Eq. (8).

Activation	**Mean of MSEs on**	
Function	**Training Set**	**Test Set**
tanh	$1.667\times10^{-3} \pm 1.183\times10^{-3}$	$1.696\times10^{-3} \pm 1.212\times10^{-3}$
ReLU	$1.480\times10^{-4} \pm 1.406\times10^{-4}$	$1.480\times10^{-4} \pm 1.402\times10^{-4}$
Leaky ReLU	$1.700\times10^{-4} \pm 1.212\times10^{-4}$	$1.719\times10^{-4} \pm 1.237\times10^{-4}$
PReLU	$1.994\times10^{-4} \pm 1.636\times10^{-4}$	$1.994\times10^{-4} \pm 1.653\times10^{-4}$
ELU	$1.794\times10^{-4} \pm 1.535\times10^{-4}$	$1.824\times10^{-4} \pm 1.556\times10^{-4}$
SWISH	$3.265\times10^{-4} \pm 3.200\times10^{-4}$	$3.381\times10^{-4} \pm 3.355\times10^{-4}$

Table 3. The absolute errors on some quadrature points in the both of training and test sets for Eq. (8) by using ReLU activation function.

k	x_k	$E = \lvert y(x_k) - y_T(x_k) \rvert$ for Training Set	k	x_k	$E = \lvert y(x_k) - y_T(x_k) \rvert$ for Test Set
1	0.00	0.000	1	0.00	0.000
2	0.02	1.072×10^{-3}	7	0.11	4.387×10^{-3}
3	0.04	2.041×10^{-3}	12	0.21	5.981×10^{-3}
4	0.06	2.907×10^{-3}	17	0.31	5.682×10^{-3}
5	0.08	3.675×10^{-3}	22	0.41	3.890×10^{-3}
6	0.10	4.346×10^{-3}	27	0.51	1.026×10^{-3}
11	0.20	6.356×10^{-3}	32	0.61	2.483×10^{-3}
21	0.40	4.879×10^{-3}	37	0.71	6.220×10^{-3}
31	0.60	1.176×10^{-3}	42	0.81	9.779×10^{-3}
41	0.80	8.448×10^{-3}	48	0.91	1.084×10^{-2}
51	1.00	1.381×10^{-2}	53	1.01	1.306×10^{-2}
61	1.20	1.473×10^{-2}	58	1.11	1.417×10^{-3}
71	1.40	9.576×10^{-3}	63	1.21	1.392×10^{-3}
79	1.56	7.403×10^{-4}	79	1.51	3.494×10^{-3}

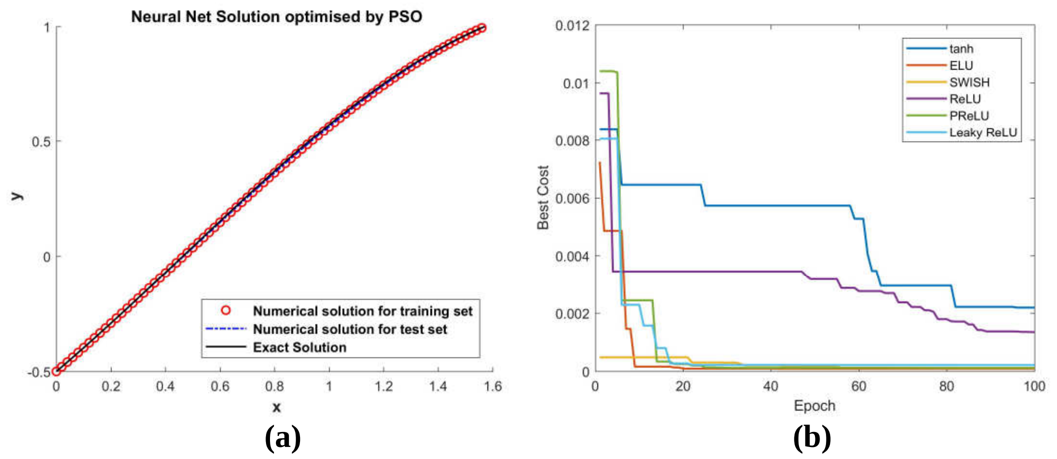

Figure 2. **(a)** The graph of the numerical and exact solution of Eq. (x1) **(b)** Best of cost values according to activation functions for 100 epochs.

Example 2. Let us consider the nonlinear differential equation with Dirichlet Boundary Condition given in Eq. (9). The the exact solution of the given problem is $y(x) = \dfrac{1}{x+1}$.

$$\begin{cases} y'' = y^3 - yy', \quad x \in [1,2] \\ y(1) = -\dfrac{1}{2}, \\ y(2) = \dfrac{1}{3} \end{cases} \tag{9}$$

Best results are obtained with SWISH function as depicted in Table 4. Thus, Table 5 gives the absolute errors when using SWISH. The exact solution and the best of RNN solution are compared in Fig. 3.(a). Moreover, Fig. 3. (b) shows the changes of the cost value relative to the iteration according to PSO. It also shows the speed of convergence of RNN depending on the activation functions, and Table 4 and Fig. 3. (b) emphasize that the worst results are occured with hyberbolic tangent as same with linear case.

Table 4. The obtained average of MSEs with standard deviations from the numerical solutions for Eq. (9).

Activation Function	**Mean of MSEs on**	
	Training Set	**Test Set**
tanh	$5.426\times10^{-5} \pm 1.537\times10^{-4}$	$5.495\times10^{-5} \pm 1.010\times 10^{-4}$
ReLU	$1.155\times10^{-5} \pm 2.609\times10^{-5}$	$1.142\times10^{-5} \pm 2.600\times 10^{-5}$
Leaky ReLU	$2.075\times10^{-5} \pm 4.491\times10^{-5}$	$2.041\times10^{-5} \pm 4.438\times 10^{-5}$
PReLU	$1.224\times10^{-5} \pm 1.897\times10^{-5}$	$1.213\times10^{-5} \pm 1.868\times 10^{-5}$
ELU	$1.084\times10^{-5} \pm 1.647\times10^{-5}$	$1.141\times10^{-5} \pm 1.928\times 10^{-5}$
SWISH	$1.003\times10^{-5} \pm 1.505\times10^{-5}$	$1.010\times10^{-5} \pm 1.518\times 10^{-5}$

Table 5. The absolute errors on some quadrature points in the both of training and test sets for Eq. (9) by using SWISH activation function.

k	x_k	$E = \lvert y(x_k) - y_T(x_k) \rvert$ for Training Set	k	x_k	$E = \lvert y(x_k) - y_T(x_k) \rvert$ for Test Set
1	1.00	0.000	1	1.00	0.000
2	1.02	1.737×10^{-4}	7	1.11	6.155×10^{-4}
3	1.04	3.172×10^{-4}	12	1.21	6.209×10^{-4}
4	1.06	4.327×10^{-4}	17	1.31	3.224×10^{-4}
5	1.08	5.226×10^{-4}	22	1.41	1.106×10^{-4}
6	1.10	5.889×10^{-4}	27	1.51	5.456×10^{-4}
11	1.20	6.344×10^{-4}	32	1.61	8.798×10^{-4}
21	1.40	6.945×10^{-5}	37	1.71	1.034×10^{-3}
31	1.60	8.574×10^{-4}	42	1.81	9.518×10^{-4}
41	1.80	9.749×10^{-4}	47	1.91	5.891×10^{-4}
51	2.00	0.000	53	2.00	0.000

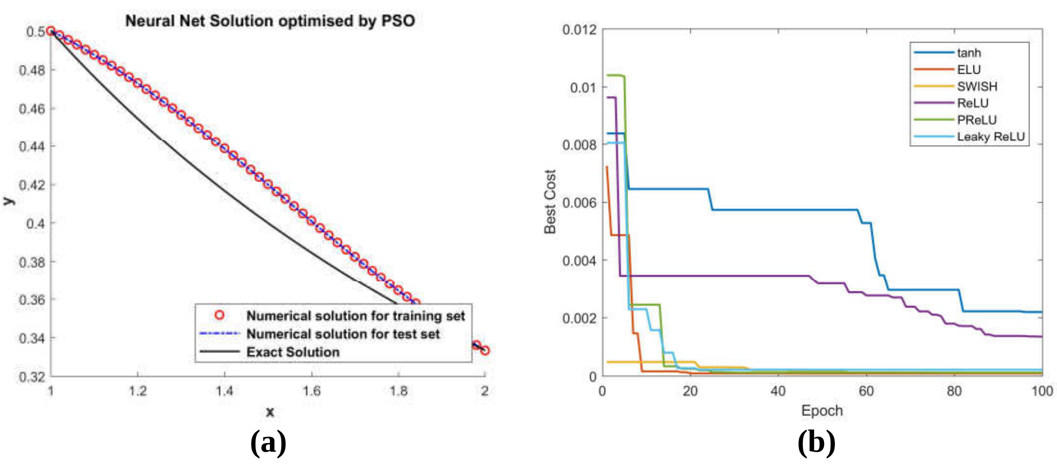

Figure 3. **(a)** The graph of the numerical and exact solution of Eq. (x2) **(b)** Best of cost values according to activation functions for 100 epochs.

4. Conclusions

In this work, a simple recurrent neural network trained by Partical Swarm Optimization is constructed for solving ODEs with Dirichlet Boundary Conditions. According to this study, it is showed that the recurrent neural networks can be a good alternative when trying to obtain fast solution of various challenging, mostly nonlinear, problems modelled by differential equations. Because, once the network is trained with some inputs, it gives nearly optimal solution at any point of problem domain unlike traditional methods. It is not required to retrain the neural network after the first construction.

Futhermore, six different activation functions are used in the network model. The convergence to the solution is obtained with all of them. In the linear case, the best results are obtained from ReLU activation functions, and the SWISH is more successful among others in the nonlinear case. However, the fastest convergence is observed with the ReLU activation function, according to Mean Squared Errors. In both cases, the slowest convergence was observed with the hyberbolic tangent function.

As a future work, the more complex neural network models, providing feedback among neurons, used in Deep Learning such as Reservoir or Echo State Networks can be used to optimize results. Moreover, some heuristics, meta-heuristics as global optimization methods, and the hybridization of them can be used to train the neural networks to solve Dirichlet Boundary Problems.

It must be emphasized that, most of problems are more complex in real worlds application, so they are modelled with delay differential equations, partial differential equations or integro-differential equations. Therefore, the model should be extended to be experienced on them.

Acknowledgments

This study is an extended version of the conference paper presented in "International Conference on Applied Mathematics in Engineering (ICAME 18)" Günel, İşman and Kocakula, (2018). We would like to thank the anonymous reviewers for their helpful comments to improve the paper.

References

[1] Lee, H., Neural algorithms for solving differential equations. **Journal of Computational Physics**, 91, 1, 110-131, (1990).

[2] Meade, A. J. ve Fernandez, A. A., The numerical solution of linear ordinary differential equations by feedforward neural networks. **Mathematical and Computer Modelling**, 19, 12, 1 – 25, (1994).

[3] Lagaris, I. E., Likas, A. and Fotiadis, D. I., Artificial neural networks for solving ordinary and partial differential equations. **IEEE Transactions on Neural Networks**, 9, 5, 987-1000, (1998).

[4] Malek, A. and Beidokhti, S. R. Numerical solution for high order differential equations using a hybrid neural network - Optimization method. **Applied Mathematics and Computation**, 183, 1, 260-271, (2006).

[5] Raja, M. A., Numerical treatment for boundary value problems of Pantograph functional differential equation using computational intelligence algorithms. **Applied Soft Computing**, 24, 806-821, (2014).

[6] Raja, M. A., Ahmad, S., and Samar, R., Solution of the 2-dimensional Bratu problem using neural network, swarm intelligence and sequential quadratic programming. **Neural Computing and Applications**, 25, 7-8, 1723-1739, (2014).

[7] Raja, M.A.Z., Stochastic numerical treatment for solving Troesch's problem. **Information Sciences**, 279, 860 – 873, (2014).

[8] Raja, M.A.Z., Manzar, M.A., Samar, R., An efficient computational intelligence approach for solving fractional order riccati equations using ann and sqp. **Applied Mathematical Modelling**, 39, 10, 3075 – 3093, (2015).

[9] Brezak, D., Bacek, T., Majetic, D., Kasac, J., and Novakovic, B., A comparison of feed-forward and recurrent neural networks in time series forecasting. **2012 IEEE Conference on Computational Intelligence for Financial Engineering** & Economics (CIFEr), New York, NY, 1 – 6, (2012).

[10] Saini, S.S., Parkhe, O. and Khadtare, T.D., Analysis of Feedforward and Recurrent Neural Network in Forecasting Foreign Exchange Rate. **Imperial Journal of Interdisciplinary Research (IJIR)**, 2, 822 – 826, (2016).

[11] Kennedy, J. and Eberhart, R.C. Particle swarm optimization, **Proceedings of the IEEE International Conference on Neural Networks**, 1942-1948, (1995).