


Approximate Solutions of Initial Value Problems for Ordinary Differential Equations Using Radial Basis Function Networks

Fatma B. Rizaner¹ · Ahmet Rizaner² 

© Springer Science+Business Media, LLC, part of Springer Nature 2017

Abstract We present a numerical approach for the approximate solutions of first order initial value problems (IVP) by using unsupervised radial basis function networks. The proposed unsupervised method is able to solve IVPs with high accuracy. In order to demonstrate the efficiency of the proposed approach, we also compare its solutions with the solutions obtained by a previously proposed neural network method for representative examples.

Keywords Initial value problems · Ordinary differential equations · Radial basis function network · Artificial neural networks · Function approximation

1 Introduction

Many problems in science and engineering can be represented in terms of ordinary differential equations (ODEs) [3, 7, 15]. As it may not always be easy to obtain the exact solutions of such ODEs, numerical approaches have been developed. In addition to numerical approaches, techniques involving the architectures of artificial neural networks are also used in finding the solution of initial value problems (IVPs) for ODEs [4, 12, 13, 17, 19, 20]. Mall and Chakraverty have proposed several neural network based approaches for the solutions of partial differential equations (PDEs) and ODEs [12–14]. They have considered a regression based artificial neural network model in [12] for the solutions of ODEs. It has been shown with examples that the approximate results obtained by the neural model are very accurate [12]. Mall and Chakraverty [13] have developed a new neural network based solution of IVPs by the use of Legendre polynomials. A simple and computationally efficient approach based on Chebyshev neural network has been proposed in order to solve PDEs [14]. It has been shown that this

✉ Fatma B. Rizaner
fatma.bayramoglu@emu.edu.tr

¹ Department of Mathematics, Eastern Mediterranean University, Mersin 10, Famagusta, North Cyprus, Turkey

² Department of Information Technology, Eastern Mediterranean University, Mersin 10, Famagusta, North Cyprus, Turkey

simple and computational efficient network model can solve elliptic PDEs with a single layer. Another scheme based on the unsupervised version of kernel least mean square algorithm has been implemented by Yazid et al. [20] for the solution of ODEs. Aarts and Van Der Veer have described a method to solve PDEs with initial conditions by using neural networks [1]. They have stated that a feed forward network with a single hidden layer with no bias is capable of approximating derivatives of the functions successfully. This observation has also been made in [2] and [11]. Shirvany et al. [19] have proposed a method for solving ODEs and PDEs based on neural networks. These authors have pointed out that the method presented in [19] can solve the Schrodinger equation effectively with a small number of unknown parameters in comparison with well-known numerical methods such as Runge–Kutta. Another neural network based solution has been employed by Jafarian et al. [8] to approximate series solutions of a class of initial value ODEs of fractional orders. Nejad et al. have proposed a fuzzy wavelet neural network based approximation approach of the first order partial derivatives in [16].

Radial basis function (RBF) networks are special types of artificial neural networks. Although these kind of networks are well suited for problems regarding classification, it has been also reported that RBF networks are capable of universal approximation even with a single hidden layer [17]. Qu [18] has proposed a cosine RBF neural network to solve fractional differential equations with IVPs. The generalization ability of multi-layer back propagation and reformulated RBF network has been compared by Choi and Lee [6]. They have concluded that reformulated RBF networks with supervised learning have better performance in convergence and generalization when solving differential equations [6].

In this paper, we propose a new approach based on RBF networks for solving ODEs. The approximate solution, whose parameters are adjusted to minimize an appropriate error function, is presented in the closed form by means of RBFs. Some numerical examples are also presented and results are compared with some existing methods to illustrate the efficiency of the proposed approach.

The rest of the paper is organized as follows. Section 2 presents the proposed RBF network model. In Sect. 3, we discuss the application of RBF network to solve first order IVPs. Section 4 presents numerical results and their discussions. Finally, in Sect. 5 we conclude with a final overview of the obtained results.

2 Radial Basis Function (RBF) Network Model

In this research, we consider a three layered network with one input, one hidden layer of radial basis functions and a single output unit. The architecture of this radial basis function network model with three layers is shown in Fig. 1. The first layer of network performs a non-linear transformation of the input with the non-linear Gaussian basis activation functions [5,9]. The i -th RBF can be represented as:

$$\varphi_i(x) = e^{-\frac{(x-c_i)^2}{\sigma^2}}, \quad (1)$$

where x is the input of the network, c_i and σ represents the i -th center and width of the Gaussians basis functions respectively. The output of the RBF network is the linear combination of the outputs from n radial basis functions weighted by appropriate coefficients (w_j),

$$z(x) = \sum_{j=1}^n w_j \varphi_j(x). \quad (2)$$

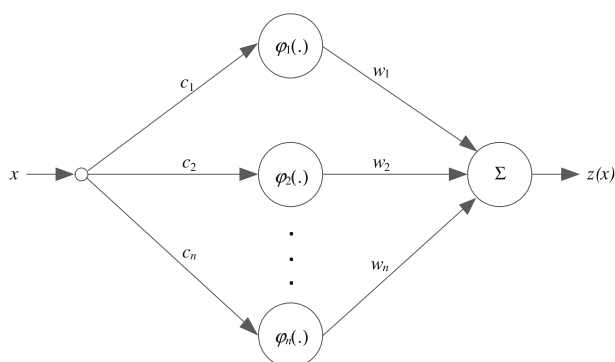


Fig. 1 Architecture of a radial basis function network

Table 1 Exact solution and approximations of proposed unsupervised RBF based network for example 1

t	$y(t)$	RBF Net. ($n = 3$)	RBF Net. ($n = 5$)	RBF Net. ($n = 7$)	RBF Net. ($n = 9$)
0.00	1.0000	1.0000	1.0000	1.0000	1.0000
0.05	0.9536	0.9500	0.9543	0.9536	0.9536
0.10	0.9137	0.9111	0.9144	0.9137	0.9137
0.15	0.8798	0.8800	0.8799	0.8798	0.8798
0.20	0.8514	0.8547	0.8507	0.8514	0.8514
0.25	0.8283	0.8344	0.8269	0.8283	0.8283
0.30	0.8104	0.8184	0.8086	0.8104	0.8104
0.35	0.7978	0.8067	0.7958	0.7978	0.7978
0.40	0.7905	0.7995	0.7887	0.7905	0.7905
0.45	0.7889	0.7971	0.7874	0.7889	0.7889
0.50	0.7931	0.7999	0.7921	0.7931	0.7930
0.55	0.8033	0.8081	0.8028	0.8034	0.8033
0.60	0.8200	0.8222	0.8197	0.8200	0.8199
0.65	0.8431	0.8426	0.8429	0.8432	0.8431
0.70	0.8731	0.8697	0.8726	0.8732	0.8731
0.75	0.9101	0.9040	0.9088	0.9101	0.9100
0.80	0.9541	0.9460	0.9519	0.9541	0.9540
0.85	1.0053	0.9962	1.0021	1.0053	1.0052
0.90	1.0637	1.0552	1.0598	1.0637	1.0637
0.95	1.1293	1.1237	1.1254	1.1294	1.1293
1.00	1.2022	1.2022	1.1991	1.2022	1.2021
Average MSE:		3.48×10^{-05}	3.54×10^{-06}	7.75×10^{-10}	6.80×10^{-10}

One of the main difference between the RBF networks and multilayer perceptron (MLP) networks is how the outputs of the activation functions of the first layer are calculated. The activations of the RBF network depend to the distances between the inputs and the centers of the RBFs not to the inner products of the inputs and the weights.

Table 2 Comparison of exact solution and approximations for example 1

t	$y(t)$	Euler	Runge–Kutta	MLP-A	MLP-R	RBF Net. ($n = 9$)
0.00	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.05	0.9536	0.9500	0.9536	0.9886	0.9677	0.9536
0.10	0.9137	0.9072	0.9138	0.9084	0.9159	0.9137
0.15	0.8798	0.8707	0.8799	0.8906	0.8815	0.8798
0.20	0.8514	0.8401	0.8515	0.8587	0.8531	0.8514
0.25	0.8283	0.8150	0.8283	0.8309	0.8264	0.8283
0.30	0.8104	0.7953	0.8105	0.8013	0.8114	0.8104
0.35	0.7978	0.7810	0.7979	0.7999	0.7953	0.7978
0.40	0.7905	0.7721	0.7907	0.7918	0.7894	0.7905
0.45	0.7889	0.7689	0.7890	0.7828	0.7845	0.7889
0.50	0.7931	0.7717	0.7932	0.8047	0.7957	0.7930
0.55	0.8033	0.7805	0.8035	0.8076	0.8041	0.8033
0.60	0.8200	0.7958	0.8201	0.8152	0.8204	0.8199
0.65	0.8431	0.8178	0.8433	0.8319	0.8399	0.8431
0.70	0.8731	0.8467	0.8733	0.8592	0.8711	0.8731
0.75	0.9101	0.8826	0.9102	0.9129	0.9151	0.9100
0.80	0.9541	0.9258	0.9542	0.9755	0.9555	0.9540
0.85	1.0053	0.9763	1.0054	1.0056	0.9948	1.0052
0.90	1.0637	1.0342	1.0638	1.0714	1.0662	1.0637
0.95	1.1293	1.0995	1.1294	1.1281	1.1306	1.1293
1.00	1.2022	1.1721	1.2022	1.2108	1.2058	1.2021
Average MSE:		4.60×10^{-04}	1.24×10^{-08}	1.26×10^{-04}	2.01×10^{-05}	6.80×10^{-10}

3 Unsupervised RBF Network Solution of First Order IVPs

A first-order initial value problem could be formulated as:

$$\frac{\partial y(x)}{\partial x} = f(x, y), x \in [a, b] \quad (3)$$

with the initial condition $y(a) = A$.

We can define a trial solution for (3) as sum of two terms in the following form [10]:

$$y_T(x) = A + (x - a)z(x). \quad (4)$$

The first part of (4) satisfies the initial condition and contains no adjustable parameters. The second part involves an RBF network whose weights must be changed in such a way that the network error is reduced. To achieve this, it is common to define the error function of the approximation as the square of the error between the derivative of the trial solution and exact solution as [7]:

$$E(w) = \frac{1}{2} \left[\frac{\partial y_T(x)}{\partial x} - f(x, y) \right]^2. \quad (5)$$

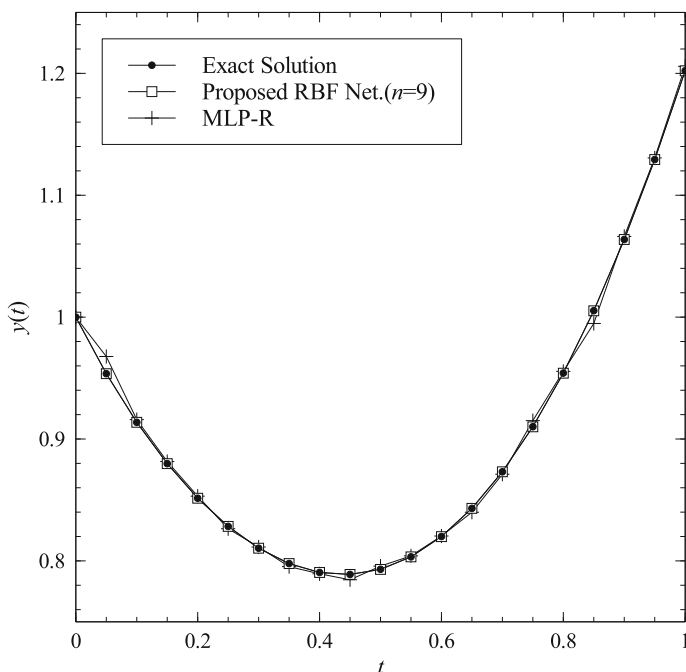


Fig. 2 Exact solution and approximations of example 1

In order to minimize $E(w)$ and update the weight coefficients, we differentiate $E(w)$ with respect to the weight coefficients. The weight coefficients of the RBF network is updated starting with initial random weight to minimize (5) with the gradient descent method as:

$$w_j^k = w_j^{k-1} - \eta \frac{\partial E(w)}{\partial w_j^k}, \quad (6)$$

where η is the learning rate parameter, k is the iteration index and, $\partial E(w)/\partial w_j^k$ can be calculated as:

$$\begin{aligned} \frac{\partial E(w)}{\partial w_j^k} = & \left(1 - \frac{2}{\sigma^2} (x - c_j)^2 (x - a) e^{-\frac{(x - c_j)^2}{\sigma^2}} \right) \\ & \times \left(\sum_{i=1}^n \left(1 - \frac{2}{\sigma^2} (x - c_i)^2 (x - a) w_i^k e^{-\frac{(x - c_i)^2}{\sigma^2}} \right) - f(x, y) \right). \end{aligned} \quad (7)$$

The proposed algorithm is unsupervised because target solution is not known and generated by iterating the algorithm to minimize $E(w)$. Traditional iterative numerical methods offer discrete solutions for IVPs. Since they use fixed step size for the approximation process, if any solution is needed between the steps, the process needs to be repeated with a new step size. However, the solution of the presented RBF network model is differentiable. The solution at any point can be calculated without repeating the training process.

Table 3 Exact solution and approximations of proposed unsupervised RBF based network for example 2

t	$y(t)$	RBF Net. ($n = 9$)	RBF Net. ($n = 15$)	RBF Net. ($n = 21$)
0.00	3.0000	3.0000	3.0000	3.0000
0.15	2.3438	2.3056	2.3447	2.3435
0.30	1.8142	1.4605	1.8155	1.8138
0.45	1.3511	0.8294	1.3490	1.3510
0.60	0.9348	0.5047	0.9312	0.9344
0.75	0.5763	0.4158	0.5743	0.5752
0.90	0.3012	0.4280	0.3015	0.2998
1.05	0.1318	0.4214	0.1324	0.1308
1.20	0.0726	0.3380	0.0710	0.0720
1.35	0.1038	0.1914	0.0985	0.1030
1.50	0.1845	0.0420	0.1757	0.1834
1.65	0.2643	-0.0441	0.2539	0.2632
1.80	0.2988	-0.0343	0.2900	0.2982
1.95	0.2638	0.0510	0.2589	0.2637
2.10	0.1625	0.1469	0.1614	0.1622
2.25	0.0235	0.1737	0.0237	0.0227
2.40	-0.1095	0.0771	-0.1109	-0.1107
2.55	-0.1937	-0.1374	-0.1966	-0.1952
2.70	-0.2025	-0.3881	-0.2032	-0.2043
2.85	-0.1348	-0.5207	-0.1300	-0.1373
3.00	-0.0157	-0.3263	-0.0099	-0.0186
Average MSE:		6.67×10^{-2}	1.96×10^{-5}	1.44×10^{-6}

4 Numerical Examples

In this section, we consider two examples to illustrate the performance of the proposed RBF based unsupervised approximation approach. The estimation accuracies of the presented approaches are calculated by means of average mean squared error (MSE) as,

$$E_{mse} = \frac{1}{n_s} \sum_{j=1}^{n_s} (y(x_j) - y_T(x_j))^2, \quad (8)$$

where n_s is the total number of test points and x_j represents the j -th test input.

4.1 Example 1

Consider the following first order ODE with time varying coefficient as follow:

$$\frac{\partial y(t)}{\partial t} + \left(t + \frac{1 + 3t^2}{1 + t + t^3} \right) y(t) = 2t + t^3 + t^2 \left(\frac{1 + 3t^2}{1 + t + t^3} \right), \quad t \in [0, 1] \quad (9)$$

with initial condition $y(0) = 1$.

The approximate solutions obtained by the proposed approach are compared with the exact solution in Table 1. The approximate solutions are obtained with the different number

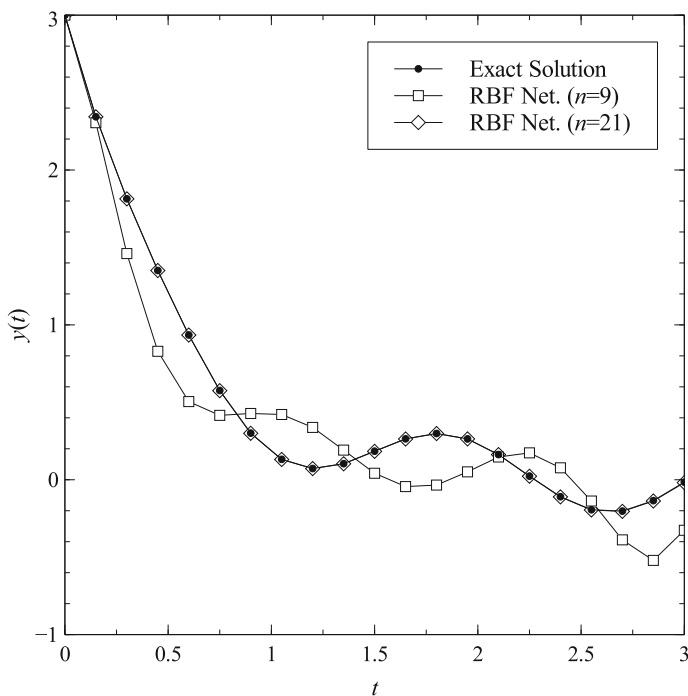


Fig. 3 Exact solution and approximations of the proposed approach for example 2

of RBFs (3, 5, 7, 9) having constant widths of 0.8 and the centers of the first layer are uniformly distributed inside $[0, 1]$. A variable learning rate factor is employed for the second layer which decreases linearly with increasing iteration number from 0.2 to 0.005. It is clearly seen from Table 1 that increasing number of RBFs improves the estimation performance. The average MSE is obtained as 6.8×10^{-10} with 9 RBFs. Increasing the number of RBFs beyond 9 does not improve the performance significantly.

The results obtained with 9 RBFs are also compared with the results obtained by multi-layer perceptron (MLP) based neural methods of [12] with arbitrary weights (MLP-A) and regression-based weights (MLP-R). Table 2 compares the exact solution with the approximation of the proposed method, MLP based approach, and also with the well-known Euler and Runge–Kutta numerical methods. As clearly seen from the results presented in Table 2, proposed RBF based network outperforms all the other approaches.

The approximate solution obtained with the proposed method is also compared with the exact solution and the approximation of MLP-R in Fig. 2. There is a perfect agreement with the exact solution and the approximate solution obtained by the proposed approach.

4.2 Example 2

Consider the following first order ODE with time varying coefficient as follow:

$$\frac{\partial y(t)}{\partial t} + 2y(t) = \cos(4t), t \in [0, 3] \quad (10)$$

with initial condition $y(0) = 3$.

Table 3 compares the approximate solutions obtained by the proposed approach with 9, 15 and 21 RBFs with the exact solution. The centers of the first layer are uniformly distributed inside $[0, 3]$, other network parameters are used as explained in example 1. The approximation obtained by 9 RBFs is not acceptable. Increasing the number of RBFs to 15 improves the average MSE of the approximations from 6.67×10^{-2} to 1.96×10^{-5} . A better approximation is obtained by increasing the number of RBFs to 21 with an average MSE of 1.44×10^{-5} . Increasing the number of RBFs beyond 21 improves the performance slightly. Figure 3 is a different way of comparing the approximations. As seen from this figure, the approximate result obtained with 21 RBFs is very close to the exact solution, however, the approximation obtained with 9 RBFs is not promising.

5 Conclusion

In this paper, an unsupervised RBF network model has been developed for solving ordinary differential equations with initial values. The accuracy of the RBF network based approach has been examined by representative examples and compared with some previously proposed neural network based approaches. It has been shown that the proposed approximation method can estimate the solutions of the first order IVPs successfully with very high accuracy.

References

1. Aarts LP, Van Der Veer P (2001) Neural network method for solving partial differential equations. *Neural Process Lett* 14(3):261–271
2. Attali JG, Pagès G (1997) Approximations of functions by a multilayer perceptron: a new approach. *Neural Netw* 10(6):1069–1081
3. Baymani M, Kerayechian A, Effati S (2010) Artificial neural networks approach for solving stokes problem. *Appl Math* 1(04):288
4. Beidokhti RS, Malek A (2009) Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques. *J Frankl Inst* 346(9):898–913
5. Chen Z, Cao F (2013) The construction and approximation of neural networks operators with gaussian activation function. *Math Commun* 18(1):185–207
6. Choi B, Lee JH (2009) Comparison of generalization ability on solving differential equations using backpropagation and reformulated radial basis function networks. *Neurocomputing* 73(1):115–118
7. Cichocki A, Unbehauen R (1992) Neural networks for solving systems of linear equations and related problems. *IEEE Trans Circuits Syst I Fundam Theory Appl* 39(2):124–138
8. Jafarian A, Mokhtarpour M, Baleanu D (2017) Artificial neural network approach for a class of fractional ordinary differential equation. *Neural Comput Appl* 28(4):765–773
9. Khosravi H (2012) A novel structure for radial basis function networksrbf. *Neural Process Lett* 35(2):177–186
10. Lagaris IE, Likas A, Fotiadis DI (1998) Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans Neural Netw* 9(5):987–1000
11. Li X (1996) Simultaneous approximations of multivariate functions and their derivatives by neural networks with one hidden layer. *Neurocomputing* 12(4):327–343
12. Mall S, Chakraverty S (2013) Comparison of artificial neural network architecture in solving ordinary differential equations. *Adv Artif Neural Syst* 2013:12
13. Mall S, Chakraverty S (2016) Application of legendre neural network for solving ordinary differential equations. *Appl Soft Comput* 43:347–356
14. Mall S, Chakraverty S (2017) Single layer chebyshev neural network model for solving elliptic partial differential equations. *Neural Process Lett* 45(3):825–840
15. Nazemi A, Karami R (2017) A neural network approach for solving optimal control problems with inequality constraints and some applications. *Neural Process Lett* 45(3):995–1023

16. Nejad HC, Farshad M, Khayat O, Rahatabad FN (2016) Performance verification of a fuzzy wavelet neural network in the first order partial derivative approximation of nonlinear functions. *Neural Process Lett* 43(1):219–230
17. Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. *Neural Comput* 3(2):246–257
18. Qu H (2017) Cosine radial basis function neural networks for solving fractional differential equations. *Adv Appl Math Mech* 9(3):667–679
19. Shirvany Y, Hayati M, Moradian R (2009) Multilayer perceptron neural networks with novel unsupervised training method for numerical solution of the partial differential equations. *Appl Soft Comput* 9(1):20–29
20. Yazdi HS, Pakdaman M, Modaghegh H (2011) Unsupervised kernel least mean square algorithm for solving ordinary differential equations. *Neurocomputing* 74(12):2062–2071