RBF Technique for Solving the Nonlinear DDEs

H. Aminikhah and R. Mehralizadeh

Department of Applied Mathematics, School of Mathematical Sciences, University of Guilan, P.O. Box 1914, P.C. 41938, Rasht, Iran

Email: aminikhah@guilan.ac.ir

Abstract: The delay differential equations (DDEs), are encountered in a wide range of application in science and engineering. In this work, we apply the radial basis function method (RBF) which is a numerical method, for solving the nonlinear delay differential equations. Illustrative examples have been discussed to demonstrate the validity and applicability of the technique and the results have been compared with the exact solution. Comparison of the approximate solution with exact solution shows that the used method is effectiveness and practical for classes of linear and nonlinear DDEs.

***Keywords**:* nonlinear delay differential equations; radial basis function.

1. Introduction

Delay differential equations (DDEs), which have some arguments with time-lags, are widely in use in many areas of physics, engineering, economics and biology. They have been studied for at least 200 years .they arise in many areas of various mathematical modeling. They have important role in applications . For example in biological application delay equations give better descriptions in population than the ordinary ones. For example, we can mention the works of Becker et al.[1]. There are many books emphasize particular DDE problems appearing in mechanical engineering, chemistry and biology. For more studies, can be referred to Driver [2], Kuang [3], Macdonald [4], Stépán [5], Fowler [6,7]. Mathematically, they have been described by researchers like Gopalsamy [8], Halanay [9],

Kolmanovskii and Myshkis [10], Kolmanovskii and Nosov [11], Hale and Verduyn Lunel [12]. Many different methods have been presented for numerical solution of DDEs. Among these are the Radau IIA method [13], Bellman's method of steps [14], wave form relaxation method [15], Runge-Kutta method and continuous Runge-Kutta method [16, 17]. During the last decade, they have paid great attention on the use of RBFs for either interpolation or for solving mathematical problem in different field of research and drew many researcher to solve the problems in higher-dimensional spaces. Due to the RBFs as a class of meshless schemes avoid grid generation and the domain of interest can be considered by a set of scattered data points among which there is no pre-defined connectivity. This method of solution is effective on scattered data points and in irregular geometries, is easy to implement in any finite dimension and is spectrally accurate. In this paper, we use the multiquadrics radial basis function for finding the solution of DDEs. The MQ was first developed by hardy [18] in 1971 as a multidimensional scattered interpolation method in modeling of the earth gravitational field. It was not recognized by most of the academic researchers until Franke [19] published a review paper on the evaluation of two- dimensional interpolation methods, whereas MQ was ranked as the best based on its accuracy, visual aspect sensitivity to parameters, execution time, storage requirements, and ease of implementation. The next remarkable time in RBF history was in 1986 when Charles Micchelli, proved that the system matrix for the MQ method was invertible [20]. After that, Edward Kansa first used the MQ method for solving differential equations [21]. Recently, study in RBF method has rapidly grown, and RBFs are now an effective way to solve various differential equation. In the present work, a numerical method based on the radial basis function method and collocation point, and then applied to the nonlinear delay differential equations as follows [22]

$$u^{(m)}(t) = \sum_{j=0}^{J} \sum_{n=0}^{m-1} \mu_{jn}(t) u^{(n)}(\alpha_{jn} t + \beta_{jn}) + g(u) + f(t) \tag{1.1}$$

with initial condition

$$\sum_{n=0}^{m-1} c_{in} u^{(n)}(0) = \lambda_i, \ i = 0,1,2,\ldots,m-1 \tag{1.2}$$

where $g(u) = bu + \sum_{i=1}^{l} d_i u^{\gamma_i}, d_i \in \mathbb{R}, \gamma_i \in \mathbb{N}, \gamma_i > 1, l \in \mathbb{N}, 0 \le \alpha_{jn} \le 1, \beta_{jn} \in \mathbb{R}$. $u^{(m)}$ in $u^{(m)}$ is considered as the $m-th$ derivative of the function u.

The rest of the article is organized as follows: Section 2, introduces the basic idea of radial basis function method. Section 3, is devoted to solving equation (1.1) using the Radial basis. In Section 4, we presented two examples, error analysis and comparisons are made between the results of the approximate solutions and exact solution is presented for this method. Finally, we give a brief conclusion in section 5.

2. Radial Basis Function

2.1. Definition of radial basis function

Let $\mathbb{R}^+ = \{x \in \mathbb{R}, x \ge 0\}$ be the non-negative half-line and let $\varphi : \mathbb{R}^+ \to \mathbb{R}$ be a continuous function with $\phi(0) \ge 0$. A radial basis function on $\mathbb{R}^d$ is a function of the form $\varphi(\|X - X_i\|)$, where $X, X_i \in \mathbb{R}^d$ and $\|.\|$ denotes the Euclidean norm. If one chooses N points $\{X_i\}_{i=1}^{N}$ in $\mathbb{R}^d$, then, by custom

$$S(x) = \sum_{j=1}^{N} \lambda_j \varphi(\|X - X_j\|);\ \lambda_j \in \mathbb{R}$$

is called a radial basis function as well [23]. Common choices of RBFs $\varphi\|X - X_j\|$, fall into three main categories [24].

Global, infinitely differentiable:

These radial basis functions are dependent on the choice of a parameter called shape parameter and infinitely differentiable like, Multiquadric (MQ), Gaussian (GA), Inverse Multiquadric (IMQ), and inverse quadric(IQ), (See Table 1).

Global and finitely smooth:

We cannot differentiate the basis functions of this category infinitely. These basis functions are shape parameter free and have comparatively less accuracy than the basis functions discussed in the Category 1. For example, thin plate spline, etc. [25]. The TPS is piece wise smooth. these function are not smooth centers.

Compactly supported and finitely smooth:

For example, the truncated power function, which is also called Askey's power function [26], given by, $\varphi_\ell(r) = \left(1-r\right)_+^\ell$, *where* $r = \left\|X\right\|_2, a_+ = \max\left\{a, 0\right\}$, this function do not have continuous derivatives at $\left\|X\right\|_2 = 0$ and $\left\|X\right\|_2 = 1$, even when ℓ is large,i.e. $\varphi_\ell \in C^0$.

Table1: Some of the most important globally supported RBFs, $(r = \|X - X_i\|, c > 0)$.

RBF	*Defination*
Multiquadric (MQ)	$\sqrt{r^2 + c^2}$
Inverse Multiquadric (IMQ)	$1/\sqrt{r^2 + c^2}$
Thin-plate Spline (TPS)	$r^2 \log r$
Guassian (GA)	$\exp\left(-r^2c^2\right)$
Inverse Quadric	$1/r^2 + c^2$

In RBF function, c is the shape parameter and we use the random variable shape strategy parameter

$$c_j = c_{\min} + (c_{\max} - c_{\min}) \times rand(1, N).$$

The function rand is the Matlab function that returns N uniformly distributed pseudo-random numbers on the unit interval , and $c_{\min}$, $c_{\max}$ are two input parameters chosen so that the ratio $\frac{c^2_{\max}}{c^2_{\min}} \cong 10$ to 10^9 [27-29].

2.2. Using RBFs to approximate a function

Let us choose N distinct nodes $\left(X_j, j = 1,2,3,\ldots,N\right)$ in the interval $[a,b]$. In RBF methods, a function $u(X)$ is approximated by $u_n(X)$ as

$$u(X) \approx u_n(X) = \sum_{j=1}^{N} \lambda_j \varphi(r_j) = \Phi^T(r)\lambda, \tag{2.1}$$

where $r_j = \left\|X - X_j\right\|$ and λ_j the set of coefficients to be determined, $\phi(r), r \geq 0$ is some RBF, $\Phi(r) = [\varphi_1(r), \varphi_2(r), \varphi_3(r), \ldots, \varphi_N(r)]^T$, and $\lambda = [\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_N]^T$.

By considering $u_n(X_i) = u_i$, Eq (1.1) can be represented as the following

$$\mathrm{A}\lambda = u, \tag{2.2}$$

where $u = [\,u_1, u_2, u_3, ..., u_N\,]^T$, and

$$A = \begin{bmatrix} \Phi^T(r_1) \\ \Phi^T(r_2) \\ \vdots \\ \Phi^T(r_N) \end{bmatrix} = \begin{bmatrix} \varphi_1(r_1) & \varphi_2(r_1) & \cdots & \varphi_N(r_1) \\ \varphi_1(r_2) & \varphi_2(r_2) & \cdots & \varphi_N(r_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(r_N) & \varphi_2(r_N) & \cdots & \varphi_N(r_N) \end{bmatrix}.$$

From Eqs. (2.1) and (2.2), $u_n(x)$ can be written as $u_n(x) = \Phi^T(r)\mathrm{A}^{-1}u$.

3. Using RBFs for delay differential equations

For the solution of equation (1.1) it is sufficient to suppose that approximate solution is

$$u(t) \approx u_n(t) = \sum_{j=1}^{N} \lambda_j \varphi(\|t - t_j\|), \tag{3.1}$$

$$u^{(m)}(t) \approx u_n^{(m)}(t) = \sum_{j=1}^{N} \lambda_j \varphi^{(m)}(\|t - t_j\|), \tag{3.2}$$

$$u^{(n)}(\alpha_{jn}t + \beta_{jn}) \approx u_n^{(n)}(\alpha_{jn}t + \beta_{jn}) = \sum_{j=1}^{N} \lambda_j \varphi^{(n)}(\alpha_{jn}t + \beta_{jn} - t_j), \tag{3.3}$$

Then, from Substituting (3.1), (3.2), (3.3) in Eq. (1.1), we have

$$\begin{aligned} \sum_{j=1}^{N} \lambda_j \varphi^{(m)}(\|t - t_j\|) = \sum_{j=0}^{J} \sum_{n=0}^{m-1} \mu_{jn}(t) \sum_{j=1}^{N} \lambda_j \varphi^{(n)}(\alpha_{jn}t + \beta_{jn} - t_j) \\ + g(\sum_{j=1}^{N} \lambda_j \varphi(\|t - t_j\|)) + f(t) \end{aligned} \tag{3.4}$$

We now collocate Eq. (3.4) At points $t_i, i = 1,2,3,...,N,$ as

$$\begin{aligned} \sum_{j=1}^{N} \lambda_j \varphi^{(m)}(\|t_i - t_j\|) - \sum_{j=0}^{J} \sum_{n=0}^{m-1} \mu_{jn}(t_i) \sum_{j=1}^{N} \lambda_j \varphi^{(n)}(\alpha_{jn}t_i + \beta_{jn} - t_j) \\ - g(\sum_{j=1}^{N} \lambda_j \varphi(\|t - t_j\|)) = f(t_i) \end{aligned} \tag{3.4}$$

with Eq. (3.5) and imposing the supplementary condition,

$\sum_{n=0}^{m-1} c_{in} u^{(n)}(0) = \sum_{n=0}^{m-1} c_{in} \sum_{j=1}^{N} \lambda_j \varphi^{(n)}(\|0 - t_j\|) = \lambda_i,\ i = 0,1,2,\ldots,m-1$ on the problem, we have a nonlinear system of equations that can be solved by the Newton's method to obtain the unknown coefficients $\{\lambda_j\}_{j=1}^{N}$. Similarly, we can do the way in above for other classes of DDEs.

4. Numerical Examples

In order to illustrate the advantages and the accuracy of the RBF technique for solving the nonlinear delay differential equations, we have applied the method to Solving two nonlinear delay differential equations from class of Eq.(1.1).

Example 1. We consider Eq.(1.1) with $m = 1, J = 2$ and the following nonzero coefficients μ_{jn}, α_{jn} and β_{jn} are

$$\mu_{00} = -2t, \mu_{10} = -t^2, \mu_{20} = 2\sin(t), \alpha_{00} = 1, \alpha_{10} = 1, \alpha_{20} = \frac{1}{2}, \beta_{00} = -\frac{1}{2}, g(u) = u^3,$$

and

$$f(t) = e^{-\frac{t}{2}-2}\left(-t^4 - t^3 + \frac{3}{2}t^2 - \frac{3}{2} - \frac{3}{2}\right) + 2te^{-\frac{t}{2}-\frac{7}{4}}\left(-t^2 + \frac{5}{4}\right) - e^{-\frac{3t}{8}-6}\left(\frac{-t^2}{16} - \frac{t}{4} + 1\right)^3 - 2\sin(t)e^{\frac{-t}{4}-2}\left(\frac{-t^2}{4} - \frac{t}{2} + 1\right).$$

The nonlinear delay differential equation which results from the above coefficients is as follows

$$u^{(1)}(t) = -2tu\left(t - \frac{1}{2}\right) - t^2 u(t) + 2\sin(t)\, u\left(\frac{t}{2}\right) + u^3\left(\frac{t}{4}\right) + f(t), \tag{4.1}$$

with the initial condition

$$u(0) = e^{-2} \tag{4.2}$$

The exact solution of this problem is

$$u(t) = \left(-t^2 + 1 - t\right)e^{-\frac{t}{2}-2} \tag{4.3}$$

For $c_{\min} = 0.01$, $c_{\max} = 55c_{\min}$ and $N = 11$, we obtain

$$c = [3.1650e-01,\ 4.9304e-01, 1.2565e-01, 1.2084e-02, 4.8551e-01, 1.3697e-01,$$
$$1.4223e-01, 3.5610e-01, 1.7444e-01, 4.5584e-01, 4.8719e-01].$$

Numerical results obtained by RBF approximate solutions are summarized in Table 2 and for comparison plots of the exact and approximate solutions are shown in Fig. 1 .and plots of the error function $u_{exact} - u_{approximate}$ are shown in Fig. 2.

Table 2: Numerical results of example 1

t	$u_{approximate}$	u_{exact}	$Absolute\ error$
0.1	0.114872862081206	0.114574064193146	2.9880e-04
0.2	0.093370021989647	0.093066885472266	3.0314e-04
0.3	0.071727656837696	0.071055336241833	6.7232e-04
0.4	0.049248753035997	0.048753389679427	4.9536e-04
0.5	0.026973373742931	0.026349806140466	6.2357e-04
0.6	0.004580380729996	0.004010353748912	5.7003e-04
0.7	-0.017469350250484	-0.018120140820954	6.5079e-04
0.8	-0.039366943721498	-0.039915899447342	5.4896e-04
0.9	-0.060721050494990	-0.061268446414553	5.4740e-04
1	-0.081667610168024	-0.082084998623899	4.1739e-04

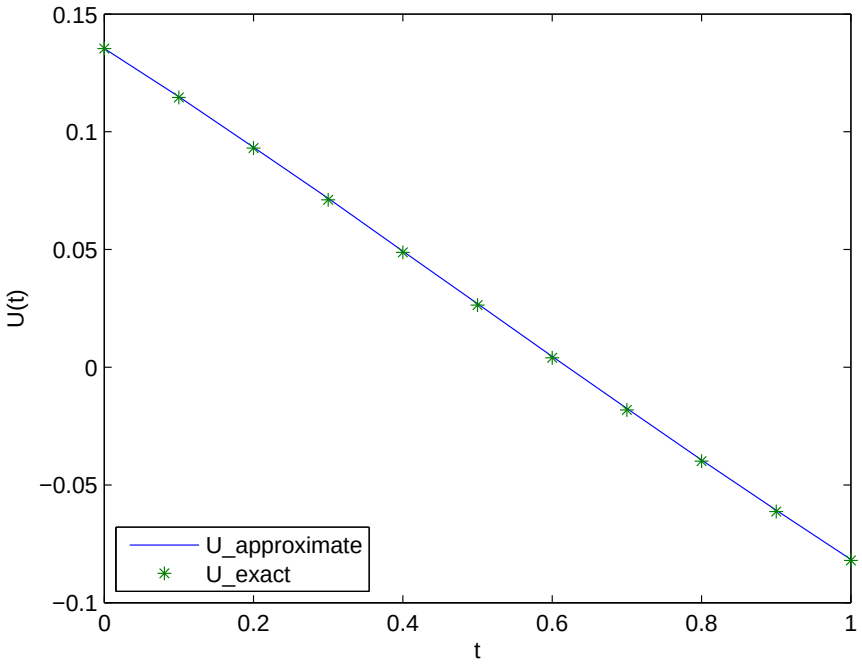

Fig. 1: The compared results for RBF solutions , and the exact solutions for Example 1

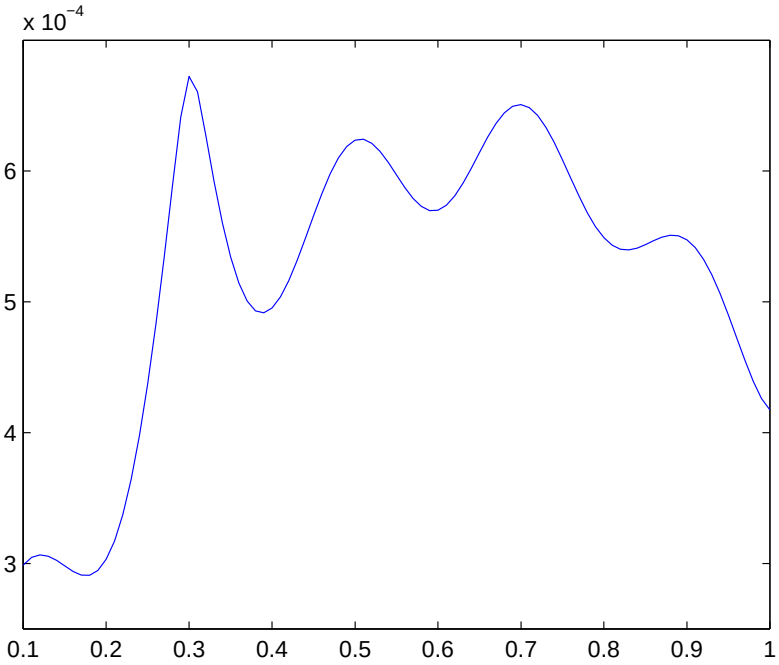

Fig. 2: The errors $u_{exact} - u_{approximate}$ in Example 1 .

Example 2: Consider the Eq.(1.1) with $m = 2, J = 1$ and the following nonzero coefficients μ_{jn}, α_{jn} and β_{jn}

$\mu_{00} = t - 1, \mu_{01} = e^{-t}, \mu_{10} = -2, \alpha_{00} = 1, \alpha_{01} = 1, \alpha_{10} = \frac{1}{3}, \beta_{01} = -\frac{1}{5}, g(u) = u^2$, and

$$f(t) = -\frac{1}{4}\sin^2\left(\frac{t}{3}\right) - \frac{1}{3}\sin\left(\frac{t}{3}\right)\cos\left(\frac{t}{2}\right) - \frac{1}{9}\cos^2\left(\frac{t}{2}\right)$$
$$+\left(-\frac{1}{2}\sin\left(\frac{t}{3}\right) - \frac{1}{3}\cos\left(\frac{t}{2}\right)\right)t + \frac{4}{9}\sin\left(\frac{t}{3}\right) + \frac{1}{4}\cos\left(\frac{t}{2}\right)$$
$$-e^{-t}\left(\frac{1}{6}\cos\left(\frac{t}{3} - \frac{1}{15}\right) - \frac{1}{6}\sin\left(\frac{t}{2} - \frac{1}{10}\right)\right) + \sin\left(\frac{t}{9}\right) + \frac{2}{3}\cos\left(\frac{t}{6}\right).$$

The nonzero coefficients c_{in} in the initial conditions are given as

$c_{00} = 3, c_{01} = 6, c_{10} = -2, c_{11} = 1$, And $\gamma_0 = 2, \gamma_1 = -\frac{1}{2}$.

These coefficients result in the following nonlinear delay differential problem

$$u^{(2)}(t) = e^{-t}u^{(1)}\left(t - \frac{1}{5}\right) + (t-1)u(t) - 2u\left(\frac{t}{3}\right) + u^2(t) + f(t), \qquad (4.8)$$

with the initial conditions

$$u(0) = \frac{1}{3}, u^{(1)}(0) = \frac{1}{6}. \tag{4.9}$$

The exact solution of this problem is

$$u(t) = \frac{1}{2}\sin\left(\frac{t}{3}\right) + \frac{1}{3}\cos\left(\frac{t}{2}\right)$$

For $c_{\min} = 0.01$, $c_{\max} = 297c_{\min}$ and $N = 11$, we obtain

$$c = [2.3980, 1.5012, 1.4592,\ 2.6061, 1.0553, 1.3404,$$
$$2.8620,\ 1.3520e-01, 2.8900, 5.7005e-01, 1.9847]$$

Numerical results obtained by these approximations are shown in Table 3. For comparison, we plot the exact and approximate solutions in Fig. 3 and plots of the error function $u_{exact} - u_{approximate}$ are shown in Fig. 4

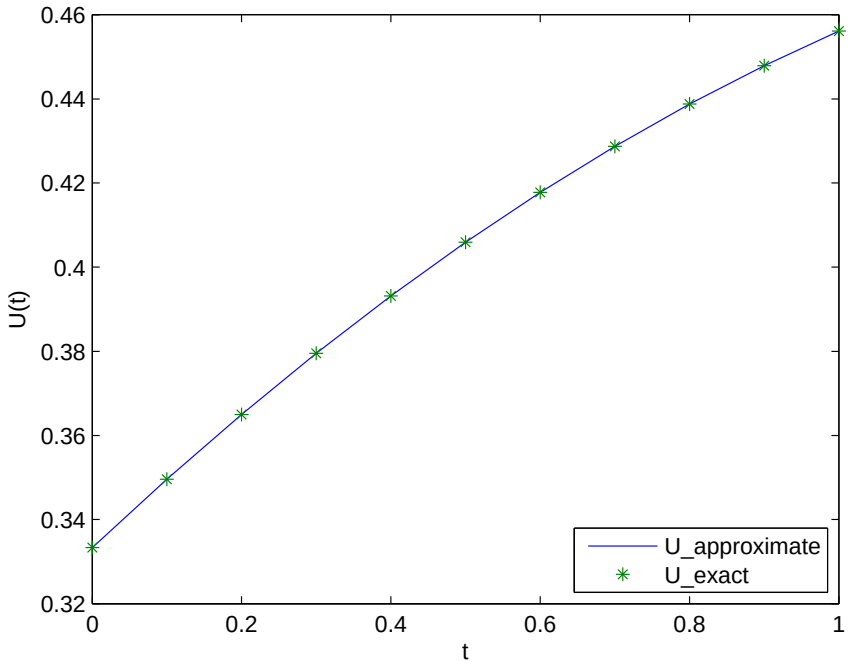

Fig. 3: The compared results for RBF solutions , and the exact solutions for Example 2.

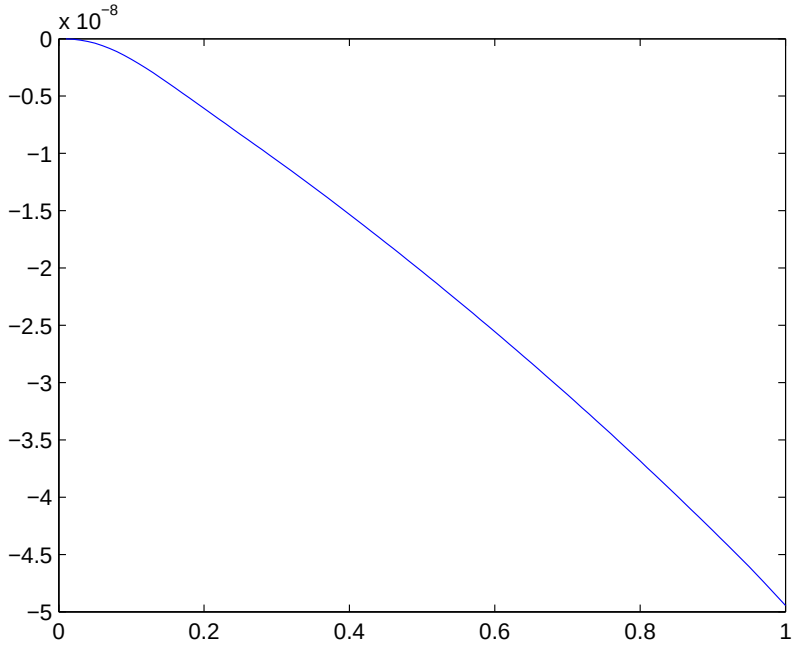

Fig. 4: The errors $u_{exact} - u_{approximate}$ in Example 2.

Table 3: Numerical results of Example 2.

t_i	$u_{approximate}$	u_{exact}	$Absolute\ error$
0.1	0.349580332091978	0.349580333883366	1.7914e-09
0.2	0.364976696473994	0.364976702554372	6.0804e-09
0.3	0.379507057032926	0.379507067635428	1.0603e-08
0.4	0.393158155267988	0.393158170591571	1.5324e-08
0.5	0.405918853268863	0.405918873583589	2.0315e-08
0.6	0.417780136215418	0.417780161772733	2.5557e-08
0.7	0.428735109377296	0.428735140432942	3.1056e-08
0.8	0.438778990068796	0.438779026906063	3.6837e-08
0.9	0.447909094540059	0.447909137448229	4.2908e-08
1	0.456124819569006	0.456124869028200	4.9459e-08

Example 3. We consider Eq. (1.1) with $m = 5, J = 1$ and the following cofficients

$$\mu_{04} = -1, \mu_{13} = 2t, \mu_{02} = -3, \mu_{01} = 1, \mu_{10} = 1, \alpha_{04} = 1, \alpha_{13} = \frac{1}{3}, \alpha_{02} = \frac{1}{4},$$
$$\alpha_{01} = 2, \alpha_{10} = \frac{1}{2}, \beta_{04} = 0, \beta_{13} = 0, \beta_{02} = 0, \beta_{01} = 0, \beta_{10} = 0,$$

and

$$f(t) = \frac{1}{32}e^{-t}(-32e^{\frac{3t}{4}}\cos(2t) - 24e^{\frac{7t}{4}}(63cs(t) - 16\sin(t)) - 8e^{\frac{5t}{6}}t(488\sin(\frac{4t}{3})$$
$$+191\cos(\frac{4t}{3}) - 65e^{\frac{t}{2}}(488\sin(4t) + 191\cos(4t) + 16(8\sin(8t) + \cos(8t)))$$

The linear delay differential equation which results from the above coefficients is as follows

$$u^{(5)}\left(t\right) = -u^{(4)}(t) + 2tu^{(3)}(\frac{t}{3}) - 3u^{(2)}(\frac{t}{4}) + u^{(1)}(2t) + u(\frac{t}{2}) + f\left(t\right), \qquad (4.10)$$

with the initial condition

$$u(0) = 1, u^{(1)}(0) = -\frac{1}{2}, u^{(2)}(0) = -\frac{63}{4}, u^{(3)}(0) = -\frac{191}{8}, u^{(4)}(0) = -\frac{3713}{16}, \qquad (4.11)$$

The exact solution of this problem is $u\left(t\right) = e^{-\frac{t}{2}}\cos(4t)$. For $c_{\min} = 2.4$, $c_{\max} = 10$ and $N = 18$, we obtain

$$c = \begin{bmatrix} 7.6775, 4.3516, 2.4742, 6.4454, 4.5234, 9.5913, 9.2890, 5.3844, 2.5889, \\ 7.5029, 8.7625, 9.7834, 2.8327, 5.8225, 6.8268, 7.6184, 7.8677, 7.3403 \end{bmatrix}$$

Numerical results obtained by these approximations are shown in Table 4. For comparison, we plot the exact and approximate solutions in Fig. 5 and plots of the error function $u_{approximate} - u_{exact}$ are shown in Fig. 6

Table 4: Numerical results of Example 3

t_i	$u_{approximate}$	u_{exact}	$Absolute\ error$
0.1	0.876139640808105	0.876140319255420	6.7845e-07
0.2	0.630407333374023	0.630406300014019	1.0334e-06
0.3	0.311876296997070	0.311884209597546	7.9126e-06
0.4	-0.023909568786621	-0.023906546883252	3.0219e-06
0.5	-0.324105262756348	-0.324095482175602	9.7806e-06
0.6	-0.546285629272461	-0.546274700289146	1.0929e-05
0.7	-0.663984298706055	-0.663972861336091	1.1437e-05
0.8	-0.669198036193848	-0.669177000067877	2.1036e-05
0.9	-0.571827888488770	-0.571798411458411	2.9477e-05
1	-0.396498680114746	-0.396454896579361	4.3784e-05

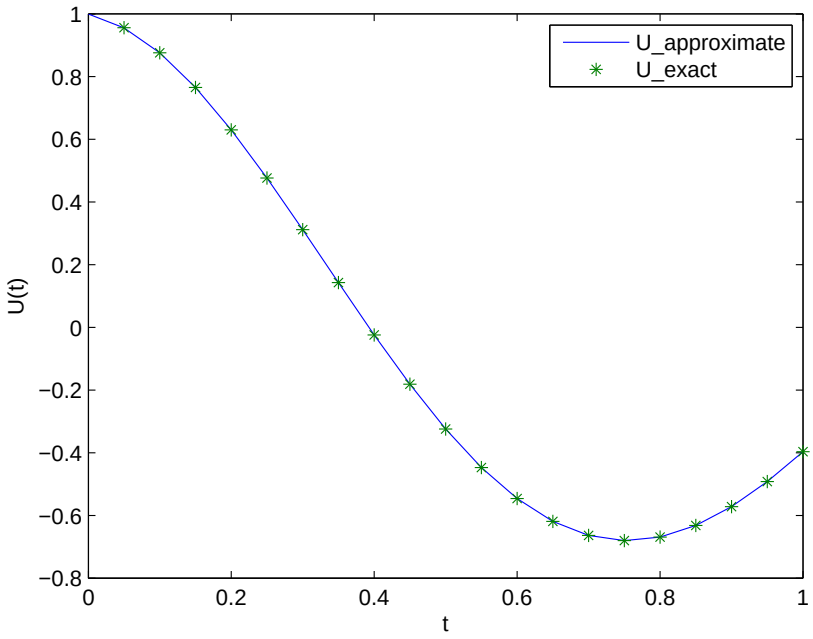

Fig. 5: The compared results for Approximate value $u_{approximate}$, and the exact value u_{exact} for Example 3.

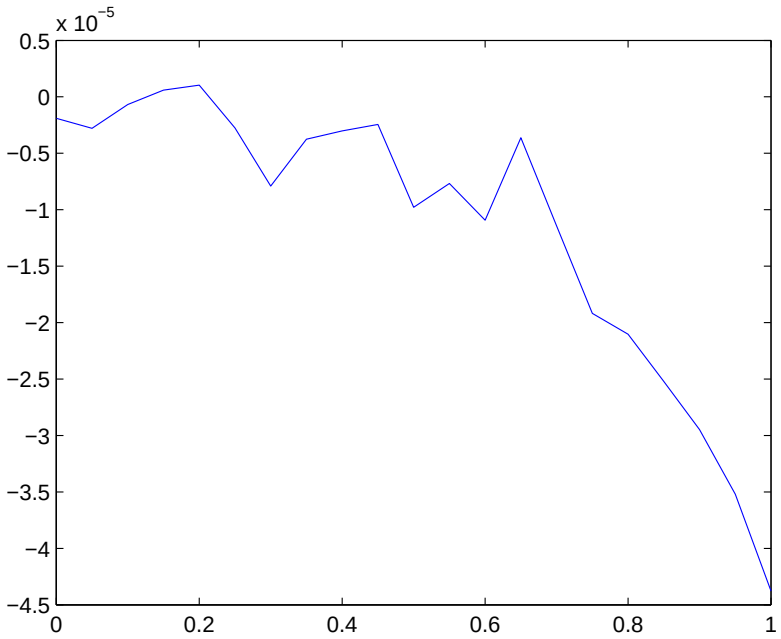

Fig. 6: The errors $u_{approximate} - u_{exact}$ in Example 3.

5. Conclusion

In this work, we proposed a numerical scheme to solve nonlinear DDEs by using collocation points and approximating the solution by using the multiquadric (MQ) radial basis function. The numerical solutions are compared with the exact solutions in two examples. The results show that this scheme provides good approximations to the solution of these nonlinear equations with high accuracy. This method which has good accuracy is useful in deal with linear and nonlinear system of delay differential equations.

Acknowledgments

We are very grateful to anonymous referees for their careful reading and valuable comments which led to the improvement of this paper.

References

[1] C. T. H. Baker, C. A. H. Paul and D. R. Wille, Issues in the Numerical Solution of Evolutionary Delay Differential Equations. Numerical Analysis Report No. 248. University of Manchester, 1994.

[2] R.D. Driver, Ordinary and Delay Differential Equations, New York: Springer-Verlag New York Inc., 1977.

[3] Y. Kuang, Delay Differential Equations with Applications in Population Dynamics, Academic Press,Boston, 1993.

[4] N. Macdonald, Biological Delay System: Linear Stability Theory, Cambridge University Press, 1989.

[5] G. Stépán, Retarded Dynamical ystems, Longman, London, 1989.

[6] A.C. Fowler, Mathematical Models in the Applied Sciences, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, 1997.

[7] A.C. Fowler, Asymptotic methods for delay equations, Journal of Engineering Mathematics, 53(2005)271–290.

[8] K. Gopalsamy, Stability and Oscillations in Delay Differential Equations of Population Dynamics, Kluwer Academic, Dordrecht, 1992.

[9] A. Halanay, Differential Equations: Stability, Oscillations, Time Lags, Academic Press, New York, 1966.

[10] V. Kolmanovskii and A. Myshkis. Applied theory of functional differential equations.Kluwer Academic Publishers, 1992.

[11] V.B. Kolmanovskii and V.R. Nosov. Stability of Functional Differential Equations, Academic Press Inc. 1986.

[12] J.K. Hale, S.M. Verduyn Lunel, Introduction to Functional Differential Equations, Springer, New York, Inc., 1993.

[13] N.Guglielmi, E. Hairer, Implementing Radau IIA Methods for stiff delay differential equations, Computing, 67(2001)1-12.

[14] R. Bellman, On the computational solution of differential-difference equations, J. Math. Anal. Appl., 2(1961)108-110.

[15] E. Lelarsmee, A. Ruehli, A. Sangiovanni-Vincentelli, The waveform relaxation method for time domain analysis of large scale integrated circuits, IEEE Transactions on Vol. CAD1(1982)131-145.

[16] A. Bellen, M. Zennaro, Adaptive integration of delay differential equations, Proceedings of the Workshop CNRS-NSF: Advances in Time Delay Systems, Paris, January 2003.

[17] A. Bellen, M. Zennaro, Numerical Methods for Delay Differential Equations, Oxford University Press, Oxford, 2003.

[18] R.L. Hardy, Multiquadric equations of topography and other irregular surfaces. J. Geophys. Res. 176(1971)1905-1915

[19] R. Franke, Scattered data interpolation: test of some methods. Math. Comput. 38(1982)181-200.

[20] C. A. Micchelli, Interpolation of scattered data: Distance matrices and conditionally positive definite functions, Constructive Approximation, 2(1986)11-22.

[21] E. J. Kansa, Multiquadrics- a scattered data approximation scheme with applications to computational fluid dynamics I: Surface approximations and partial derivative estimates.Computers and Mathematics with Applications, 19(1990)127-145.

[22] F. Shakeri, M. Dehghan, Solution of delay differential equations via a homotopy perturbation method, Math. Comput. Modeling 48 (2008) 486–498.

[23] M.A.Golberg, Some recentresults and proposals for the use of radial basis functions in the BEM, Eng. Anal. Bound. Elem., 23(1999)285-296.

[24] S. A. Sarra, E. J. Kansa, Multiquadric Radial Basis Function Approximation Methods for the Numerical Solution of Partial Differential Equations, Adv. Comput. Mech., in www.scottsarra.org. 2009.

[25] A. J. Khattak, S. I. A.Tirmizi, and S. U. Islam. Application of meshfree collocation method to a class of nonlinear partial differential equations, Eng. Anal. Bound. Elem., 33(2009)661–667.

[26] R. Askey, Radial Characteristic Functions (2nd ed.) Math. Research Centre, University of Wisconsin-Madison (1973) Tech. Report No. 1262.

[27] S. A. Sarra and D. Sturgill, A random variable shape parameter strategy for radial basis function approximation methods, Engineering Analysis with Boundary Elements, 33(2009)1239-1245.

[28] E. J. Kansa, Multiquadrics- a scattered data approximation scheme with applications to computa tional fluid-dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations, Computers & Mathematics with Applications, 19(1990)147-161.

[29] G. J. Moridis, E. J. Kansa, The Laplace transform multiquadrics method: a highly accurate scheme for the numerical solution of linear partial differential equations, Journal of Applied Science and Computations, 1(1994)375-407.

Matlab code (RBF_function)

```
if type==IMQ
  for k=1:N
    RBF(k,1)=1/(sqrt((t-x_center(k))^2+c(k)^2));
  end
elseif type==GA
  for k=1:N
    RBF(k,1)=exp(-((t-x_center(k))^2*c(k)^2));
  end
elseif type==MQ
  for k=1:N
```

```
        RBF(k,1)=sqrt((t-x_center(k))^2+c(k)^2)
      end
    elseif type==IQ
      for k=1:N
        RBF(k,1)=1/((t-x_center(k))^2+c(k)^2)
      end
    end
```

Matlab code for example1

```
    clear all
    clc
    format short e
    syms t
    Y=inline(exp((-t/2)-2)*(-t^4-t^3+(3/2)*t^2-(3/2)*t-(3/2))+2*t*exp((-t/2)-(7/4))*(-t^2+5/4)-exp((-
3*t/8)-6)*(-t^2/16-t/4+1)^3-2*sin(t)*exp(-t/4-2)*(-t^2/4-t/2+1));
    begin=0;% Begin of Intervals
    endof=1;% End of Intervals
    N=11;
    syms a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11
    a=[a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 ];
    x_center=linspace(begin,endof,N);
    %%
    %%use this varable shape parametre for this example
    %  Cmin=0.01
    %Cmax=55*Rmin
    %R=zeros(1,N);
    %  for j=1:N
    %  s_j =Rmin+(Rmax-Rmin)*rand(1,11);
    % R =s_j;
    %   end
    %  c=R

    %%
    %%. {For multiquadric }& N=11 %Cmin=0.01and Cmax=55*Rmin& Intervals [0,1]&RMS error 5.0372e-04
    c=[  3.1650e-01   4.9304e-01   1.2565e-01   1.2084e-02   4.8551e-01   1.3697e-01   1.4223e-01   3.5610e-
01   1.7444e-01   4.5584e-01   4.8719e-01];
    %%
    syms MQ IMQ GA IQ Me
    type=MQ;%{MQ IMQ GA IQ}
    RBF_function
    Coef_mat(1,1)=subs(a*RBF,t,begin);
    Y_mat(1,1)=exp(-2);
    for i=1:N
      linear(i,1)=diff(RBF(i))+2*t*(subs(RBF(i),t,t-1/2))+(t^2)*RBF(i)-2*sin(t)*(subs(RBF(i),t,t/2))
      nonlinear(i,1)=subs(RBF(i),t,t/4)
      end
    All_phrase=a*linear-(a*nonlinear)^3
    for j=2:N
      Coef_mat(j,1)=subs(All_phrase,t,x_center(j))
      Y_mat(j,1)=Y(x_center(j))
    end
```

```
%%% Newton iterative method
F=Coef_mat-Y_mat;
for i=1:N
  for j=1:N
    JACOBI(i,j)=diff(F(i),a(j))
  end
end
solu_0=[13;15;17;29;23;11;2;3;5;7;9];
ring_num=1;
for k=1:20
  for i=1:N
    Fun_mat(i,1)=vpa(subs(F(i),a,solu_0'))
    for j=1:N
      Jacobian(i,j)=vpa(subs(JACOBI(i,j),a,solu_0'))
    end
  end
  V=eye(N)/Jacobian
  solu_1=solu_0-V*Fun_mat;
  if norm(solu_1-solu_0,2)<0.000001
    break
  end
  ring_num=ring_num+1;
  solu_0=solu_1;
end
ring_num
solution_a=solu_1
% display
U_approximate=inline(solution_a'*RBF);% approximation polynomial of solution
t_plot=0.1:0.01:endof;
U_exact=inline(((-t^2+1-t)*(exp(-.5*t-2))));
plot(x_center,U_approximate(x_center),x_center,U_exact(x_center),'*')
title('');
xlabel('t');
ylabel('U(t)');
hleg1 = legend('U_approximate','U_exact');
set(hleg1,'Location','SouthWest')
set(hleg1,'Interpreter','none')
[siz1,siz]=size(x_center);
RMS_error=norm(U_approximate(x_center)-U_exact((x_center)),2)/sqrt(siz)
Absoluteeror=abs(U_approximate(x_center)-U_exact((x_center)))'
[siz1,siz]=size(t_plot);
RMS_error=norm(U_approximate(t_plot)-U_exact(t_plot),2)/sqrt(siz)
compare=[x_center',U_approximate(x_center)',U_exact(x_center)']
plot(t_plot,(U_approximate(t_plot)-U_exact((t_plot)))');%plot the error function
```

Matlab code for example 2

```
%%RBF for  nonlinear delay-Differential Equations
%Paper: RBF Technique for Solving the Nonlinear DDEs
%Exame 2
clear all
clc
format short e
```

```
syms t
Y=inline(-.25*(sin(t/3))^2-(1/3)*sin(t/3)*cos(t/2)-1/9*(cos(t/2))^2+(-1/2*sin(t/3)-
1/3*cos(t/2))*t+4/9*sin(t/3)+1/4*cos(t/2)-exp(-t)*(1/6*cos(t/3-1/15)-1/6*sin(t/2-
1/10))+sin(t/9)+2/3*cos(t/6));
begin=0;% begin of Intervals
endof=1;% end of  Intervals
N=11;
syms a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11
a=[a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11];
x_center=linspace(begin,endof,N);% collocation point
%%
%%use this varable shape parametre for this example
%  Rmin=0.01
%  Rmax=297*Rmin
%  R=zeros(1,N);
%  for j=1:N
%        s_j =Rmin+(Rmax-Rmin)*rand(1,11);
%   R =s_j;
%  end
%   c=R

%%
%{For multiquadric }& N=11 %Cmin=0.01 and Cmax=297*Rmin& Intervals [0,1]&N=11&RMS error
2.7072e-08..
c=[ 2.3980e+00  1.5012e+00  1.4592e+00  2.6061e+00  1.0553e+00  1.3404e+00  2.8620e+00
1.3520e-01  2.8900e+00  5.7005e-01  1.9847e+00];%Rmin=0.01 and Rmax=297*Rmin[0,1]N=11&10^-8
%%
syms MQ IMQ GA IQ Me
type=MQ;%{MQ IMQ GA IQ}
RBF_function
Coef_mat(1,1)=subs(a*RBF,t,begin);
Y_mat(1,1)=1/3;
Coef_mat(2,1)=subs(a*diff(RBF),t,begin);
Y_mat(2,1)=1/6;
for i=1:N
  linear(i,1)=diff(diff(RBF(i)))-exp(-t)*subs(diff(RBF(i)),t,t-1/5)-(t-1)*RBF(i)+2*subs((RBF(i)),t,t/3)
  nonlinear(i,1)=(RBF(i))
end
All_phrase=a*linear-(a*nonlinear)^2
for j=3:N
  Coef_mat(j,1)=subs(All_phrase,t,x_center(j))
  Y_mat(j,1)=Y(x_center(j))
end
%%
% Newton iterative method
F=Coef_mat-Y_mat;
for i=1:N
  for j=1:N
    JACOBI(i,j)=diff(F(i),a(j))
  end
end
solu_0=[13;17;19;23;29;11;2;3;5;7;9];
ring_num=1;
```

```
for k=1:20
  for i=1:N
    Fun_mat(i,1)=vpa(subs(F(i),a,solu_0'))
    for j=1:N
      Jacobian(i,j)=vpa(subs(JACOBI(i,j),a,solu_0'))
    end
  end
  V=eye(N)/Jacobian
  solu_1=solu_0-V*Fun_mat;
  if norm(solu_1-solu_0,2)<0.00001
    break
  end
  ring_num=ring_num+1;
  solu_0=solu_1;
end
ring_num;
solution_a=solu_1;
%%
U_approximate=inline(solution_a'*RBF);% approximation polynomial of solution
t_plot=0.01:0.01:endof;
U_exact=inline(0.5*sin(t/3)+(1/3)*cos(t/2));
plot(x_center,U_approximate(x_center),x_center,U_exact(x_center),'*');%plot the approximate and exact solutions
title('');
xlabel('t');
ylabel('U(t)');
hleg1 = legend('U_approximate','U_exact');
set(hleg1,'Location','SouthEast')
set(hleg1,'Interpreter','none')
[siz1,siz]=size(x_center);
RMS_error=norm(U_approximate(x_center)-U_exact((x_center)),2)/sqrt(siz)
Absoluteeror=abs(U_approximate(x_center)-U_exact((x_center)))'
compare=[x_center',U_approximate(x_center)',U_exact(x_center)']
% plot(t_plot,(U_approximate(t_plot)-U_exact((t_plot)))');%plot the error function
```

Matlab code for example 3

```
%%
%%RBF for  linear delay-Differential Equations
%Paper: RBF Technique for Solving the Nonlinear DDEs
%Exame 3
clear all
clc
format short e
syms t
Y=inline(1/32*exp(-t)*(-32*exp(3*t/4)*cos(2*t)-24*exp(7*t/8)*(63*cos(t)-16*sin(t))-
8*exp(5*t/6)*t*(488*sin(4*t/3)+191*cos(4*t/3))-
65*exp(t/2)*(488*sin(4*t)+191*cos(4*t))+16*(8*sin(8*t)+cos(8*t))));
begin=0;% begin of Intervals
endof=1;% end of  Intervals
N=18;
x_center=linspace(begin,endof,N);% collocation point
%%
```

```
%%use this varable shape parametre for this example
% Cmin=2.4
% Cmax=10
% R=zeros(1,N);
% for j=1:N
%    s_j =Cmin+(Cmax-Cmin)*rand(1,18);
%    R =s_j;
% end
% c=R
%%
%  {For multiquadric }& N=18 %Cmin=2.4 and Cmax=10 & Intervals [0,1] & RMS error = 1.8315e-05
c=[7.6775e+00  4.3516e+00  2.4742e+00  6.4454e+00  4.5234e+00  9.5913e+00  9.2890e+00
5.3844e+00  2.5889e+00  7.5029e+00  8.7625e+00  9.7834e+00 2.8327e+00  5.8225e+00  6.8268e+00
7.6184e+00  7.8677e+00  7.3403e+00];
%%
syms MQ IMQ GA IQ Me
type=MQ%{MQ IMQ GA IQ}
RBF_function
%%
%Initial conditions
for i=1:N
   D1=inline(RBF(i));
   Coef_mat(1,i)=D1(begin);
   D2=inline(diff(RBF(i)));
   Coef_mat(2,i)=D2(begin);
   D3=inline(diff(RBF(i),2));
   Coef_mat(3,i)=D3(begin);
   D4=inline(diff(RBF(i),3));
   Coef_mat(4,i)=D4(begin);
   D5=inline(diff(RBF(i),4));
   Coef_mat(5,i)=D5(begin);

end
Y_mat(1,1)=1;
Y_mat(2,1)=-0.5;
Y_mat(3,1)=(-63)/4;
Y_mat(4,1)=191/8;
Y_mat(5,1)=3713/16;
%%
for j=6:N
   for i=1:N
      A1=inline(diff(RBF(i),5));
      A2=inline(diff(RBF(i),4));
      A3=inline(2*t*subs(diff(RBF(i),3),t,t/3));
      A4=inline(3*subs(diff(RBF(i),2),t,t/4));
      A5=inline(subs(diff(RBF(i)),t,2*t));
      A6=inline(subs(RBF(i),t,t/2));
      Coef_mat(j,i)=A1(x_center(j))+A2(x_center(j))-A3(x_center(j))+A4(x_center(j))-A5(x_center(j))-
A6(x_center(j))
   end
   Y_mat(j,1)=Y(x_center(j));
end
landa=Coef_mat\Y_mat;
```

```
U_approximate=inline(landa'*RBF);% approximation polynomial of solution
xx_center=linspace(begin,endof,11);
t_plot=begin:0.05:endof;
U_exact=inline(exp(-t/2)*cos(4*t));
plot(t_plot,U_approximate(t_plot),t_plot,U_exact(t_plot),'*')
title('');
xlabel('t');
ylabel('U(t)');
hleg1 = legend('U_approximate','U_exact');
set(hleg1,'Location','northeast')
set(hleg1,'Interpreter','none')
%%
%Calculation  RMS error
[siz1,siz]=size(x_center);
RMS_error=norm(U_approximate(x_center)-U_exact((x_center)),2)/sqrt(siz)
%%
%Calculate the absolute error
Absoluteeror=abs(U_approximate(xx_center)-U_exact((xx_center)))'
%%
compare=[xx_center',U_approximate(xx_center)',U_exact(xx_center)']
% plot(t_plot,(U_approximate(t_plot)-U_exact((t_plot)))');%plot the error function
```