Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **9**, No. 3, pp. 667-679

DOI: 10.4208/aamm.2015.m909
June 2017

Cosine Radial Basis Function Neural Networks for Solving Fractional Differential Equations

Haidong Qu*

Department of Mathematics, Hanshan Normal University, Chaozhou, Guangdong 521041, China

Received 10 January 2015; Accepted (in revised version) 29 April 2016

Abstract. In this paper, we first apply cosine radial basis function neural networks to solve the fractional differential equations with initial value problems or boundary value problems. In the examples, we successfully obtained the numerical solutions for the fractional Riccati equations and fractional Langevin equations. The computer graphics and numerical solutions show that this method is very effective.

AMS subject classifications: 26A33, 60G22

Key words: Fractional Riccati equation, fractional Langevin equation, numerical solution, radial basis function neural network.

1 Introduction

Cosine radial basis function (RBF) neural networks which first introduced by M. Mary and B. Nicolaos are constructed by linear generator functions of a special form [1]. In [1], experimental data indicate that Cosine RBF outperform considerably conventional Cosine RBF neural networks with Gaussian radial basis functions. Cosine RBFs are also strong competitors to existing reformulated RBF models trained by gradient descent and conventional feedforward neural networks with sigmoid hidden units. The more detail about Cosine RBF neural networks, the reader can see [1].

In this paper, by using the method of Cosine RBF neural network, we first obtain the numerical solutions for the fractional Riccati equation of the following form

$$ {}^{c}D^{\alpha}_{0+}y(x)=f(x)+g(x)y+h(x)y^2, \quad 0<x\leq 1, \quad 0<\alpha\leq 1, \tag{1.1}$$

with initial condition as follows

$$y(0)=c,$$

*Corresponding author.
Email: qhaidong@163.com (H. D. Qu)

 ©2017 Global Science Press

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

where ${}^cD^{\alpha}_{0+}$ is the Caputo fractional derivative of order α, and c is a constant. In case of $\alpha=1$, the fractional equation reduces to the classical Riccati differential equation, and this equation of various forms paly a key role in many fields of robust control, systems identification, dynamic games, calculus of variations, network synthesis, diffusion problems and financial mathematics [2–4]. The value $\alpha=1/2$ is especially popular. This is because in classical fractional calculus, many of the model equations developed used order of $\alpha=1/2$ [5]. Recently, some effective methods for solving fractional Riccati equations have been presented, such as homotopy analysis method (HAM) [6], homotopy perturbation method (HPM) [7], Decomposition method [8]. All these methods tried to obtain the series solutions for the equations. In this paper, we presented a different method for solving fractional Riccati equation, which were constructed from cosine functions with adjustable parameters, and by adjusting the parameters repeatedly to minimize an appropriate error function, we obtain the numerical solutions when the error values are small enough.

We then obtain the numerical solution for the following three-point boundary value problem of fractional Langevin equation with two different fractional orders

$$ {}^cD^{\beta}_{0+}({}^cD^{\alpha}_{0+}+\lambda)y(x)=f(x,y(x)), \quad 0<x\leq 1, \quad 0<\alpha\leq 1, \quad 1<\beta\leq 2, \tag{1.2}$$

satisfying the boundary conditions

$$y(0)=c_1, \quad y(\eta)=c_2, \quad y(1)=c_3, \quad 0<\eta<1,$$

where ${}^cD^{\alpha}_{0+}$ is the Caputo fractional derivative, $\lambda>0$, c_1, c_2 and c_3 are positive real numbers. The classical Langevin equation was first formulated by Langevin to describe Brownian motion which generated by collisions with the molecules of the fluid, and now it has been used in many areas, such as modelling the evacuation processes [9], photoelectron counting [10], analyzing the stock market [11], etc. However, There are still some phenomena, for example the anomalous diffusion, power-law phenomena, long-range interaction, etc, which can not be accurately described by classical Langevin equation. Therefore, fractional Langevin equations have been presented to model the above behaviors. More detail the reader can see [12–16]. The existence and uniqueness of solution of Langevin equation involving two fractional orders in different intervals ($0<\alpha\leq 1$, $1<\beta\leq 2$) have been studied in [17].

2 Definitions and lemma

Definition 2.1 (see [18])**.** The Riemann-Liouville fractional integral of order $\alpha\in R$, $\alpha>0$ of a function $f(x)\in C_{\mu}$, $\mu\geq -1$ is defined as

$$I^{\alpha}_{0+}f(x):=\frac{1}{\Gamma(\alpha)}\int_0^x \frac{f(t)dt}{(x-t)^{1-\alpha}}, \quad x>0.$$

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

Definition 2.2 (see [18]). The Riemann-Liouville and Caputo fractional derivatives of order $\alpha \in R$, $\alpha > 0$ are given by

$$D_{0+}^{\alpha} f(x) := \left(\frac{d}{dx}\right)^n I_{0+}^{n-\alpha} f(x) = \frac{1}{\Gamma(n-\alpha)} \left(\frac{d}{dx}\right)^n \int_0^x \frac{f(t)dt}{(x-t)^{\alpha-n+1}},$$

and

$${}^{c}D_{0+}^{\alpha} f(x) := I_{0+}^{n-\alpha} \left(\frac{d}{dx}\right)^n f(x) = \frac{1}{\Gamma(n-\alpha)} \int_0^x \frac{(\frac{d}{dt})^n f(t)dt}{(x-t)^{\alpha-n+1}},$$

where $n = [\alpha]+1$, $x > 0$.

Definition 2.3 (see [18]). The classical Mittag-Leffler function is defined by

$$E_{\alpha}(x) := \sum_{k=0}^{\infty} \frac{x^k}{\Gamma(\alpha k+1)}, \quad x \in C, \quad \alpha > 0.$$

The generalized Mittag-Leffler function is defined by

$$E_{\alpha,\beta}(x) := \sum_{k=0}^{\infty} \frac{x^k}{\Gamma(\alpha k+\beta)}, \quad x,\beta \in C, \quad \alpha > 0.$$

Definition 2.4. The functions $\mathrm{Sin}_{\alpha,\beta}(x)$, $\mathrm{Cos}_{\alpha,\beta}(x)$ $(x,\beta \in C, \alpha > 0)$ are defined by

$$\mathrm{Sin}_{\alpha,\beta}(x) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{x^{2k-1}}{\Gamma(\alpha(2k-1)+\beta)}, \quad \mathrm{Cos}_{\alpha,\beta}(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{\Gamma(\alpha(2k)+\beta)}.$$

Easily, we have

$$\mathrm{Sin}_{\alpha,\beta}(x) = \frac{E_{\alpha,\beta}(ix) - E_{\alpha,\beta}(-ix)}{2i}, \quad \mathrm{Cos}_{\alpha,\beta}(x) = \frac{E_{\alpha,\beta}(ix) + E_{\alpha,\beta}(-ix)}{2}. \tag{2.1}$$

Lemma 2.1. *If* $\mathrm{Sin}_{\alpha,\beta}(x)$ *and* $\mathrm{Cos}_{\alpha,\beta}(x)$ *are defined as in Definition 2.4, then we have*

$$\begin{aligned} {}^{c}D_{0+}^{\alpha}(x-0)^{\beta-1}\mathrm{Sin}_{\mu,\beta}[\lambda(x-0)^{\mu}] &= (x-0)^{\beta-\alpha-1}\mathrm{Sin}_{\mu,\beta-\alpha}[\lambda(x-0)^{\mu}], \\ {}^{c}D_{0+}^{\alpha}(x-0)^{\beta-1}\mathrm{Cos}_{\mu,\beta}[\lambda(x-0)^{\mu}] &= (x-0)^{\beta-\alpha-1}\mathrm{Cos}_{\mu,\beta-\alpha}[\lambda(x-0)^{\mu}]. \end{aligned}$$

In particular, when $\beta = 1$, $\mu = 1$, *we have*

$${}^{c}D_{0+}^{\alpha}\mathrm{Sin}_{1,1}[\lambda(x-0)] = {}^{c}D_{0+}^{\alpha}\sin[\lambda(x-0)] = (x-0)^{-\alpha}\mathrm{Sin}_{1,1-\alpha}[\lambda(x-a)],$$

and

$${}^{c}D_{0+}^{\alpha}\mathrm{Cos}_{1,1}[\lambda(x-0)] = {}^{c}D_{0+}^{\alpha}\cos[\lambda(x-0)] = (x-0)^{-\alpha}\mathrm{Cos}_{1,1-\alpha}[\lambda(x-a)].$$

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

3 Numerical solutions for fractional Riccati equation

3.1 Illustration of the method

To describe the method, we consider the problem (1.1) with initial condition $y(0)=c$. The j-th trial solution satisfying the initial condition is written as

$$y_j(x)=\sum_{i=1}^{n_1} w_{i,j}\cos(ix)+\Big(c-\sum_{i=1}^{n_1} w_{i,j}\Big)\cos((n_1+1)x),$$

where n_1 represents the number of neurons and $w_{i,j}$ are unknown weights of the network determined in training procedures to reduce the error function.

Let the interval $(0,1)$ be partitioned into n_2 uniform subintervals, and x_k, (k=1,2,$\cdots$,n_2) denote the right end points of the subintervals, where n_2 represents the number of sample points. The error function is defined by

$$J=\frac{1}{2}\|E_j\|_2^2=\frac{1}{2}\sum_{k=1}^{n_2} e_j(k),\quad E_j=(e_j(1),e_j(2),\cdots,e_j(n_2))^T, \tag{3.1}$$

where $\|\cdot\|_2$ is Euclidean norm, and

$$\begin{aligned} e_j(k)&=f(x_k)+g(x_k)y_j(x_k)+h(x_k)y_j^2(x_k)-{}^cD_{0+}^{\alpha}y_j(x_k)\\ &=f(x_k)+g(x_k)y_j(x_k)+h(x_k)y_j^2(x_k)\\ &\quad -x_k^{-\alpha}\Big(\sum_{i=1}^{n_1} w_{i,j}\mathrm{Cos}_{1,1-\alpha}(ix_k)+\Big(C-\sum_{i=1}^{n_1} w_{i,j}\Big)\mathrm{Cos}_{1,1-\alpha}((n_1+1)x_k)\Big), \end{aligned}$$

$k=1,2,\cdots,n_2$, then we can adjust the weights $w_{i,j}$ by the following equation

$$w_{i,j+1}=w_{i,j}+\Delta w_{i,j},\quad i=1,2,\cdots,n_1,$$

where

$$\begin{aligned} \Delta w_{i,j}&=-\mu\frac{\partial J}{\partial w_{i,j}}=-\mu\sum_{k=1}^{n_2}\frac{\partial J}{\partial e_j(k)}\frac{\partial e_j(k)}{\partial w_{i,j}}\\ &=-\mu\sum_{k=1}^{n_2} e_j(k)(g(x_k)+2h(x_k)y_j(x_k))(\cos(x_k)-\cos((n_1+1)x_k))\\ &\quad -(x_k)^{-\alpha}e_j(k)(\mathrm{Cos}_{1,1-\alpha}x_k-Cos_{1,1-\alpha}((n_1+1)x_k)). \end{aligned}$$

The algorithm is given in the following steps:

Step 1 Initialization. Select n decile points on the interval $[0,1]$ as the set of sample points and randomly assign real value for $\omega_{i,j}$.

Step 2 Calculate. Calculate the error function J and incremental weight $\Delta\omega_{i,j}$.

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

Step 3 Termination Criteria. If the error function $|J|>0.001$ and training times less than the preset value, adjust the weights according to the formula $\omega_{i,j+1}=\omega_{i,j}+\Delta\omega_{i,j}$, then go to Step 2 else continue.

Step 4 Output. Calculate $y_j(x)$, and output numerical solution $y_j(x)$ and network weights $\omega_{i,j}$.

3.2 Examples

Example 3.1. We consider the following Riccati equation

$$ {}^{c}D^{\alpha}_{0+}y(x)=-x^3-x^5+\frac{2}{\Gamma(3-\alpha)}x^{2-\alpha}+xy+xy^2,\quad 0<x\leq 1,\quad 0<\alpha\leq 1, \tag{3.2}$$

with condition $y(0)=0$. By the Definition 2.2, we easily calculate the exact solution for the problem is $y(x)=x^2$. Then, we set the parameters $\mu=0.001$, $n_1=7$, $n_2=7$, by training the neural network about 12000 times, we obtain the numerical solutions for $\alpha=1$, $\alpha=0.9$ and $\alpha=0.8$, respectively. To achieve the same accuracy for different values of α, we found that the number of training time increase with the decrease of parameters α. The numerical solutions and accuracy for above three cases are provided in Table 2, and the weights after training for the network are given in Table 1. Fig. 1 and Fig. 2 show that both sample points and check points are in well agree with the exact solution.

Example 3.2. We consider another Riccati equation as the following form

$$\begin{aligned} {}^{c}D^{\alpha}_{0+}y(x)=&-\sin(x)\cos(x)-\cos^3(x)+x^{-\alpha}\mathrm{Cos}_{1,1-\alpha}(x)\\ &+\sin(x)y+\cos(x)y^2,\quad 0<x\leq 1,\quad 0<\alpha\leq 1, \end{aligned}$$

with condition $y(0)=1$. From the Lemma 2.1, we calculate the exact solution for the problem is $y(x)=\cos(x)$. Then, we set the parameters $\mu=0.001$, $n_1=7$, $n_2=7$, by training the neural network 15000 times, we obtain the numerical solutions for $\alpha=1$, $\alpha=0.9$ and $\alpha=0.8$, respectively. These adaptive weights are restricted to real numbers between 0 to

Table 1: Weights obtained along with the solution of Examples 3.1 and 3.2.

α	Example 3.1			Example 3.2		
	1	0.9	0.8	1	0.9	0.8
w_1	0.4821	0.7192	0.5299	0.8018	0.8166	0.9340
w_2	-0.0025	-0.5077	-0.0243	0.5268	0.4109	0.1006
w_3	-0.5272	-0.1919	-0.7359	-0.5440	-0.3423	0.0239
w_4	-0.3183	-0.3574	0.0209	0.2116	0.1674	-0.1275
w_5	0.3926	0.4306	0.2762	0.1189	-0.1478	0.0619
w_6	0.2687	0.0910	0.0983	-0.2005	0.1910	0.0507
w_7	-0.4555	-0.2903	-0.2580	0.1101	-0.1352	-0.0657

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

Table 2: Exact solution, approximate solution and accuracy for Example 3.1.

x	α x^2	Numerical solution			Accuracy		
		1	0.9	0.8	1	0.9	0.8
0.1	0.01	0.0168	0.0176	0.0168	10^{-3}	10^{-3}	10^{-3}
0.25	0.0625	0.0602	0.0631	0.0602	10^{-3}	10^{-4}	10^{-4}
0.4	0.16	0.1582	0.1604	0.1582	10^{-3}	10^{-4}	10^{-3}
0.55	0.3025	0.3042	0.3051	0.3043	10^{-3}	10^{-3}	10^{-3}
0.7	0.49	0.4821	0.4858	0.4821	10^{-3}	10^{-3}	10^{-2}
0.85	0.7225	0.7079	0.7143	0.7079	10^{-2}	10^{-3}	10^{-2}
1	1	0.9928	0.9972	0.9927	10^{-4}	10^{-3}	10^{-2}

1, and the weights after adjustment for the network are given in Table 1. Fig. 3 and Fig. 4 show that both sample points and check points are in well agree with the exact solution. It also be seen from Table 3 that the numerical solutions we got have a very high accuracy.

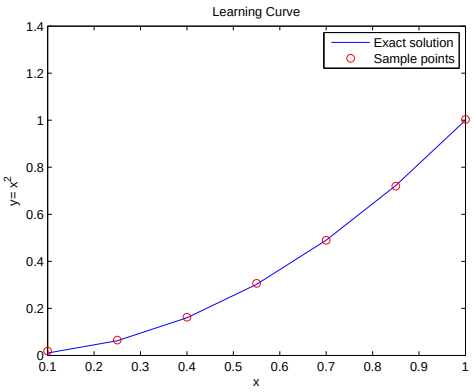


Figure 1: The learning curve for Example 3.1.

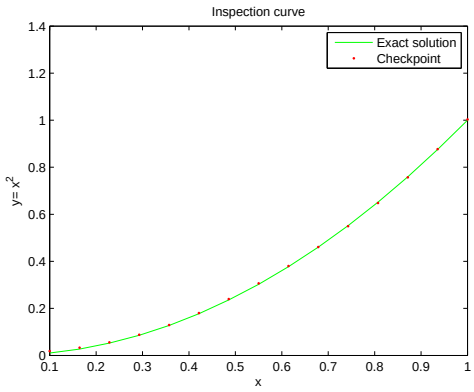


Figure 2: The inspection curve for Example 3.1.

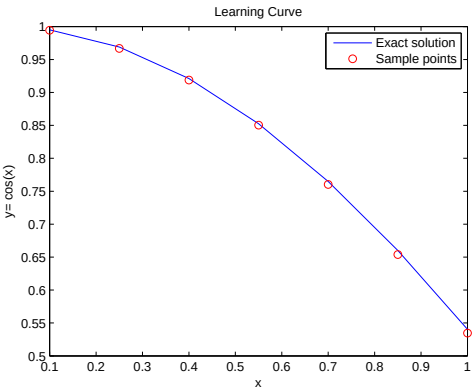


Figure 3: The learning curve for Example 3.2.

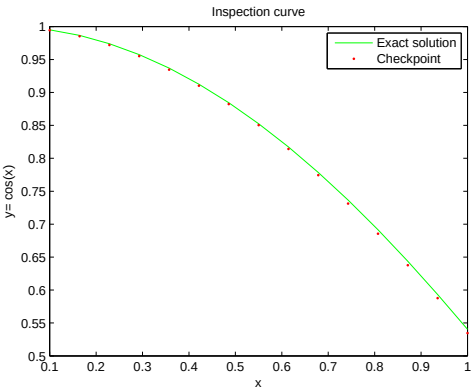


Figure 4: The inspection curve for Example 3.2.

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

Table 3: Exact solution, approximate solution and accuracy for Example 3.2.

x	${}^{\alpha}\cos(x)$	Numerical solution			Accuracy		
		1	0.9	0.8	1	0.9	0.8
0.1	0.9950	0.9951	0.9944	0.9945	10^{-4}	10^{-4}	10^{-4}
0.25	0.9689	0.9696	0.9666	0.9675	10^{-4}	10^{-3}	10^{-3}
0.4	0.9210	0.9218	0.9184	0.9195	10^{-4}	10^{-3}	10^{-3}
0.55	0.8525	0.8532	0.8497	0.8508	10^{-4}	10^{-3}	10^{-3}
0.7	0.7648	0.7662	0.7595	0.7615	10^{-3}	10^{-3}	10^{-3}
0.85	0.6600	0.6618	0.6527	0.6555	10^{-3}	10^{-3}	10^{-3}
1	0.5403	0.5420	0.5334	0.5360	10^{-3}	10^{-3}	10^{-3}

Fig. 5 show that the error value can decline rapidly in the first about 500 training times, and slowly after 1000 training times. So we need long time to train the network to reduce the error function. Both the Example 3.1 and Example 3.2 are spent about 1800s. In the process of compiling the program, we calculate $\mathrm{Cos}_{\alpha,\beta}(x)$ and $\mathrm{Sin}_{\alpha,\beta}(x)$ by the formulas

$$\mathrm{Cos}_{\alpha,\beta}(x)=\frac{1}{2}(E_{\alpha,\beta}(ix)+E_{\alpha,\beta}(-ix))$$

and

$$\mathrm{Sin}_{\alpha,\beta}(x)=\frac{1}{2i}(E_{\alpha,\beta}(ix)-E_{\alpha,\beta}(-ix)),$$

respectively. The Matlab code for Mittag-Leffler function was provided by Prof. Dr. I. Podlubny [19].

4 Numerical solutions for fractional Langevin equation

4.1 Illustration of the method and application

To describe this method, we consider the problem (1.2) with the boundary conditions $y(0)=c_1$, $y(\eta)=c_2$, $y(1)=c_3$. The j-th trial solution satisfying the boundary conditions is written as

$$\begin{aligned}y_j(x)=&\sum_{i=1}^{n_1}P_i^j\cos(ix)-\frac{(1-x)(\eta-x)}{\eta}\Big(\sum_{i=1}^{n_1}P_i^j-c_1\Big)\\&-\frac{x(1-x)}{\eta(1-\eta)}\Big(\sum_{i=1}^{n_1}P_i^j\cos(i\eta)-c_2\Big)-\frac{x(\eta-x)}{\eta-1}\Big(\sum_{i=1}^{n_1}P_i^j\cos(i)-c_3\Big),\end{aligned}$$

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

where n_1 represents the number of neurons and P_i^j are unknown weights of the network determined in training procedures. Then from the Lemma 2.1, we have

$$
\begin{aligned}
{}^cD_{0+}^{\beta+\alpha}y_j(x)=&\Big(\sum_{i=1}^{n_1}P_i^jCos_{1,1-\beta-\alpha}(ix)\Big)x^{-\beta-\alpha}\\
&-\Big(\frac{2x^{2-\beta-\alpha}}{\eta\Gamma(3-\beta-\alpha)}-\frac{(\eta+1)x^{1-\beta-\alpha}}{\eta\Gamma(2-\beta-\alpha)}+\frac{x^{-\beta-\alpha}}{\Gamma(1-\beta-\alpha)}\Big)\Big(\sum_{i=1}^{n_1}P_i^j-c_1\Big)\\
&-\Big(-\frac{2x^{2-\beta-\alpha}}{\eta(1-\eta)\Gamma(3-\beta-\alpha)}+\frac{x^{1-\beta-\alpha}}{\eta(1-\eta)\Gamma(2-\beta-\alpha)}\Big)\Big(\sum_{i=1}^{n_1}P_i^j\cos(i\eta)-c_2\Big)\\
&-\Big(-\frac{2x^{2-\beta-\alpha}}{(\eta-1)\Gamma(3-\beta-\alpha)}+\frac{\eta x^{1-\beta-\alpha}}{(\eta-1)\Gamma(2-\beta-\alpha)}\Big)\Big(\sum_{i=1}^{n_1}P_i^j\cos(i)-c_3\Big),\\
{}^cD_{0+}^{\beta}y_j(x)=&\Big(\sum_{i=1}^{n_1}P_i^jCos_{1,1-\beta}(ix)\Big)x^{-\beta}\\
&-\Big(\frac{2x^{2-\beta}}{\eta\Gamma(3-\beta)}-\frac{(\eta+1)x^{1-\beta}}{\eta\Gamma(2-\beta)}+\frac{x^{-\beta}}{\Gamma(1-\beta)}\Big)\Big(\sum_{i=1}^{n_1}P_i^j-c_1\Big)\\
&-\Big(-\frac{2x^{2-\beta}}{\eta(1-\eta)\Gamma(3-\beta)}+\frac{x^{1-\beta}}{\eta(1-\eta)\Gamma(2-\beta)}\Big)\Big(\sum_{i=1}^{n_1}P_i^j\cos(i\eta)-c_2\Big)\\
&-\Big(-\frac{2x^{2-\beta}}{(\eta-1)\Gamma(3-\beta)}+\frac{\eta x^{1-\beta}}{(\eta-1)\Gamma(2-\beta)}\Big)\Big(\sum_{i=1}^{n_1}P_i^j\cos(i)-c_3\Big).
\end{aligned}
$$

Let the interval $(0,1)$ be partitioned into n_2 uniform subintervals, and x_k, $(k=1,2,\cdots,n_2)$ denote the right end points of the subintervals. Then

$$e_j(k)={}^cD_{0+}^{\beta+\alpha}y_j(x_k)+\lambda{}^cD_{0+}^{\beta}y_j(x_k)-f(x_k,y_j(x_k)),\quad k=1,2,\cdots,n_2.$$

Suppose

$$J=\frac{1}{2}\|E_j\|_2^2=\frac{1}{2}\sum_{k=1}^{n_2}e_j^2(k),\quad E_j=(e_j(1),e_j(2),\cdots,e_j(n_2))^T,$$

where $\|\cdot\|_2$ is Euclidean norm, let μ denotes the learning rate, and we have

$$
\begin{aligned}
\Delta P_i^j=&-\mu\frac{\partial J}{\partial P_i^j}=-\mu\sum_{k=1}^{n_2}\frac{\partial J}{\partial e_j(k)}\frac{\partial e_j(k)}{\partial P_i^j}\\
=&-\mu\sum_{k=1}^{n_2}e_j(k)\Big\{\mathrm{Cos}_{1,1-\beta-\alpha}(ix_k)x_k^{-\beta-\alpha}-\Big(\frac{x_k^{2-\beta-\alpha}}{\eta\Gamma(3-\beta-\alpha)}-\frac{(\eta+1)x_k^{1-\beta-\alpha}}{\eta\Gamma(2-\beta-\alpha)}+\frac{x_k^{-\beta-\alpha}}{\eta\Gamma(1-\beta-\alpha)}\Big)\\
&-\Big(-\frac{x_k^{2-\beta-\alpha}}{\eta(1-\eta)\Gamma(3-\beta-\alpha)}+\frac{x_k^{1-\beta-\alpha}}{\eta(1-\eta)\Gamma(2-\beta-\alpha)}\Big)\cos(i\eta)\\
&-\Big(-\frac{x_k^{2-\beta-\alpha}}{(\eta-1)\Gamma(3-\beta-\alpha)}+\frac{\eta x_k^{1-\beta-\alpha}}{(\eta-1)\Gamma(2-\beta-\alpha)}\Big)\cos(i)
\end{aligned}
$$

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

$$+\lambda \mathrm{Cos}_{1,1-\beta}(ix_k)x_k^{-\beta}-\lambda\Big(\frac{x_k^{2-\beta}}{\eta\Gamma(3-\beta)}-\lambda\frac{(\eta+1)x_k^{1-\beta}}{\eta\Gamma(2-\beta)}+\frac{x_k^{-\beta}}{\eta\Gamma(1-\beta)}\Big)$$
$$-\lambda\Big(-\frac{x_k^{2-\beta}}{\eta(1-\eta)\Gamma(3-\beta)}+\frac{x_k^{1-\beta}}{\eta(1-\eta)\Gamma(2-\beta)}\Big)\cos(i\eta)$$
$$-\lambda\Big(-\frac{x_k^{2-\beta}}{(\eta-1)\Gamma(3-\beta)}+\frac{\eta x_k^{1-\beta}}{(\eta-1)\Gamma(2-\beta)}\Big)\cos(i)$$
$$-f_y\Big(\cos(ix_k)-\frac{(1-x_k)(\eta-x_k)}{\eta}-\frac{x_k(1-x_k)}{\eta(1-\eta)}\cos(i\eta)-\frac{x_k(\eta-x_k)}{\eta-1}\cos(i)\Big)\Bigg\}.$$

Finally, we adjust the parameters P_i^j according to the formula

$$P_i^{j+1}=P_i^j+\Delta P_i^j,\quad i=1,2,\cdots,n_1.$$

4.2 Examples

Example 4.1. We consider the following nonlinear fractional Langevin equation

$${}^cD_{0+}^{\beta}({}^cD_{0+}^{\alpha}+\lambda)y(x)=x^4+\frac{2}{\Gamma(3-\beta-\alpha)}x^{2-\beta-\alpha}+\lambda\frac{2}{\Gamma(3-\beta)}x^{2-\beta}-y^2(x),$$

with conditions $y(0)=0$, $y(0.5)=0.25$, $y(1)=1$. The exact solution is $y(x)=x^2$. We set the parameters $\mu=0.001$, $n_1=7$, $n_2=10$, then train the neural network 1000 times, the weights for the network are given in Table 4, The numerical solutions and accuracy for $\alpha=0.9$, $\beta=1.8$ and $\alpha=0.8$, $\beta=1.9$ are provided in Table 5. Fig. 6 show that sample points are on the exact solution curve after training is completed. Then we check the other points are also match well with the exact solution (see Fig. 7). From Fig. 8 we see the error value decrease rapidly within the first 200 training times.

Table 4: Weights obtained along with the solution of Examples 4.1 and 4.2.

α,β	Example 4.1		Example 4.2	
	0.9,1.8	0.8,1.9	0.9,1.8	0.8,1.9
P_1	0.1461	-0.0979	-0.2699	0.1549
P_2	-0.0576	-0.0271	0.3786	0.2331
P_3	-0.2427	0.0452	0.0720	-0.5760
P_4	0.0635	0.1926	-0.4095	0.2171
P_5	0.1094	-0.3347	0.4042	0.0966
P_6	-0.1099	0.2106	-0.2041	-0.1344
P_7	0.0336	-0.0572	0.0476	0.0412

Example 4.2. We consider the following linear fractional Langevin equation

$${}^cD_{0+}^{\beta}({}^cD_{0+}^{\alpha}+\lambda)y(x)=\sin(x)+x^{-\beta-\alpha}\mathrm{Sin}_{1,1-\beta-\alpha}(x)+\lambda x^{-\beta}\mathrm{Sin}_{1,1-\beta}(x)-y(x)$$

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

Table 5: Exact solution, approximate solution and accuracy for Example 4.1.

x	α,β x^2	Numerical solution		Accuracy	
		0.9,1.8	0.8,1.9	0.9,1.8	0.8,1,9
0.1	0.01	0.0093	−0.0014	10^{-4}	10^{-2}
0.2	0.04	0.0383	0.0240	10^{-2}	10^{-2}
0.3	0.09	0.0877	0.0755	10^{-3}	10^{-4}
0.4	0.16	0.1584	0.1518	10^{-3}	10^{-3}
0.5	0.25	0.2500	0.2500	10^{-4}	10^{-4}
0.6	0.36	0.3616	0.3671	10^{-3}	10^{-3}
0.7	0.49	0.4922	0.5007	10^{-3}	10^{-2}
0.8	0.64	0.6417	0.6501	10^{-3}	10^{-2}
0.9	0.81	0.8108	0.8163	10^{-4}	10^{-3}
1	1	1	1	10^{-4}	10^{-4}

Table 6: Exact solution, approximate solution and accuracy for Example 4.2.

x	α,β $\sin(x)$	Numerical solution		Accuracy	
		0.9,1.8	0.8,1.9	0.9,1.8	0.8,1,9
0.1	0.0998	0.0979	0.0988	10^{-3}	10^{-4}
0.2	0.1986	0.1952	0.1984	10^{-3}	10^{-4}
0.3	0.2955	0.2917	0.2965	10^{-3}	10^{-3}
0.4	0.3894	0.3870	0.3906	10^{-3}	10^{-3}
0.5	0.4794	0.4794	0.4794	10^{-4}	10^{-4}
0.6	0.5646	0.5667	0.5628	10^{-3}	10^{-3}
0.7	0.6442	0.6468	0.6413	10^{-3}	10^{-3}
0.8	0.7173	0.7190	0.7146	10^{-3}	10^{-3}
0.9	0.7833	0.7837	0.7815	10^{-4}	10^{-3}
1	0.8414	0.8414	0.8414	10^{-4}	10^{-4}

with conditions $y(0)=0$, $y(0.5)=\sin(1/2)$, $y(1)=\sin(1)$. The exact solution is $y(x)=\sin(x)$. We set the parameters $\mu=0.001$, $n_1=7$, $n_2=10$, then train the neural network 1000 times, the weights for the network are given in Table 4. Fig. 9 and Fig. 10 show that both sample points and check points are in well agree with the exact solution. It also be seen from Table 5 that the numerical solutions we got have a very high accuracy. In the process of compiling the program, we calculate $\mathrm{Cos}_{\alpha,\beta}(x)$ and $\mathrm{Sin}_{\alpha,\beta}(x)$ by the formulas 2.1 in Definition 2.4. The Matlab code for Mittag-Leffler function was provided by Prof. Dr. I. Podlubny [19].

5 Conclusions

In this paper, we obtain the numerical solutions for the fractional Riccati equations and fractional Langevin equation by using the method of Cosine RBF neural networks, numerical experiments indicate that the Cosine RBFs are efficient in solving the numerical

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

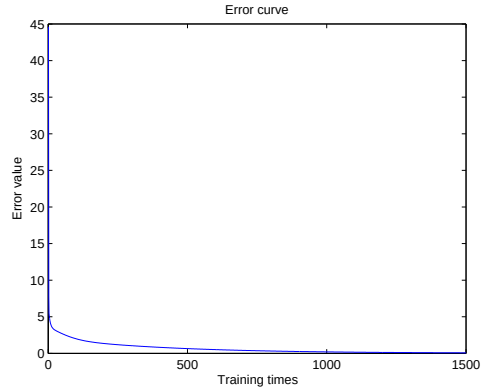


Figure 5: The error curve for Example 3.2.

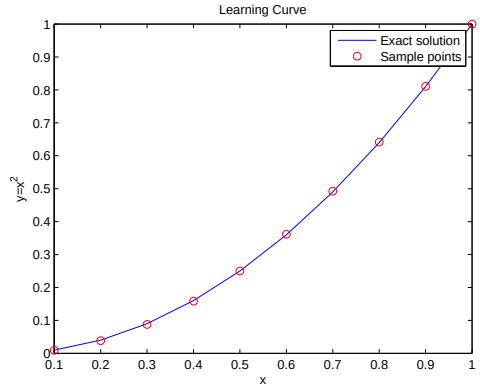


Figure 6: The learning curve for Example 4.1.

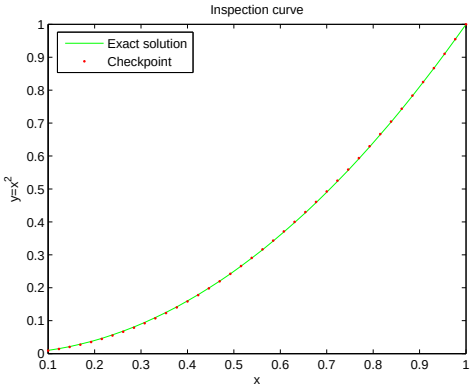


Figure 7: The inspection curve for Example 4.1.

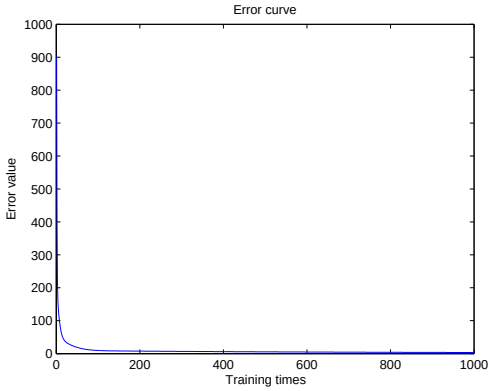


Figure 8: The error curve for Example 4.1.

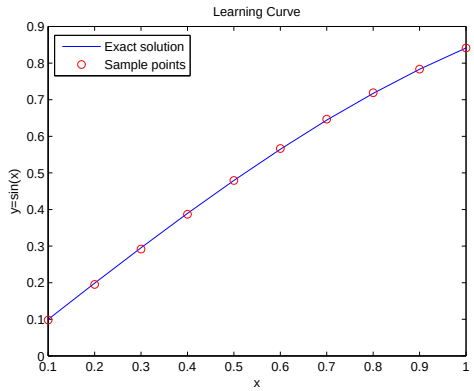


Figure 9: The learning curve for Example 4.2.

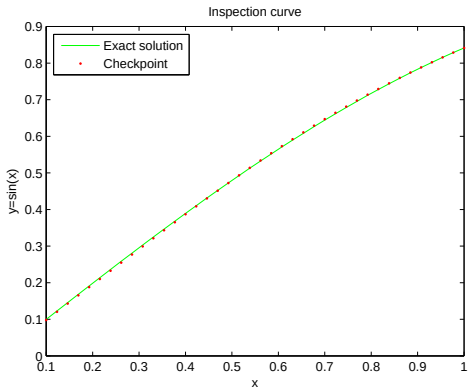


Figure 10: The inspection curve for Example 4.2.

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

solutions for fractional differential equations. Different from other methods (HAM, HPM and Decomposition method), we can easily get a high accuracy by increasing the number of training times. However, this will lead to longer running time of the program. Cosine RBF neural network is a powerful tool, and we are trying to solve partial differential equations of fractional.

Acknowledgements

The authors would like to thank the referees for their many constructive comments and suggestions to improve the paper. This work was partly supported by the Special Funds of the National Natural Science Foundations of China under Grant No. 11247310, the National Natural Science Foundations of China under Grant No. 11505042, the Foundations for Distinguished Young Talents in Higher Education of Guangdong under Grant No. 2012LYM0096, and the Fund of Hanshan Normal University under Grant No. LF201403.

References

[1] M. MARY, RANDOLPH-GIPS, NICOLAOS B. KARAYIANNIS, *Cosine radial basis function neural networks,* Proceedings of the International Joint Conference on Neural Networks, 1 (2003), pp. 96–101.
[2] ANDERSON MITTER AND W. T. REID, Riccati Differential Equations, Mathematics in Science and Engineering, Vol. 86. Academic Press, New York, 1972.
[3] B. D. ANDERSON AND J. B. MOORE, Optimal Control-linear Quadratic Methods, Prentice-Hall, New Jersey, 1999.
[4] S. K. MITTER, *Filtering and stochastic control: a historical perspective*, IEEE Control Syst. Mag. 16(3) (1996), pp. 67–76.
[5] K. DIETHELM, J. M. FORD, N. J. FORD AND W. WEILBEER, *Pitfalls in fast numerical solvers for fractional differential equations*, J. Comput. Appl. Math., 186 (2006), pp. 482–503.
[6] J. CANG, Y. TAN, H. XU AND S. LIAO, *Series solutions of non-linear Riccati differential equations with fractional order*, Chaos, Solitons and Fractals, 40 (2009), pp. 1–9.
[7] ZAID ODIBAT AND S. MOMANI, *Modified homotopy perturbation method: application to quadratic Riccati differential equation of fractional order*, Chaos, Solitons and Fractals, 36 (2008), pp. 167–174.
[8] S. MOMANI AND N. SHAWAGFEH, *Decomposition method for solving fractional Riccati differential equations*, Appl. Math. Comput., 182 (2006), pp. 1083–1092.
[9] R. A. KOSINSKI AND A. GRABOWSKI, *Theory and Langevin equations for modeling evacuation processes,* Acta Phys. Pol., 2 (2006), pp. 365–377.
[10] K. WODKIEWICZ AND M. S. ZUBAIRY, *Exact solution of a nonlinear Langevin equation with applications to photoelectron counting and noise-induced instability,* J. Math. Phys., 6 (1983), pp. 1401–1404.
[11] J. P. BOUCHAUD AND R. CONT, *A Langevin approach to stock market fluctuations and crashes,* Euro. Phys. J., 4 (1998), pp. 543–550.

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press

[12] P. GUO, C. ZENG, C. P. LI AND Y. Q. CHEN, *Numerics for the fractional langevin equation driven by the fractional brownian motion,* Fract. Calc. Appl. Anal., 16 (2013), pp. 123–141.
[13] V. KOBELEV AND E. ROMANOV, *Fractional Langevin equation to describe anomalous diffusion*, Prog. Theory Phys., 139 (2000), pp. 470–479.
[14] W. T. COFFEY, Y. P. KALMYKOV AND J. T. WALDRON, The Langevin Equation: with Applications to Stochastic Problems in Physics, Chemistry and Electrical Engineering. World Scientific Press, Singapore, 2004.
[15] C. H. EAB AND S. C. LIM, *Fractional generalized Langevin equation approach to Single-File diffusion*, Phys. A, 389(13) (2010), pp. 2510–2521.
[16] S. C. LIM, M. LI AND L. P. TEO, *Langevin equation with two fractional orders,* Phys. Lett. A, 372(42) (2008), pp. 6309–6320.
[17] B. N. AHMAD, J. J. ALSAEDI AND A. M. EL-SHAHED, *A study of nonlinear Langevin equation involving two fractional orders in different intervals*, Nonlinear Anal. Real World Appl., 13 (2012), pp. 599–606.
[18] A. A. KILBSA, H. M. SRIVASTAVA AND J. J. TRUJILLO, Theory and Applications of Fractional Differential Equations, Elsevier, Amsterdam, 2006.
[19] I. PODLUBNY, *Mittag-Leffler function,* File ID 8738, Matlab Central, `http://www.mathworks.com/matlabcentral/fileexchange/8738-mittag-leffler-function`, 2009.

https://doi.org/10.4208/aamm.2015.m909 Published online by Cambridge University Press