A Numerical Method for Solving Fractional Differential Equations by Using Neural Network

Haidong Qu and Xuan Liu

1Department of Mathematics, Hanshan Normal University, Chaozhou 521041, China
2Department of Basic Education, Hanshan Normal University, Chaozhou 521041, China

Correspondences should be addressed to Haidong Qu; qhaidong@163.com and Xuan Liu; czliuxuan@126.com

Received 5 February 2015; Revised 27 April 2015; Accepted 27 April 2015

Academic Editor: Fawang Liu

Copyright © 2015 H. Qu and X. Liu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We present a new method for solving the fractional differential equations of initial value problems by using neural networks which are constructed from cosine basis functions with adjustable parameters. By training the neural networks repeatedly the numerical solutions for the fractional differential equations were obtained. Moreover, the technique is still applicable for the coupled differential equations of fractional order. The computer graphics and numerical solutions show that the proposed method is very effective.

1. Introduction

Recently, fractional differential equations have gained considerable importance due to their frequent appearance applications in fluid flow, rheology, dynamical processes in self-similar and porous structures, diffusive transport akin to diffusion, electrical networks, probability and statistics, control theory of dynamical systems, viscoelasticity, electrochemistry of corrosion, chemical physics, optics and signal processing [1–7], and so on. These applications in interdisciplinary sciences motivate us to try to find out the analytic or numerical solutions for the fractional differential equations. But for most ones it is difficult to find out or even have exact solutions. Thus, necessarily, the numerical techniques are applied to the fractional differential equations.

Now, many effective methods for solving fractional differential equations have been presented, such as nonlinear functional analysis method including monotone iterative technique [8, 9], topological degree theory [10], and fixed point theorems [11–13]. Also, numerical solutions are obtained by the following methods: random walk [2], matrix approach [14], the Adomian decomposition method and variational iteration method [15], HAM [16–19], homotopy perturbation method (HPM) [20], and so forth. Not long ago, in [21], Raja et al. by applying Particle Swarm Optimization (PSO) algorithm along with feedforward ANN obtained the numerical solutions for fractional differential equations. But the convergence of the algorithm has not been proven, and this method is only applied to the single fractional differential equations. In this paper, we construct two different neural networks based on cosine functions and obtain the conditions of algorithm convergence.

The first neural network (NU) is applied to linear and nonlinear fractional differential equations of the form

\begin{equation}
D_0^\alpha y(x) = f(x, y(x)), \quad 0 < x \leq 1, \quad 0 < \alpha \leq 1
\end{equation}

with initial condition as follows:

\begin{equation}
y(0) = C,
\end{equation}

where \(D_0^\alpha\) is the Caputo fractional derivatives of order \(\alpha\).

The second neural network (NU) is applied to the fractional coupled differential equations of the form

\begin{equation}
D_0^\alpha y(x) = f(x, y(x), z(x)), \quad 0 < x \leq 1, \quad 0 < \alpha \leq 1, \quad (3)
\end{equation}

\begin{equation}
D_0^\alpha z(x) = g(x, y(x), z(x)),
\end{equation}

with initial conditions as follows:

\begin{equation}
y(0) = C_1, \quad z(0) = C_2.
\end{equation}
with initial conditions as follows:
\[ y(0) = C_1, \quad z(0) = C_2, \]  
(4)
where \( D^\alpha_0 \) is the Caputo fractional derivatives of order \( \alpha \). The solutions for the above two problems are written as cosine basis functions, whose parameters can be adjusted to minimize an appropriate error function. So we need to compute the gradient of the error with respect to the network parameters. By adjusting the parameters repeatedly, we obtain the numerical solutions when the error values are less than the required accuracy or the training times reach maximum.

2. Definitions and Lemma

Definition 1 (see [22]). The Riemann-Liouville fractional integral of order \( \alpha \in \mathbb{R}, \alpha > 0 \) of a function \( f(x) \in C, \mu \geq -1 \) is defined as
\[
\left( I^\alpha_{0+} f(t) \right)(x) := \frac{1}{\Gamma(\alpha)} \int_0^x \frac{f(t) dt}{(x-t)^{1-\alpha}}, \quad (x > 0). 
\]  
(5)

Definition 2 (see [22]). The Riemann-Liouville and Caputo fractional derivatives of order \( \alpha \in \mathbb{R}, \alpha > 0 \) are given by
\[
RLD^\alpha_0 f(x) := (d/dx)^n I^{n-\alpha}_{0+} f(x) = \frac{1}{\Gamma(n-\alpha)} \int_0^x (d/d\zeta)^n \frac{f(t) dt}{(x-t)^{\alpha-n+1}}, 
\]  
\[
D^\alpha_0 f(x) := I^{n-\alpha}_{0+} (d/dx)^n f(x) = \frac{1}{\Gamma(n-\alpha)} \int_0^x \frac{(d/dt)^n f(t) dt}{(x-t)^{\alpha-n+1}}, 
\]  
(6)
where \( (n = [\alpha] + 1, x > 0) \).

Definition 3 (see [22]). The classical Mittag-Leffler function is defined by
\[
E_\alpha(x) := \sum_{k=0}^{\infty} \frac{x^k}{\Gamma(ak+1)}, \quad (x \in \mathbb{C}, \alpha > 0). 
\]  
(7)
The generalized Mittag-Leffler function is defined by
\[
E_{\alpha,\beta}(x) := \sum_{k=0}^{\infty} \frac{x^k}{\Gamma(ak+\beta)}, \quad (x, \beta \in \mathbb{C}, \alpha > 0). 
\]  
(8)

Definition 4. The functions Sin_{\alpha,\beta}(x), Cos_{\alpha,\beta}(x) \( (x, \beta \in \mathbb{C}, \alpha > 0) \) are defined by
\[
Sin_{\alpha,\beta}(x) = \sum_{k=1}^{\infty} \frac{(-1)^k x^{2k-1}}{\Gamma(\alpha (2k-1) + \beta)}, 
\]  
\[
Cos_{\alpha,\beta}(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{\Gamma(\alpha (2k) + \beta)}. 
\]  
(9)

Lemma 5. If Sin_{\alpha,\beta}(x) and Cos_{\alpha,\beta}(x) are defined as in Definition 4, then
\[
D^\alpha_{a+} (x-a)^{\beta-1} Sin_{\mu,\beta} [\lambda (x-a)^\mu] = \frac{1}{\Gamma(n-\alpha)} \int_{a+}^x \frac{(d/dt)^n (t-a)^{\beta-1} Sin_{\mu,\beta} [\lambda (t-a)^\mu]}{(x-t)^{\alpha-n+1}} dt 
\]  
\[
= \frac{1}{\Gamma(n-\alpha)} \int_{0+}^x (\frac{d}{d\xi})^n (x-\xi)^{\beta-1} Sin_{\mu,\beta} [\lambda \xi^\mu] \frac{d\xi}{(x-\xi-a)^{\alpha-n+1}} \quad (\xi = t-a) 
\]  
\[
= \frac{1}{\Gamma(n-\alpha)} \int_{0+}^{x-a} (\frac{d}{dt})^n (x-a)^{\beta-1} Sin_{\mu,\beta} [\lambda t^\mu (x-a)^\mu] \frac{dt}{(x-a)^{\alpha-n+1}(1-t)^{\alpha-n+1}} \quad (\xi = t(x-a)) 
\]  
(10)
\[
\frac{1}{\Gamma(n-\alpha)} \int_0^1 (d/dt)^n t^{\beta-1+\mu(2k-1)-n} dt
\]

Suppose \(|f_1| \leq L_1, |\cos 1,1-\alpha(x)| \leq L_2\) on the interval \((\delta,1)\) for \(0 < \delta < 1\). (From Figure 10, we see that the function)

\section*{3. Illustration of the Method and Application}

\subsection*{3.1. The First Neural Network}

To describe the method, we consider (I) with initial condition \(y(0) = C\). The jth trial solution satisfying the initial condition is written as

\[
y_j(x) = \sum_{i=1}^M w_{i,j} \cos((x_1 + \) \ldots + \)

where \(M\) represents the number of neurons and \(w_{i,j}\) are unknown weights of the network determined in training procedures to reduce the error function:

\[
J = \frac{1}{2} \|E_j\|^2 = \frac{1}{2} \sum_{k=1}^M \left( e_j(k) \right)^2,
\]

\[
E_j = \left( e_j(1), e_j(2), \ldots, e_j(N) \right)^T,
\]

where \(N\) represents the number of sample points, \(\| \cdot \|_2\) is Euclidean norm, and

\[
e_j(k) = f \left( x_k, y_j(x_k) \right) - D_0^\alpha y_j^\alpha x_k
\]

\[
+ \left( C - \sum_{i=1}^M w_{i,j} \cos((x_1 + \) \ldots + \)
\]

where \(k = 1, 2, \ldots, N\); then we can adjust the weights \(w_{i,j}\) by the following equation:

\[
w_{i,j+1} = w_{i,j} + \Delta w_{i,j},
\]

\[\Delta w_{i,j} = -\frac{\partial J}{\partial w_{i,j}} = -\sum_{k=1}^N \frac{\partial J}{\partial e_j(k)} \frac{\partial e_j(k)}{\partial w_{i,j}}
\]

\[= -\sum_{k=1}^N e_j(k) f_y \left( x_k, y_j(x_k) \right)
\]

\[
\cdot \left( \cos(x_k) - \cos((x_1 + \) \ldots + \)
\]

\[\cdot (\cos_1,1-\alpha x_k - \cos_1,1-\alpha((x_1 + \) \ldots + \)
\]

\[\cdot (\cos_1,1-\alpha x_k - \cos_1,1-\alpha((x_1 + \) \ldots + \)
\]

3.2. Convergence of the Algorithm

\textbf{Theorem A.} Let \(\mu\) represent learning rate, let \(N\) represent the number of sample points, and let \(M\) represent the number of neurons: \(X = (x_1, x_2, \ldots, x_M), \delta < x_i < 1, 1 \leq i \leq N\). Suppose \(|f_j| \leq L_1, |\cos_1,1-\alpha(x)| \leq L_2\) on the interval \((\delta,1)\) for \(0 < \delta < 1\). (From Figure 10, we see that the function
Cos$_{1,1-\alpha}(x)$ is bounded when $0 \leq \alpha \leq 1$.) Then the neural network is convergent on the interval $(\delta, 1)$ when

$$0 < \mu < \frac{\mathcal{N} + 1}{4\mathcal{M}N(L_1 + \delta^{-\alpha}L_2)^2}. \quad (21)$$

Proof. Let $W_j = (w_{1j}, w_{2j}, \ldots, w_{mj})$, and then we denote $y_j(X)$ by

$$y_j(X) = W_j \cdot G - (C - W_j \cdot I_1) \cos((\mathcal{M} + 1) X), \quad (22)$$

where $G = (\cos(X), \cos(2X), \ldots, \cos(\mathcal{M}X))^T$ and $I_1 = (1, 1, \ldots, 1)^T$. Then according to (17), we have

$$E_j = f_j(X, y_j(X)) = f_j(X, y_j(X)) - D_0^\alpha y_j(X) \quad (23)$$

$$= f_j(X, y_j(X)) - X^\alpha (W_jH + (C - W_jI_1) \cos((\mathcal{M} + 1) X)), \quad (24)$$

where $H = (\cos_{1,1-\alpha}(X), \cos_{1,1-\alpha}(2X), \ldots, \cos_{1,1-\alpha}(\mathcal{M}X))^T$. Then we have

$$\frac{\partial E_j}{\partial W_j} = f_j(G - I_1 \cos((\mathcal{M} + 1) X))$$

Thus,

$$E_j = f_j(X, y_j(X)) - X^\alpha (H - I_1 \cos_{1,1-\alpha}((\mathcal{M} + 1) X)). \quad (25)$$

Define Lyapunov function $V_j = (1/2)\|E_j\|^2$; we have

$$\Delta V_j = \frac{1}{2} \left\|E_{j+1}\right\|^2 - \frac{1}{2} \left\|E_j\right\|^2. \quad (26)$$

Suppose

$$E_{j+1} = E_j + \Delta E_j = E_j + \left(\frac{\partial E_j}{\partial W_j}\right)^T \Delta W_j, \quad (27)$$

and then in accordance with (25) that yields

$$E_{j+1} = \left(I - \mu \left(\frac{\partial E_j}{\partial W_j}\right)^T \frac{\partial E_j}{\partial W_j}\right) E_j, \quad (28)$$

where

$$I = \begin{pmatrix} 1 & \cdots & 1 \\
&\ddots & \\
& & 1 \end{pmatrix}_{\mathcal{N} \times \mathcal{N}}. \quad (29)$$

By calculating (25), we get

$$\frac{\partial E_j}{\partial W_j} \right\|_F$$

and then in accordance with (25) that yields

$$E_{j+1} = \left(I - \mu \left(\frac{\partial E_j}{\partial W_j}\right)^T \frac{\partial E_j}{\partial W_j}\right) E_j, \quad (28)$$

where

$$I = \begin{pmatrix} 1 & \cdots & 1 \\
&\ddots & \\
& & 1 \end{pmatrix}_{\mathcal{N} \times \mathcal{N}}. \quad (29)$$

Thus,

$$\Delta V_j = \frac{1}{2} \left\|\left(I - \mu \left(\frac{\partial E_j}{\partial W_j}\right)^T \frac{\partial E_j}{\partial W_j}\right) E_j\right\|^2 - \frac{1}{2} \left\|E_j\right\|^2 \quad (30)$$

$$\leq \frac{1}{2} \left\|\left(I - \mu \left(\frac{\partial E_j}{\partial W_j}\right)^T \frac{\partial E_j}{\partial W_j}\right)\right\|_F^2 \leq 1 \left\|\left[I - \mu \left(\frac{\partial E_j}{\partial W_j}\right)^T \frac{\partial E_j}{\partial W_j}\right]\right\|_F^2, \quad (33)$$

where $\|\cdot\|_F$ is Frobenius matrix norm, defined by $\|(a_{ij})_{\mathcal{N} \times \mathcal{N}}\|_F = (\sum_{i=1}^{\mathcal{N}} \sum_{j=1}^{\mathcal{N}} a_{ij}^2)^{1/2}$. Since $\|E_j\|^2 > 0$, in order to make this neural network converge, we have

$$\left\|\left(I - \mu \left(\frac{\partial E_j}{\partial W_j}\right)^T \frac{\partial E_j}{\partial W_j}\right)\right\|_F < 1, \quad (31)$$

which yields

$$\left\|I - \mu \left(\frac{\partial E_j}{\partial W_j}\right)^T \frac{\partial E_j}{\partial W_j}\right\|_F < 1. \quad (32)$$

Hence,

$$0 < \mu < \frac{\mathcal{N} + 1}{4\mathcal{M}N(L_1 + \delta^{-\alpha}L_2)^2}. \quad (34)$$

In accordance with $0 < \mu < 1$, we obtain

$$0 < \mu < \frac{\mathcal{N} + 1}{4\mathcal{M}N(L_1 + \delta^{-\alpha}L_2)^2}. \quad (36)$$

□
Table 1: Weights ($\times 10^{-4}$) obtained along with the solution of Examples 1, 2, and 3.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Example 1</th>
<th></th>
<th>Example 2</th>
<th></th>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>4857</td>
<td>6310</td>
<td>6665</td>
<td>8226</td>
<td>9941</td>
</tr>
<tr>
<td>$w_2$</td>
<td>-0506</td>
<td>-3466</td>
<td>-4424</td>
<td>4080</td>
<td>0219</td>
</tr>
<tr>
<td>$w_3$</td>
<td>-4434</td>
<td>-2721</td>
<td>-1362</td>
<td>-3468</td>
<td>-0621</td>
</tr>
<tr>
<td>$w_4$</td>
<td>-3170</td>
<td>-3110</td>
<td>-4990</td>
<td>1680</td>
<td>1372</td>
</tr>
<tr>
<td>$w_5$</td>
<td>1896</td>
<td>4204</td>
<td>5290</td>
<td>-1455</td>
<td>-1468</td>
</tr>
<tr>
<td>$w_6$</td>
<td>5534</td>
<td>-0926</td>
<td>0222</td>
<td>1895</td>
<td>0482</td>
</tr>
<tr>
<td>$w_7$</td>
<td>-6316</td>
<td>0182</td>
<td>-2151</td>
<td>-1350</td>
<td>0424</td>
</tr>
</tbody>
</table>

Learning curve

![Learning curve](image1)

Inspection curve

![Inspection curve](image2)

3.3. Example

3.3.1. Example 1. We first consider the following linear fractional differential equation:

$$D_{0+}^{\alpha} y(x) = x^2 + \frac{2}{\Gamma(3-\alpha)} x^{2-\alpha} - y(x),$$

with condition $y(0) = 0$. The exact solution is $y(x) = x^2$. This equation also can be solved by the following methods: Genetic Algorithm (GA) [21], Grünwald-Letnikov classical numerical technique (GL) [23], and Particle Swarm Optimization (PSO) algorithm [23]. We set the parameters $\mu = 0.001$, $M = 7$, and $N = 10$ and train the neural network 4500 times, and the weights of the network for Example 1 are given in Table 1. Figure 1 shows that sample points are on the exact solution curve after training is completed. Then we check whether the other points also match well with the exact solution (see Figure 2). From Figure 3 we see the error values decrease rapidly. Tables 2(a) and 2(b) show the numerical solutions and accuracy for Example 1 by the different methods. In this paper, all numerical experiments are done by using Lenovo T400, Intel Core 2 Duo CPU P8700, 2.53 GHz, and Matlab version R2010b. The neural networks with cosine basis functions have taken about 850 s, but the other algorithms mentioned above need to run about 2,240 s.

3.3.2. Example 2. We secondly consider the following linear fractional differential equation:

$$D_{0+}^{\alpha} y(x) = \cos(x) + x^{-\alpha} \cos_{1,1-\alpha}(x) - y(x),$$

with condition $y(0) = 1$. The exact solution is $y(x) = \cos(x)$. We set the parameters $\mu = 0.001$, $M = 7$, and $N = 10$ and train the neural network 1000 times, and the weights of the network for Example 2 are given in Table 1. Figures 4, 5, and 6 show that the neural network is still applicable when $C \neq 0$. Table 3 shows the exact solution, approximate solution, and accuracy for Example 2.
3.3.3. Example 3. We thirdly consider the following nonlinear fractional differential equation:

\[ D_{0+}^{\alpha} y(x) = x^6 + \frac{\Gamma (3.5)}{\Gamma (3.5 - \alpha)} x^{2.5 - \alpha} - xy^2(x), \quad (39) \]

with condition \( y(0) = 0 \). The exact solution is \( y(x) = x^{5/2} \). We set the parameters \( \mu = 0.001, \mathcal{M} = 7, \) and \( \mathcal{N} = 10 \) and train the neural network 1000 times, and the weights of the network for Example 2 are given in Table 1. Table 4 shows the exact solution, approximate solution, and accuracy for Example 3.

\[ y_j(x) = \sum_{i=1}^{\mathcal{M}} w_{i,j} \cos(ix) + \left( C_1 - \sum_{i=1}^{\mathcal{M}} w_{i,j} \right) \cos((\mathcal{M} + 1)x), \quad (40) \]

\[ z_j(x) = \sum_{i=1}^{\mathcal{M}} q_{i,j} \cos(ix) + \left( C_2 - \sum_{i=1}^{\mathcal{M}} q_{i,j} \right) \cos((\mathcal{M} + 1)x), \]
Table 2: (a) Comparison of results for the solution of Example 1 for $\alpha = 0.5$. (b) Comparison of results for the solution of Example 1 for $\alpha = 0.75$.

(a)

<table>
<thead>
<tr>
<th></th>
<th>$y(x)$</th>
<th>Numerical solution</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GL</td>
<td>PSO</td>
<td>GA</td>
</tr>
<tr>
<td>0.1</td>
<td>0.01</td>
<td>0.0107</td>
<td>0.0103</td>
</tr>
<tr>
<td>0.2</td>
<td>0.04</td>
<td>0.0401</td>
<td>0.0404</td>
</tr>
<tr>
<td>0.3</td>
<td>0.09</td>
<td>0.0907</td>
<td>0.0907</td>
</tr>
<tr>
<td>0.4</td>
<td>0.16</td>
<td>0.1604</td>
<td>0.1596</td>
</tr>
<tr>
<td>0.5</td>
<td>0.25</td>
<td>0.2496</td>
<td>0.2496</td>
</tr>
<tr>
<td>0.6</td>
<td>0.36</td>
<td>0.3583</td>
<td>0.3573</td>
</tr>
<tr>
<td>0.7</td>
<td>0.49</td>
<td>0.4869</td>
<td>0.4869</td>
</tr>
<tr>
<td>0.8</td>
<td>0.64</td>
<td>0.6362</td>
<td>0.6352</td>
</tr>
<tr>
<td>0.9</td>
<td>0.81</td>
<td>0.8069</td>
<td>0.8069</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.0001</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

(b)

<table>
<thead>
<tr>
<th></th>
<th>$y(x)$</th>
<th>Numerical solution</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GL</td>
<td>PSO</td>
<td>GA</td>
</tr>
<tr>
<td>0.1</td>
<td>0.01</td>
<td>0.0107</td>
<td>0.0103</td>
</tr>
<tr>
<td>0.2</td>
<td>0.04</td>
<td>0.0401</td>
<td>0.0404</td>
</tr>
<tr>
<td>0.3</td>
<td>0.09</td>
<td>0.0907</td>
<td>0.0907</td>
</tr>
<tr>
<td>0.4</td>
<td>0.16</td>
<td>0.1604</td>
<td>0.1596</td>
</tr>
<tr>
<td>0.5</td>
<td>0.25</td>
<td>0.2496</td>
<td>0.2496</td>
</tr>
<tr>
<td>0.6</td>
<td>0.36</td>
<td>0.3583</td>
<td>0.3573</td>
</tr>
<tr>
<td>0.7</td>
<td>0.49</td>
<td>0.4869</td>
<td>0.4869</td>
</tr>
<tr>
<td>0.8</td>
<td>0.64</td>
<td>0.6362</td>
<td>0.6352</td>
</tr>
<tr>
<td>0.9</td>
<td>0.81</td>
<td>0.8069</td>
<td>0.8069</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.0001</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Table 3: Exact solution, approximate solution, and accuracy for Example 2.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Numerical solution</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\cos(x)$</td>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9950</td>
<td>0.9945</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9800</td>
<td>0.9878</td>
</tr>
<tr>
<td>0.3</td>
<td>0.9553</td>
<td>0.9538</td>
</tr>
<tr>
<td>0.4</td>
<td>0.9210</td>
<td>0.9203</td>
</tr>
<tr>
<td>0.5</td>
<td>0.8775</td>
<td>0.8777</td>
</tr>
<tr>
<td>0.6</td>
<td>0.8253</td>
<td>0.8254</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7648</td>
<td>0.7639</td>
</tr>
<tr>
<td>0.8</td>
<td>0.6967</td>
<td>0.6951</td>
</tr>
<tr>
<td>0.9</td>
<td>0.6216</td>
<td>0.6213</td>
</tr>
<tr>
<td>1</td>
<td>0.5403</td>
<td>0.5414</td>
</tr>
</tbody>
</table>

Table 4: Exact solution, approximate solution, and accuracy for Example 3.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Numerical solution</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x^{3/2}$</td>
<td>1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.0031</td>
<td>0.0022</td>
</tr>
<tr>
<td>0.2</td>
<td>0.0178</td>
<td>0.0133</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0492</td>
<td>0.0426</td>
</tr>
<tr>
<td>0.4</td>
<td>0.1011</td>
<td>0.0972</td>
</tr>
<tr>
<td>0.5</td>
<td>0.1767</td>
<td>0.1773</td>
</tr>
<tr>
<td>0.6</td>
<td>0.2788</td>
<td>0.2797</td>
</tr>
<tr>
<td>0.7</td>
<td>0.4099</td>
<td>0.4055</td>
</tr>
<tr>
<td>0.8</td>
<td>0.5724</td>
<td>0.5643</td>
</tr>
<tr>
<td>0.9</td>
<td>0.7684</td>
<td>0.7670</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.0064</td>
</tr>
</tbody>
</table>

where $N$ represents the number of sample points and $w_{ij}$ and $q_{ij}$ are unknown weights of the network determined in training procedures to reduce error function:

$$J^{xyz} = \frac{1}{2} \left\| E_j^{xyz} \right\|_2^2 = \frac{1}{2} \sum_{k=1}^{x} (e_j^x(k))^2 + \frac{1}{2} \sum_{k=1}^{y} (e_j^y(k))^2 + \frac{1}{2} \sum_{k=1}^{z} (e_j^z(k))^2,$$  \hspace{1em} (41)

where

$$E_j^{xyz} = (e_j^z(1), e_j^z(2), \ldots, e_j^z(N), e_j^y(1), e_j^y(2), \ldots, e_j^x(1), e_j^x(2), \ldots),$$

$$e_j^x(k) = f(x_k, y_j(x_k), z_j(x_k)) - D_{x,y}(x_k).$$
\[
\begin{align*}
= f(x_k, y_j(x_k), z_j(x_k)) \\
- x^\alpha \left( \sum_{i=1}^d w_{i,j} \cos_{1,\alpha} (ix_k) + (C_1 - \sum_{i=1}^d w_{i,j}) \right) \\
\cdot \cos_{1,\alpha} ((\mathcal{M} + 1)x_k),
\end{align*}
\]

e^j(k) = g(x_k, y_j(x_k), z_j(x_k)) - D_0^\alpha z_j(x_k)

e^j(k) = g(x_k, y_j(x_k), z_j(x_k)) - D_0^\alpha z_j(x_k)

\begin{align*}
\Delta w_{i,j} &= -\mu \frac{\partial E_j}{\partial w_{i,j}} \\
&= -\mu \left( \sum_{k=1}^N \frac{\partial E_j}{\partial e^x_j(k)} \frac{\partial e^x_j(k)}{\partial w_{i,j}} \right) \\
&\cdot \left( \cos (x_k) - \cos ((\mathcal{M} + 1)x_k) - (x_k)^{(-\alpha)} \right) \\
&\cdot \left( \cos_{1,\alpha} x_k - \cos_{1,\alpha} ((\mathcal{M} + 1)x_k) \right) \\
&\cdot \left( \cos (x_k) - \cos ((\mathcal{M} + 1)x_k) \right),
\end{align*}

\begin{align*}
\Delta q_{i,j} &= -\mu \frac{\partial E_j}{\partial q_{i,j}} \\
&= -\mu \left( \sum_{k=1}^N \frac{\partial E_j}{\partial e^y_j(k)} \frac{\partial e^y_j(k)}{\partial q_{i,j}} \right) \\
&\cdot \left( \cos (x_k) - \cos ((\mathcal{M} + 1)x_k) \right) \\
&\cdot \left( \cos_{1,\alpha} x_k - \cos_{1,\alpha} ((\mathcal{M} + 1)x_k) \right),
\end{align*}

3.5. Convergence of the Algorithm

**Theorem B.** Let \( \mu \) represent learning rate, let \( \mathcal{N} \) represent the number of sample points, and let \( \mathcal{M} \) represent the number of neurons: \( X = (x_1, x_2, \ldots, x_d), \delta < x_i < 1, 1 \leq i \leq \mathcal{N}. \) Suppose \( |f_j| \leq L_1^f, |f_j| \leq L_1^g, |g_j| \leq L_1^y, \delta_1 \leq L_2^\delta, |\cos_{1,\alpha}(x)| \leq L_2 \) on the interval \((\delta, 1)\) for \( 0 < \delta < 1.\) Then the neural network is convergent on the interval \((\delta, 1)\) when

\[
\begin{align*}
0 &< \mu \\
&< \frac{2\mathcal{N} + 1}{4\mathcal{M}\mathcal{N} \left( (L_1^f + L_1^g + \delta^{-\alpha}L_2) \right)^2 + (L_1^f + L_1^g + \delta^{-\alpha}L_2)^2}. 
\end{align*}
\]

**Proof.** Let \( W_j^{ryz} = (w_{1,j}, w_{2,j}, \ldots, w_{\mathcal{M},j}, q_{1,j}, q_{2,j}, \ldots, q_{\mathcal{M},j}), \) and then we denote \( y_j(X) \) and \( z_j(X) \) by

\[
\begin{align*}
y_j(X) &= W_j^{ryz} g_1 \cos ((\mathcal{M} + 1)x_k), \\
z_j(X) &= W_j^{ryz} g_1 + (C_2 - W_j^{ryz} 0,1) \cos ((\mathcal{M} + 1)x_k),
\end{align*}
\]

respectively, where

\[
\begin{align*}
E_j^{ryz} = (e^y_j(1), e^y_j(2), \ldots, e^y_j(\mathcal{N}), e^x_j(1), e^x_j(2), \ldots), \\
e^y_j(\mathcal{N}) = (f(X, y_j(X), z_j(X)) - D_0^\alpha y_j(X), \\
g(X, y_j(X), z_j(X)) - D_0^\alpha z_j(X) = f(X), \\
y_j(X), z_j(X) - X^\alpha (W_j^{ryz} H^0) \\
+ (C_1 - W_j^{ryz} 1,0) \cos_{1,\alpha} ((\mathcal{M} + 1)x_k) + g(X, \\
y_j(X), z_j(X) - X^\alpha (W_j^{ryz} H^0) \\
+ (C_2 - W_j^{ryz} 0,1) \cos_{1,\alpha} ((\mathcal{M} + 1)x_k),
\end{align*}
\]
\[
H_{1,0} = \left( \cos_{1,1-\alpha}(X), \cos_{1,1-\alpha}(2X), \ldots \right) ^T,
\]
\[
H_{0,1} = \left( \cos_{1,1-\alpha}(X), \cos_{1,1-\alpha}(2X), \ldots \right) ^T,
\]
\[
\mathcal{G}_{1,0} = \left( \cos(X), \cos(2X), \ldots \right) ^T,
\]
\[
\mathcal{G}_{0,1} = \left( \cos(X), \cos(2X), \ldots \right) ^T,
\]
\[
i_{1,0} = \left( \frac{\mathcal{M}}{1,1,\ldots,1,0,0,\ldots,0} \right) ^T,
\]
\[
i_{0,1} = \left( \frac{\mathcal{M}}{0,0,\ldots,0,1,1,\ldots,1} \right) ^T.
\]

Then we have
\[
\frac{\partial E_{y,z}^{y,z}}{\partial W_{y,z}^{y,z}} = f_y \left( \mathcal{G}_{1,0} - i_{1,0} \cos ((\mathcal{M} + 1)X) \right) + f_z \left( \mathcal{G}_{0,1} - i_{0,1} \cos ((\mathcal{M} + 1)X) \right) - X^\alpha \left( H_{1,0} - i_{1,0} \cos_{1,1-\alpha} ((\mathcal{M} + 1)X) \right) + g_z \left( \mathcal{G}_{0,1} - i_{0,1} \cos ((\mathcal{M} + 1)X) \right). \tag{48}
\]

Define Lyapunov function \( V_{y,z}^{y,z} = \left( 1/2 \right) \| E_{y,z}^{y,z} \|_2^2 \); then similarly to the proof of Theorem A we get
\[
0 < \mu < \frac{2\mathcal{N} + 1}{\| \partial E_{y,z}^{y,z} / \partial W_{y,z}^{y,z} \|_F^2}. \tag{50}
\]
By simply calculating \( \partial E_{y,z}^{y,z} / \partial W_{y,z}^{y,z} \), we have
\[
\left\| \frac{\partial E_{y,z}^{y,z}}{\partial W_{y,z}^{y,z}} \right\|_F^2 \leq 2 \left( \begin{array}{c} \mathcal{A} \cdots \mathcal{A} \mathcal{B} \cdots \mathcal{B} \\ \vdots \vdots \vdots \vdots \vdots \vdots \end{array} \right) \left( \begin{array}{c} \mathcal{A} \cdots \mathcal{A} \mathcal{B} \cdots \mathcal{B} \\ \vdots \vdots \vdots \vdots \vdots \vdots \end{array} \right)_{\mathcal{M} \times 2\mathcal{N}}^F \leq 4\mathcal{M}\mathcal{N} \left( \mathcal{A}^2 + \mathcal{B}^2 \right), \tag{51}
\]
where \( \mathcal{A} = L_1^2 + L_2 + \delta^{-\alpha} L_2 \) and \( \mathcal{B} = L_1^2 + L_2 + \delta^{-\alpha} L_2 \), and finally we obtain
\[
0 < \mu < \frac{2\mathcal{N} + 1}{4\mathcal{M}\mathcal{N} \left( (L_1^2 + L_2^2 + \delta^{-\alpha} L_2^2)^2 + (L_1^2 + L_2^2 + \delta^{-\alpha} L_2^2)^2 \right)}. \tag{52}
\]
This completes the proof. \( \square \)

### 3.6. Example

#### 3.6.1. Example 4
We first consider the following linear coupled fractional differential equations:
\[
D_0^\alpha y(x) = x^2 + x^3 + \frac{2}{1 - \alpha} x^{2-\alpha} - y(x) - z(x),
\]
\[
0 < x < 1, \quad 0 < \alpha \leq 1. \tag{53}
\]
\[
D_0^\alpha z(x) = x^2 + x^3 + \frac{6}{1 - \alpha} x^{3-\alpha} - y(x) - z(x),
\]
\[
0 < x < 1, \quad 0 < \alpha \leq 1. \tag{54}
\]

with initial condition as follows:
\[
y(0) = 0,
\]
\[
z(0) = 0.
\]

The exact solution is \( y(x) = x^2 \) and \( z(x) = x^3 \). We set the parameters \( \alpha = 0.9, \mu = 0.001, \mathcal{M} = 7, \) and \( \mathcal{N} = 10 \) and train the neural network 2000 times, and the weights of the network for Example 4 are given in Table 5. Figures 7 and 8 show that the sample points and checkpoints are in well agreement with the exact solutions for the problem. Figure 9 shows that the error of the numerical solutions decreases rapidly within the first 50 training times. Table 6 shows the exact solution, approximate solution, and accuracy for Example 4.
Table 5: Weights obtained along with the solution of Examples 4 and 5.

<table>
<thead>
<tr>
<th>( x = 0.9 )</th>
<th>Example 4</th>
<th>Example 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{w_1}{q_1} )</td>
<td>0.5307/0.4494</td>
<td>0.8215/0.7682</td>
</tr>
<tr>
<td>( \frac{w_2}{q_2} )</td>
<td>-0.0592/0.0013</td>
<td>0.3545/-0.7084</td>
</tr>
<tr>
<td>( \frac{w_3}{q_3} )</td>
<td>-0.6414/-0.7451</td>
<td>-0.1512/-0.2292</td>
</tr>
<tr>
<td>( \frac{w_4}{q_4} )</td>
<td>0.0052/-0.0392</td>
<td>-0.1342/-0.1116</td>
</tr>
<tr>
<td>( \frac{w_5}{q_5} )</td>
<td>0.0559/0.3705</td>
<td>0.1657/0.3582</td>
</tr>
<tr>
<td>( \frac{w_6}{q_6} )</td>
<td>0.3998/0.2282</td>
<td>-0.0815/0.0781</td>
</tr>
<tr>
<td>( \frac{w_7}{q_7} )</td>
<td>-0.4141/-0.4014</td>
<td>0.0533/-0.2259</td>
</tr>
</tbody>
</table>

3.6.2. Example 5. We second consider the following nonlinear fractional coupled differential equations:

\[
D_0^\alpha y(x) = x^6 + \cos(x) + \frac{6}{\Gamma (4-x)}x^{3-x} - (y(x))^2 - z(x), \quad 0 < x \leq 1, \quad 0 < \alpha \leq 1, \tag{55}
\]

\[
D_0^\mu z(x) = x^3 + \cos(x) + x^{\alpha} \cos_{\alpha}(x) - y(x) - z(x),
\]

with initial conditions as follows:

\[
y(0) = 0, \\
z(0) = 1, \tag{56}
\]

The exact solution is \( y(x) = x^3 \) and \( z(x) = \cos(x) \). We set the parameters \( \alpha = 0.9, \mu = 0.001, M = 7, \) and \( N = 10 \). Numerical solutions in Table 7 show that this network can also be applied to the nonlinear fractional coupled differential equations but we need more time to train the network.

4. Conclusion

In this paper, by using the neural network, we obtained the numerical solutions for single fractional differential equations and the systems of coupled differential equations of fractional order. The computer graphics demonstrates that numerical results are in well agreement with the exact solutions. In (1), suppose that \( f(x, y(x)) = A(x) + B(x)y + C(x)y^2 \); then the problem transformed into Fractional Riccati Equations (Example 3 in this paper). In (3), suppose that \( f(x, y(x)) = y(x)(r - ay(x) - bz(x)) \) and \( g(x, z(x)) = z(x)(-d + cy(x)) \); then the problem transformed into...
fractional-order Lotka-Volterra predator-prey system. We will consider this problem in another paper. The neural network is a powerful method and is effective for the above two problems, which should be also able to solve fractional partial differential equations.

**Conflict of Interests**

The authors declare that there is no conflict of interests regarding the publication of this paper.

**Acknowledgments**

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 Foundations for Distinguished Young Talents in Higher Education of Guangdong under Grant no. 2012LYM0096, and the Fund of Hanshan Normal University under Grant nos. LY201302 and LF201403.

**References**


