Chapter 4

Study of fractional-order reaction-advection-diffusion equation using Neural Network Method

4.1 Introduction

The fractional order integrals and derivatives were developed practically at the same time when the integer calculus was developed. Fractional calculus has attracted a lot of attention recently due to its wide applications in a variety of fields including financial systems, physical, chemical, geological and biological systems [95], [123]. No approach provides an exact solution to the fractional-order differential equation (FDE). However, the nonlinear fractional order partial differential equations (FPDEs) have attracted special attentions of the scientific community [3], [75], [124],

The contents of this chapter have been published in **Mathematics and Computers in Simulation**, (7), **22**(2019)787-798.

[125], [126], [127], [128] and [129]. The time memory or historical inheritance of a fractional derivative is its most notable characteristic [130], because of which, fractional derivatives have extensive applications in diverse domains [131]. FPDE can be used to represent a variety of natural systems, including the thermal pollution of river systems, atmospheric pollution and groundwater pollution. The equation that describes the flows in porous media is solved to determine the velocities of the transport medium. Although the flow equations are nonlinear, diffusion and advection are the most crucial factors. The advection-diffusion equation describes how a solute is transported when advection and diffusion are acting together. Reaction-advection-diffusion equations, another type of chemical equation, are shown to exist when the chemical being carried through soil is reactive. In general, fractional differential models do not have any analytical solution, or if they do, the analytical solutions are challenging to calculate because of the involvement of many complicated functions in such models. Therefore, it is crucial to put more effort in research related to FDEs for their numerical solutions. Nonlinear FDEs are typically difficult to solve precisely, necessitating the use of numerical methods and methods of approximation. Researchers in this field have primarily suggested three numerical procedures to effectively solve the corresponding FDEs: the finite element method (FEM), the spectrum method (SM) and the finite difference method (FDM). In order to solve various types of linear and nonlinear ordinary FDEs, Raja et al. [132] have developed a feed-forward artificial neural network (ANN)-based method. He demonstrated the usefulness of this approach along with the limitation that this approach provides less accuracy while dealing with difficult nonlinear FDEs.

The ANN based approach for numerical solutions to FDEs was examined by [133] to show the effectiveness of these networks. The effectiveness of this technique was demonstrated by comparing with the analytical solution and a number of numerical techniques which are currently in use. Improvement of the performance in thermal

and environmental processes using ANN with conformable transfer function is done in [134]. An overview of real-world uses for ANN in fractional calculus can be found in [135]. The trial solutions of models are built from a combination of adjustable and non-adjustable elements. PDEs, linked with ODE systems, and individual ODEs can be solved using the said technique. In [136], the authors thought about using an iterative approach to solve FDEs and used the generalised sigmoid function as the cost function. Wei et al. [137] suggested that the time-fractional Fokker-Planck equation can be solved using a neural network. In [138], a complete description of radial basis function in neural network algorithms to solve various kinds of differential equations has been provided. The sequential quadratic programming (SQP) algorithm is used to effectively update the weights of the network via restricted optimization. In [139], the authors have created a new computing method employing fractional neural networks to solve fractional-order Bagley-Torvik equations with initial conditions. In recent years, it is seen a growth of crucial dynamical problems, dependent on time, space, or both, showing behaviour of the fractional-order. The fractional Adams-Bashforth approach was described in its right form in [140]. Functional link neural network (FLNN), a higher order neural network, was used to represent linear and nonlinear delay fractional optimal control problems (DFOCPs) with mixed control-state constraints in [141]. Using a novel method based on ANN, the approximate solutions of FDEs were investigated in [142].Neural network approach being a part of Artificial intelligence (AI) has become very popular due to its ability to do incredible tasks like finding the patterns in data which otherwise is very difficult. B. Shiri et al. [143] have proposed an adaptive gradient descent method based on NNM to minimize energy functions. For image processing, speech recognition, manufacturing virtual assistants like Alexa, training machines, this can perform more efficiently than a human being. ANN is also performing in those areas very efficiently, where human beings could not even think of it even one or two

decades ago.

In present paper, we are putting forth a numerical method to solve the nonlinear time-space fractional PDEs. This method, consisting of Legendre polynomials, is motivated by the literary work [144], which employed the NNM to solve an integer order differential equation. Here an effort has been given to employ the method to solve a nonlinear FPDE, which is first of its kind.

The benefit of the studied method is its primary concept which is based on training the networks by using a small number of sample points on the solution area. It is noticed that the accuracy improves with more training. However it was very challenging to deal with the accuracy of the method with larger number of training set. The NNM is regarded as superior to the FEM, FDM or any other numerical method for solving the nonlinear PDEs in integer as well as fractional-order systems. This is because NNM can improve the calculation accuracy by encrypting the grid or by adding interpolating nodes. In contrast to the grid-based method, which can only find approximate solutions for grid points, NNM can obtain the approximate solutions for all of the points in the interval from a small sample of points FRADEs are in general solved using numerical methods. The scientific community is trying to develop and employ new and advanced techniques to achieve the more accurate solutions of FRADEs and also to reduce the effort in obtaining these solutions. Hence, state of the art methods developed in recent years such as NNM need to be tested for such problems. This led us to attempt the present study.

Here an endeavour is made to apply the NNM based on Legendre polynomials to investigate the following nonlinear time-space fractional order reaction-advection-diffusion equation (FRADE) given by

$$\frac{\partial^{\alpha} u(x,t)}{\partial t^{\alpha}} = \frac{\partial^{\beta} u(x,t)}{\partial x^{\beta}} + \nu u^{\eta}(x,t)\frac{\partial u(x,t)}{\partial x} + \lambda R(u), 0 < \alpha < 1, 1 < \beta < 2, \quad (4.1)$$

under the prescribed initial condition and boundary conditions as

$$u(x,0) = \xi_1(x), 0 \leq x \leq X, \tag{4.2}$$

$$u(0,t) = \xi_2(t), 0 \leq t \leq T, \tag{4.3}$$

$$\frac{\partial u(1,t)}{\partial x} = \xi_3(t), 0 \leq t \leq T, \tag{4.4}$$

where $\eta \in N$, the set of all natural numbers and ν is the advection coefficient. The reaction term is considered here as $R(u) = u(x,t)(1-u^{\eta}(x,t))$ which fulfils the Lipschitz criterion:

$$|R(u_1) - R(u_2)| \leq L|u_1 - u_2|, \qquad \forall u_1, u_2.$$

In this chapter, it is shown that how NNM works to solve nonlinear FRADE. The q^{th} test result is denoted by [144]:

$$u^q(x,t) = \sum_{i=o}^{M_x} \sum_{j=o}^{M_t} u_{i,j}^q L_{X,i}(x) L_{T,j}(t), \tag{4.5}$$

where $L_{X,i}(x)$ and $L_{T,j}(t)$ are the shifted Legendre polynomials stated in equation (1.43). So, the function $u^q(x,t)$ defined in (4.5) is a continuous function on $[0,1] \times [0,1]$. The network is depicted in the Figure 4.1. The neural networks change the weights to reduce the loss function using the unknown weights $u_{i,j}^q$, $i \in \{0,1,...,M_x\}$ and $j \in \{0,1,...,M_t\}$ which are first allocated at random. The model (4.1) will be approximated by inserting the test solution $u^q(x,t)$ with unknown weights $u_{i,j}^q$ and iteratively training to change the unknown weights $u_{i,j}^q$.

The basis functions $L_{T,j}(t)$, and $L_{X,i}(x)$ have their time and spatial derivatives determined from the model (4.1). To begin, let us compute the time derivative. For

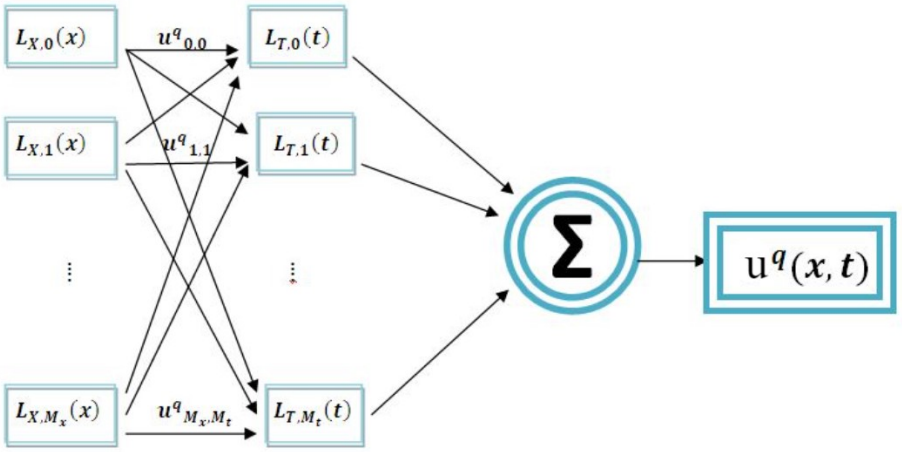

Figure 4.1: Neural network structure.

convenience, let us consider

$$T(t) = [L_{T,0}(t), L_{T,1}(t), ..., L_{T,M_t}(t)],$$

where $L_{T,j}(t)$ is defined in (1.43).

Case 1: If $\alpha = 1$, we have

$$\frac{dT(t)}{dt} = \frac{d}{dt}[L_{T,0}(t), L_{T,1}(t), ..., L_{T,M_t}(t)] = [0, L'_{T,1}(t), ..., L'_{T,M_t}(t)], \tag{4.6}$$

where

$$L'_{T,j}(t) = \sum_{s=0}^{j} \frac{(-1)^{j+s}(j+s)!}{(j-s)!} \frac{t^{s-1}}{T^s(s!)(s-1)!}, j = 1, 2, ..., M_t. \tag{4.7}$$

Case 2: For $\alpha \in (0, 1)$, we have

$${}_0^cD_t^\alpha T(t) = [0, L^\alpha_{T,1}(t), ..., L^\alpha_{T,M_t}(t)], \tag{4.8}$$

where ${}_0^cD_t^\alpha$ is the Caputo derivative and it is defined by

$${}_0^cD_t^\alpha L_{T,j}(t) = L^\alpha_{T,j}(t) = \sum_{s=0}^{j} \frac{(-1)^{j+s}(j+s)!\, t^{s-\alpha}}{(j-s)!\,(s!)T^s\Gamma(s+1-\alpha)}, j = 1, 2, ..., M_t. \tag{4.9}$$

Now we will calculate spatial derivatives of

$$L(x) = [L_{X,0}(x), L_{X,1}(x), ..., L_{X,M_x}(x)]^T.$$

For first order derivative of $L(x)$,

$$\frac{dL(x)}{dx} = [L'_{X,0}(x), L'_{X,1}(x), ..., L'_{X,M_x}(x)]^T. \tag{4.10}$$

Similarly for second order derivative of $L(x)$,

$$\frac{d^2L(x)}{dx^2} = [L''_{X,0}(x), L''_{X,1}(x), ..., L''_{X,M_x}(x)]^T, \tag{4.11}$$

and for $\beta \in (1, 2)$, we get

$${}_0^cD_x^\beta L(x) = [L^\beta_{X,0}(x), L^\beta_{X,1}(x), ..., L^\beta_{X,M_x}(x)], \tag{4.12}$$

where ${}_0^cD_x^\beta$ is Caputo derivative and it is defined as

$${}_0^cD_x^\beta L_{X,i}(x) = \sum_{s=0}^{i} \frac{(-1)^{i+s}(i+s)!\, x^{s-\beta}}{(i-s)!\,(s!\,)X^s\Gamma(s+1-\beta)}. \tag{4.13}$$

The parameter for the training set is chosen on uniform grid points as $x_m = \frac{(m-1)X}{(M_m-1)}$, $t_n = \frac{(n-1)T}{(M_n-1)}$, where $m \in \{1, 2, ..., M_m\}$ and $n \in \{1, 2, ..., M_n\}$.
Substituting (4.5) into the model (4.1) and using the equations (4.8), (4.10) and (4.12), the errors $er^q_{m,n}$ at sample points (x_m, t_n) for non-initial states

$m \in \{2, 3, ..., M_m - 1\}$ and $n \in \{2, 3, ..., M_n\}$ are calculated as

$$\begin{aligned} er^q_{m,n} = &\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L_{X,i}(x_m) L^{\alpha}_{T,j}(t_n) - \sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L^{\beta}_{X,i}(x_m) L_{T,j}(t_n) \\ &- \nu(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L_{X,i}(x_m) L_{T,j}(t_n))^{\eta}(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L'_{X,i}(x_m) L_{T,j}(t_n)) \\ &- \lambda(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L_{X,i}(x_m) L_{T,j}(t_n)) + \lambda(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L_{X,i}(x_m) L_{T,j}(t_n))^{\eta+1}. \end{aligned} \tag{4.14}$$

While for initial condition (4.2), $n = 1$ and $m \in \{1, 2, ..., M_m\}$ and we have

$$er^q_{m,1} = \sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L_{X,i}(x_m) L_{T,j}(t_1) - \Psi_1(x_m). \tag{4.15}$$

For boundary condition (4.3), $m = 1$ and $n \in \{2, ..., M_n\}$. Therefore,

$$er^q_{1,n} = \sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L_{X,i}(x_1) L_{T,j}(t_n) - \Psi_2(t_n), \tag{4.16}$$

and for boundary condition (4.4), $m = M_m$ and $n \in \{2, ..., M_n\}$ so that

$$er^q_{M_m,n} = \sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j} L'_{X,i}(x_{M_m}) L_{T,j}(t_n) - \Psi_3(t_n). \tag{4.17}$$

The q^{th} error matrix is defined by $E^q = (er^q_{m,n})_{M_m \times M_n}$. The Frobenius matrix norm of E^q matrix is defined by

$$||E^q||^2_F = \frac{1}{2}\sum_{m=1}^{M_m}\sum_{n=1}^{M_n}(er^q_{m,n})^2. \tag{4.18}$$

Next, the weight adjustment formula [144] is given by

$$u_{i,j}^{q+1} = u_{i,j}^{q} + \Delta u_{i,j}^{q}, \tag{4.19}$$

for $q = 0, 1, 2, ..., N$, with

$$\Delta u_{i,j}^{q} = -\rho \frac{\partial ||E^q||_F^2}{\partial u_{i,j}^{q}} = -\rho \sum_{m=1}^{M_m} \sum_{n=1}^{M_n} er_{m,n}^{q} \frac{\partial er_{m,n}^{q}}{\partial u_{i,j}^{q}}.$$

Case 1: When $m \in \{2, 3, ..., M_m - 1\}$ and $n \in \{2, 3, ..., M_n\}$, we get

$$\begin{aligned}
\frac{\partial er_{m,n}^{q}}{\partial u_{i,j}^{q}} &= L_{X,i}(x_m) L_{T,j}^{\alpha}(t_n) - L_{X,i}^{\beta}(x_m) L_{T,j}(t_n) \\
&- \nu \Big(\sum_{i=0}^{M_x} \sum_{j=0}^{M_t} u_{i,j}^{q} L_{X,i}(x_m) L_{T,j}(t_n) \Big)^{\eta} \big(L'_{X,i}(x_m) L_{T,j}(t_n) \big) \\
&- \nu\eta \Big(\sum_{i=0}^{M_x} \sum_{j=0}^{M_t} u_{i,j}^{q} L_{X,i}(x_m) L_{T,j}(t_n) \Big)^{\eta - 1} \big(L_{X,i}(x_m) L_{T,j}(t_n) \big) \\
&\times \Big(\sum_{i=0}^{M_x} \sum_{j=0}^{M_t} u_{i,j}^{q} L'_{X,i}(x_m) L_{T,j}(t_n) \Big) - \lambda \big(L_{X,i}(x_m) L_{T,j}(t_n) \big) \\
&+ \lambda(\eta + 1) \Big(\sum_{i=0}^{M_x} \sum_{j=0}^{M_t} u_{i,j}^{q} L_{X,i}(x_m) L_{T,j}(t_n) \Big)^{\eta} \big(L_{X,i}(x_m) L_{T,j}(t_n) \big).
\end{aligned}$$

Case 2: When $n = 1$ and $m \in \{1, 2, ..., M_m\}$, we obtain

$$\frac{\partial er_{m,1}^{q}}{\partial u_{i,j}^{q}} = L_{X,i}(x_m) L_{T,j}(t_1),$$

Case 3: When $m = 1$ and $n \in \{2, 3, ..., M_n\}$, we find

$$\frac{\partial er_{1,n}^{q}}{\partial u_{i,j}^{q}} = L_{X,i}(x_1) L_{T,j}(t_n).$$

Case 4: When $m = M_m$ and $n \in \{2, 3, ..., M_n\}$, we have

$$\frac{\partial er^q_{M_m,n}}{\partial u^q_{i,j}} = L'_{X,i}(x_{M_m})L_{T,j}(t_n).$$

Hence,

$$\begin{aligned}
\Delta u^q_{i,j} = -\rho \sum_{m=2}^{M_m-1}\sum_{n=2}^{M_n} er^q_{m,n}\Big\{ & L_{X,i}(x_m)L^{\alpha}_{T,j}(t_n) - L^{\beta}_{X,i}(x_m)L_{T,j}(t_n) \\
& - \nu(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j}L_{X,i}(x_m)L_{T,j}(t_n))^{\eta}(L'_{X,i}(x_m)L_{T,j}(t_n)) \\
& - \nu\eta(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j}L_{X,i}(x_m)L_{T,j}(t_n))^{\eta-1}(L_{X,i}(x_m)L_{T,j}(t_n)) \\
& \times (\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j}L'_{X,i}(x_m)L_{T,j}(t_n)) - \lambda(L_{X,i}(x_m)L_{T,j}(t_n)) \\
& + \lambda(\eta+1)(\sum_{i=0}^{M_x}\sum_{j=0}^{M_t} u^q_{i,j}L_{X,i}(x_m)L_{T,j}(t_n))^{\eta}(L_{X,i}(x_m)L_{T,j}(t_n))\Big\} \\
& - \rho\sum_{m=1}^{M_m} er^q_{m,1}(L_{X,i}(x_m)L_{T,j}(t_1)) - \rho\sum_{n=2}^{M_n} er^q_{1,n}(L_{X,i}(x_1)L_{T,j}(t_n)) \\
& - \rho\sum_{n=2}^{M_n} er^q_{M_m,n}(L'_{X,i}(x_{M_m})L_{T,j}(t_n)).
\end{aligned}$$

Initial values of weights $u^0_{i,j}$ for $i = 0, 1, 2, \cdots, M_x$ and $j = 0, 1, 2, \cdots, M_t$ can be chosen at random. The maximum number of training allowed is N and the learning rate of the neural network is ρ.

Algorithm.

1. Construct sample points (x_m, t_n), where $m = 1, 2, ..., M_m$, and $n = 1, 2, .., M_n$.

2. Generate the weights $u^0_{i,j}$ for $i = 0, 1, 2, ..., M_x$ and $j = 0, 1, 2, ..., M_t$ at random.

3. Calculate the norm of Frobenius matrix E^q, $q \leq N$.

4. Calculate the weights' increase $\Delta u_{i,j}^q$.

5. Adjust the weights' $u_{i,j}^q$ in accordance to the relation (4.19).

6. Take a rest if $||E^q||_F < \epsilon$ or training time $q < N$. Go to Step 3 if not.

7. Network test.

The theorem below discusses the learning rate ρ of NNM.

Theorem[144] Assume that the parameter settings for the neural networks for model (4.1) are given by spatio-temporal neurons $M_x \times M_t$ and the number of sample points are $M_m \times M_n$. If the Lipschitz condition with parameter L is satisfied by the reaction term $R(w)$ and the constant associated with the Lipschitz parameter L and reaction term $R(w)$ is $M_{R,L} > 0$, then in order to guarantee that the error function reduces with practise, should satisfy

$$0 < \rho < \frac{1 + \sqrt{M_m M_n}}{M_{R,L}^2 M_m M_n M_x M_t}.$$

4.2 Numerical Application

This section focuses on validating the suggested approach by employing it to solve two well-known standard test cases.

4.2.1 Example [5]

Let us consider the following FRADE as

$$\frac{\partial^{\alpha} u(x,t)}{\partial t^{\alpha}} = \frac{\partial^{\beta} u(x,t)}{\partial x^{\beta}} + u(x,t)\frac{\partial u(x,t)}{\partial x} + \lambda u(x,t)(1-u(x,t)), \tag{4.20}$$

where $u(x,t)$ is a continuous function on $[0,1]\times[0,1]$ with initial condition

$$u(x,0) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{4}\right), \tag{4.21}$$

and boundary conditions

$$u(0,t) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{5t}{8}\right), \tag{4.22}$$

$$\frac{\partial u(1,t)}{\partial x} = \frac{1}{8}\operatorname{sech}^2\left(\frac{1}{4}\left(1+\frac{5t}{2}\right)\right), \tag{4.23}$$

which has the exact solution [5] at $\alpha = 1$, $\beta = 2$ and $\lambda = 1$ as

$$u(x,t) = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{1}{4}\left(x+\frac{5t}{2}\right)\right). \tag{4.24}$$

This example represents a nonlinear model which is particularly challenging to solve. The goal of this example is to evaluate the precision of NNM when solving a nonlinear model. The following parameters are used to set up the model for spatial interval $X = 1$, time interval $T = 1$, $\eta = 1$, learning rate $\rho = 5\times10^{-5}$, $M_x = M_t = M_m = M_n = 5$. We can see from Table 4.1 that as number of training sets is increased, the error reduces and the accuracy of NNM improves. The comparison of the numerical solutions estimated by employing NNM with different number of training data points and exact solution is shown in Figure 4.2. The graph perfectly demonstrates that increased number of training sets leads to enhanced numerical accuracy and also demonstrates that there is good agreement between the exact and numerical solutions.

N	Error
$10,000$	2.607×10^{-4}
$20,000$	5.1217×10^{-5}
$30,000$	9.8995×10^{-6}
$40,000$	1.6785×10^{-6}

Table 4.1 Frobenius norm of error matrix for Example 1 at different number of training sets.

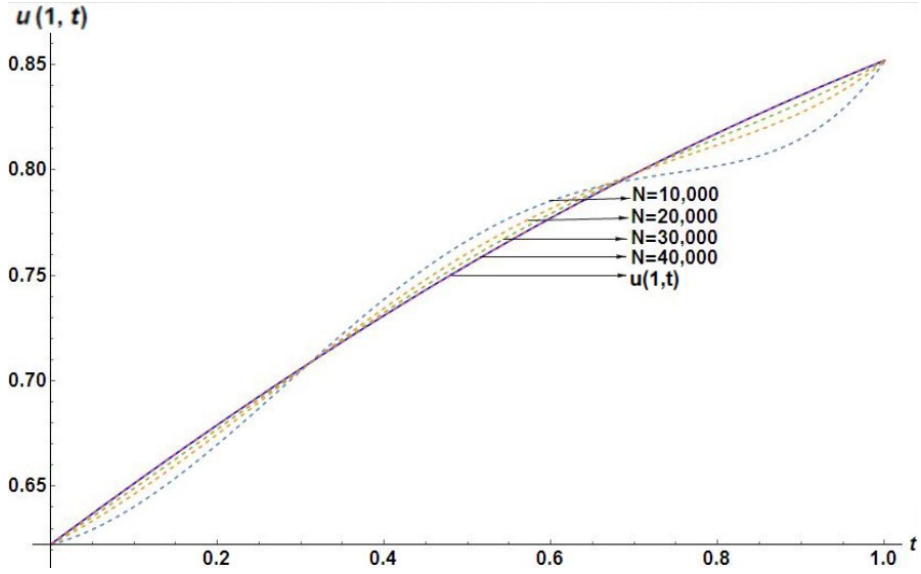

Figure 4.2: Comparison of numerical solution of Example 1 for $N = 10,000$, $N = 20,000$, $N = 30,000$ and $N = 40,000$ with the exact solution at $x = 1$.

4.2.2 Example [6]

Let us consider the following FRADE given by

$$\frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = \frac{\partial^\beta u(x,t)}{\partial x^\beta} + u^2(x,t)\frac{\partial u(x,t)}{\partial x} + u(x,t)(1-u^2(x,t)), \tag{4.25}$$

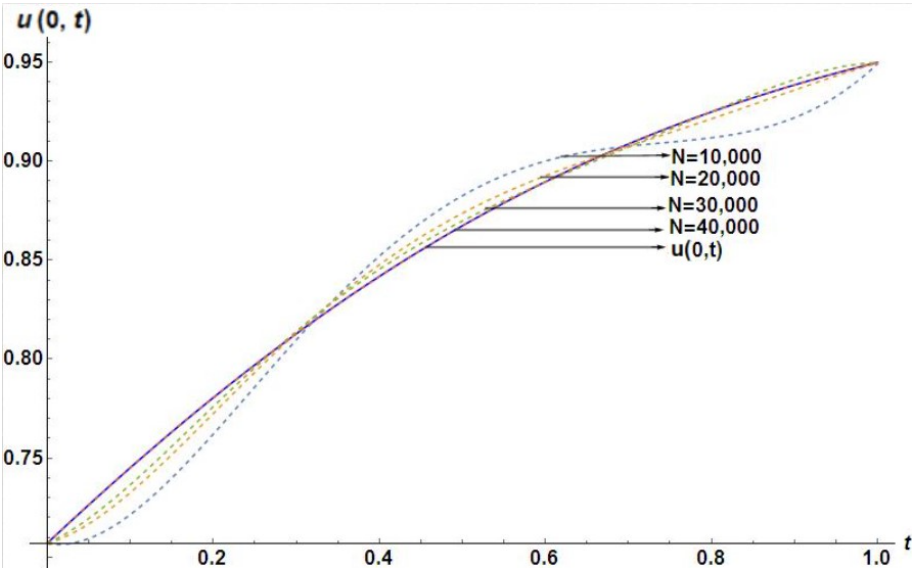

Figure 4.3: Comparison of numerical solution of Example 2 for $N = 10,000$, $N = 20,000$, $N = 30,000$ and $N = 40,000$ with the exact solution at $x = 0$.

where $u(x,t)$ is a function in $C[0,1] \times C[0,1]$ with initial condition

$$u(x,0) = \sqrt{\frac{1}{2} + \frac{1}{2}\tanh\left(\frac{x}{3}\right)}, \tag{4.26}$$

and boundary conditions

$$u(0,t) = \sqrt{\frac{1}{2} + \frac{1}{2}\tanh\left(\frac{10t}{9}\right)}, \tag{4.27}$$

$$\frac{\partial u(1,t)}{\partial x} = \frac{\operatorname{sech}^2\left(\frac{1}{3}(1+\frac{10t}{3})\right)}{6\sqrt{2}\sqrt{1+\tanh\left(\frac{1}{3}(1+\frac{10t}{3})\right)}}, \tag{4.28}$$

which has the exact solution at $\alpha = 1$, $\beta = 2$ and $\lambda = 1$ as [6]

$$u(x,t) = \sqrt{\frac{1}{2} + \frac{1}{2}\tanh\left(\frac{1}{3}(x+\frac{10t}{3})\right)}. \tag{4.29}$$

As in previous example, this example also aims to demonstrate how NNM solves a nonlinear model accurately for $\eta = 2$ for the model (4.1). The model is configured using the parameters viz., spatial interval $X = 1$, time interval $T = 1$, and learning rate $\rho = 5 \times 10^{-5}$, $M_x = M_t = M_m = M_n = 5$. Table 2 demonstrates that as number of training sets is extended, the error decreases and NNM accuracy increases. Figure 4.3 depicts the comparison of the exact solution and numerical results for different numbers of training sets. The figure clearly shows that more number of training sets result in higher numerical accuracy. The agreement between the numerical and exact results is also demonstrated in Figure 4.3.

N	$Error$
$10,000$	4.0862×10^{-4}
$20,000$	5.8519×10^{-5}
$30,000$	8.57706×10^{-6}
$40,000$	1.1286×10^{-6}

Table 4.2 Frobenius norm of error matrix for Example 2, at different number of training sets.

4.3 Application of NNM to solve the considered nonlinear FRADE

This section takes the following initial condition and boundary conditions into consideration as it attempts to solve the nonlinear FRADE given in (4.1).

$$u(x, 0) = x^2, \quad 0 \leq x \leq 1, \tag{4.30}$$

$$u(0, t) = 0, \quad 0 \leq t \leq 1, \tag{4.31}$$

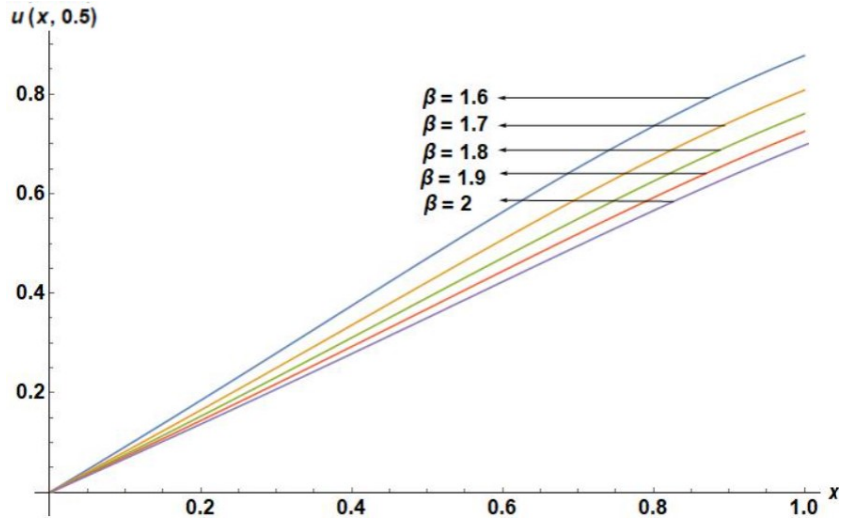

Figure 4.4: Plots of concentration of the solute at fixed $t = 0.5$ for $\alpha = 0.6$, $\beta = 1.6, 1.7, 1.8, 1.9, 2$, $\lambda = -1$ and $\nu = 0.6$ where training set $N = 30,000$.

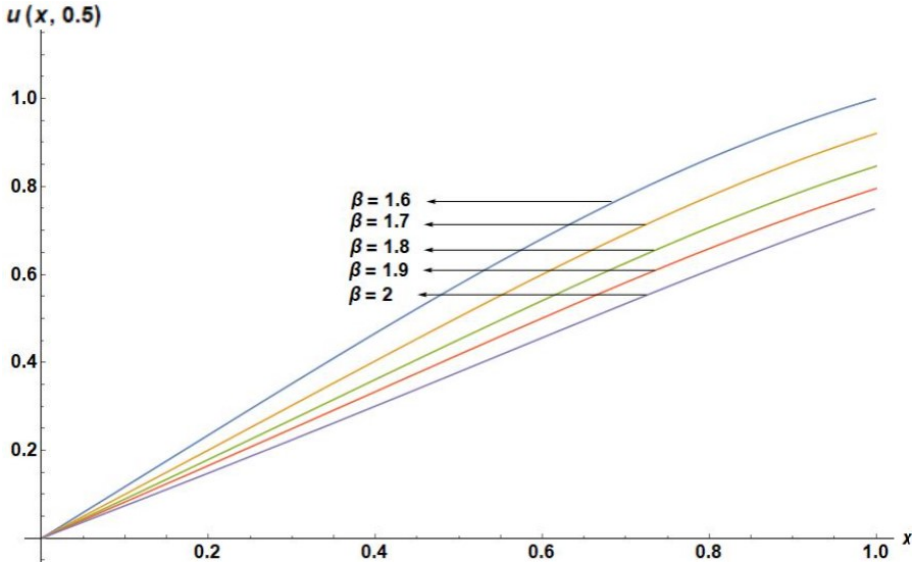

Figure 4.5: Plots of concentration of the solute at fixed $t = 0.5$ for $\alpha = 1$, $\beta = 1.6, 1.7, 1.8, 1.9, 2$, $\lambda = -1$ and $\nu = 0.6$ where training set $N = 30,000$.

$$\frac{\partial u(1,t)}{\partial x} = e^{-t}, \quad 0 \leq t \leq 1. \tag{4.32}$$

Our goal is to use the validated numerical method based on NNM to solve the model (4.1) with initial condition (4.30) and boundary conditions (4.31) and (4.32) for different particular cases. Figure 4.4 shows the variations of solution for fixed $\alpha = 0.6$

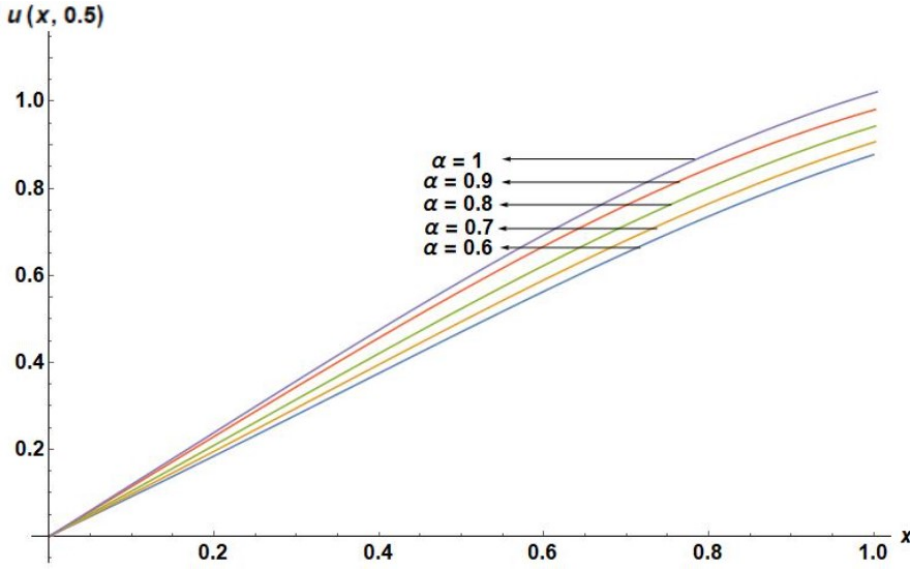

Figure 4.6: Plots of concentration of the solute at fixed $t = 0.5$ for $\beta = 1.6$, $\alpha = 0.6, 0.7, 0.8, 0.9, 1$, $\lambda = -1$ and $\nu = 0.6$ where training set $N = 30,000$.

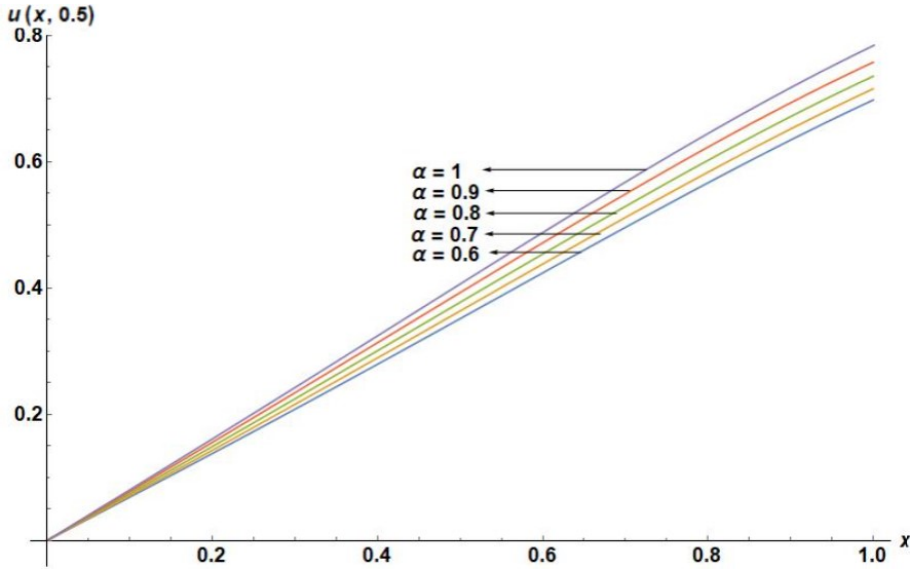

Figure 4.7: Plots of concentration of the solute at fixed $t = 0.5$ for $\beta = 2$, $\alpha = 0.6, 0.7, 0.8, 0.9, 1$, $\lambda = -1$ and $\nu = 0.6$ where training set $N = 30,000$.

and $\beta = 1.6, 1.7, 1.8, 1.9, 2$, and advection coefficient $\nu = 0.6$, at $\lambda = -1$, whereas Figure 4.5 depicts the nature of solution for $\alpha = 1$ and $\beta = 1.6, 1.7, 1.8, 1.9, 2$, and $\nu = 0.6$ at $\lambda = -1$. The graph shows that the concentration of the solute drops as the order of the spatial derivative β moves from the fractional-order to the standard

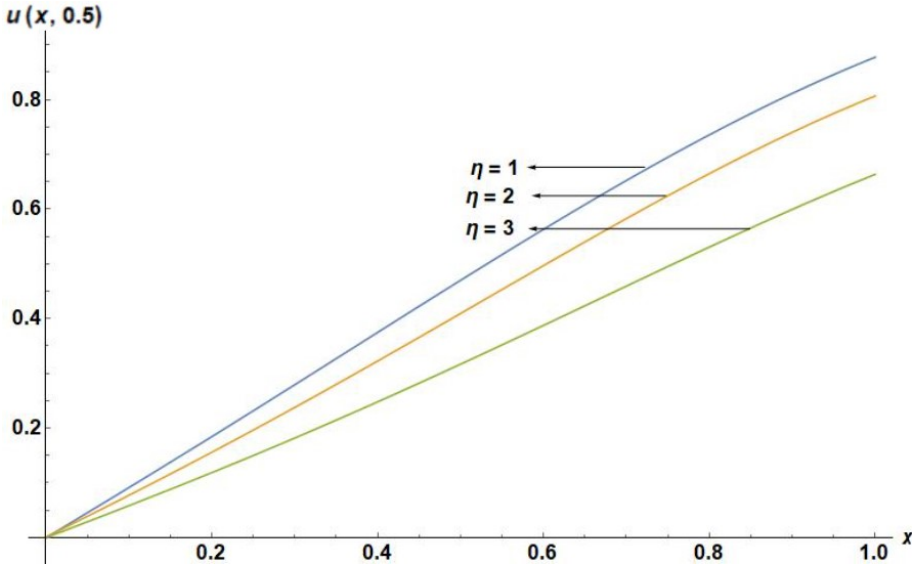

Figure 4.8: Plots of concentration of the solute at fixed $t = 0.5$ for $\alpha = 0.6$, $\beta = 1.6$, $\eta = 1, 2, 3$ and $\nu = 0.6$ where training set $N = 30,000$.

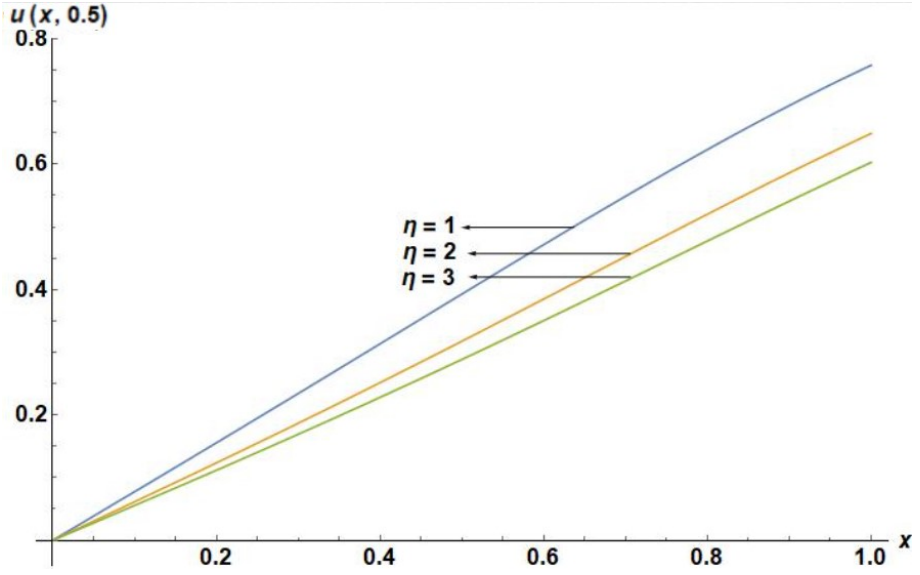

Figure 4.9: Plots of concentration of the solute at fixed $t = 0.5$ for $\alpha = 1$, $\beta = 2$, $\eta = 1, 2, 3$ and $\nu = 0.6$ where training set $N = 30,000$.

order. The solute travels a shorter distance in the soil column when $\beta = 2$ (standard order case) as compared to a fractional-order system, which is physically justified. Figure 4.6 shows the solutions for fixed $\beta = 1.6$ and $\alpha = 0.6, 0.7, 0.8, 0.9, 1$ and advection coefficient $\nu = 0.6$ at $\lambda = -1$ and Figure 4.7 depicts the nature of solution at fixed $\beta = 2$ and $\alpha = 0.6, 0.7, 0.8, 0.9, 1$ and $\nu = 0.6$ at $\lambda = -1$. These graphical

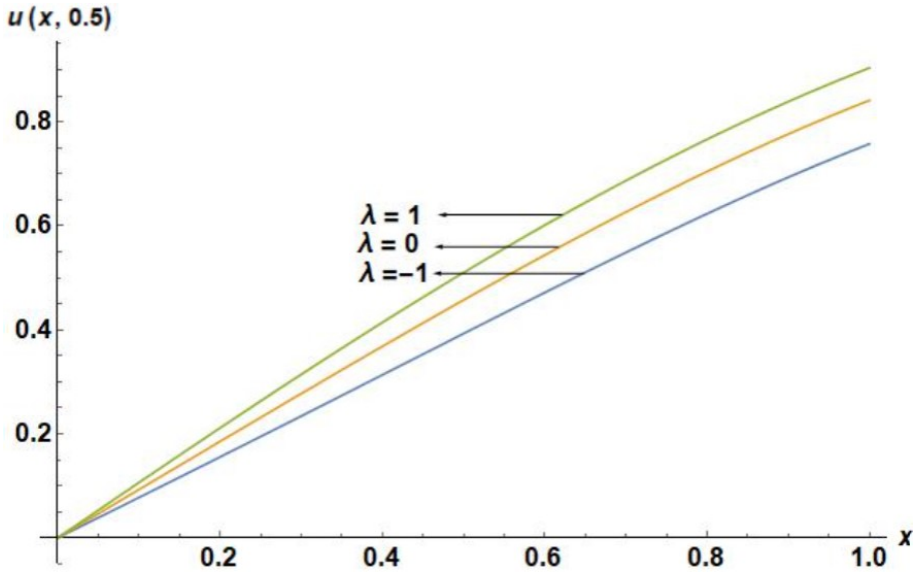

Figure 4.10: Plots of concentration of the solute at fixed $t = 0.5$ for $\alpha = 1$, $\beta = 2$, $\lambda = 1, 0, -1$ and $\nu = 0.6$ where training set $N = 30,000$.

representations demonstrate how the solute covers more length as the order of the time derivative α goes from fractional-order to standard order. Figure 4.8 and Figure 4.9 depict the solution profiles at fixed $\beta = 1.6$, $\alpha = 0.6$ and $\beta = 2$, $\alpha = 1$, respectively for $\lambda = -1$, $\nu = 0.6$ and $\eta = 1, 2, 3$. These graphical results show that the solute covers less distance in soil column with the increase in the order of nonlinearity in the reaction term.

Figure 4.10 shows the numerical solution obtained at fixed $\beta = 2$ and $\alpha = 1$ at $\lambda = -1, 0, 1$ and $\nu = 0.6$. It is observed that for the fixed values of α and β, the concentration of the solute is less for the system with sink term (λ = -1), as compared to the system with conservative contaminant ($\lambda = 0$) and for the system with source term ($\lambda = 1$). This is physically justified that more damping will be found in the presence of sink term (λ = -1) as compared to the source term ($\lambda = 1$) as well as for the case of conservative system ($\lambda = 0$). It is noticed in all the graphs that the nature of the solution is similar for each case.

4.4 Conclusion

The nonlinear FRADE studied in this chapter is an attempt to determine its approximate solution under the specified initial and boundary conditions using neural network method (NNM). This method includes training using a limited number of sample points in order to determine the weights of neurons. The trained neural network is then utilized for obtaining the solutions for the PDEs under unknown conditions. The applicability of the method is demonstrated by employing it to two test cases where exact solutions are known. Then the NNM is utilized to solve a FRADE. The results explain that the concentration of solution profile decreases when the time derivative of the system approaches a standard order from the fractional order. The most crucial finding of the chapter is that, as the spatial order derivative decreases, the solute travels a shorter distance in the soil column for the standard order case ($\beta = 2$) as compared to a fractional-order system. As a result, the concentration of solute will cover a longer soil column length, which is evident from the visual representations of the data. The successful implementation of NNM for solving FRADE as done in the present work opens the doors for future investigations on suitable basis functions for additional boundary conditions. After achieving the outstanding results while applying on the nonlinear time-space FRADE, the proposed method can be confidently applied to handle the nonlinear FRADE in two dimensional case and also for solving many such nonlinear FPDEs under the prescribed Robin or Neumann boundary conditions having physical relevances. Moreover there is significant scope of upgrading the proposed method NNM to increase the accuracy, normalization and dropout of the method which will be taken care in near future during handling the nonlinear problems in fractional as well as integer order systems.

****