Neural Processing Letters (2021) 53:2687–2699
https://doi.org/10.1007/s11063-021-10513-x

Numerical Solution of Fractional Order Advection Reaction Diffusion Equation with Fibonacci Neural Network

Kushal Dhar Dwivedi[1] · **Rajeev**[1]

Accepted: 9 April 2021 / Published online: 24 April 2021

Abstract
The authors have developed an efficient method with the Fibonacci neural network's help to solve the fractional-order reaction-diffusion equation in the present article. The Fibonacci neural network consists of an input layer with one perceptron, a hidden layer with $n \times m$ perceptions, and an output layer with one perceptron. The authors have used various degrees of the Fibonacci polynomial as an activation function to the input in the hidden layer. The authors then convert the fractional order diffusion equation with initial and boundary conditions into a non-constrained optimization problem, which is then called the cost function. Marquardt's method is used to update the values of weights to minimize the cost function. After that, the authors used the discussed method on four examples with an exact solution and showed that our method works more accurately than previously existing methods through comparison. In the last, authors have used the discussed method to solve the unsolved diffusion equation and observe the change in solute concentration for different fractional-order at different times.

Keywords Diffusion equation · Neural network · Fibonacci polynomial

1 Introduction

In recent years, a artificial neural network (ANN) is getting popular due to its ability to do unbelievable tasks, for example, finding such patterns in data which is nearly impossible for human being to find, image processing, speech recognition, in manufacturing virtual assistant like Alexa, training machine that can perform more efficiently than a human, and much more. ANN is also performing in such an area very efficiently, where we couldn't imagine one or two decades ago. Now the main question is if ANN can help us find the solutions of fractional order advection reaction-diffusion equations (FRDE) more accurately than existing traditional methods. Due to the rapid increase in popularity of artificial intelli-

✉ Kushal Dhar Dwivedi
kddwivedi1993@gmail.com

Rajeev
rajeev.apm@iitbhu.ac.in

[1] Department of Mathematical Sciences, Indian Institute of Technology (BHU), Varanasi 221005, India

gence, many authors have used the artificial neural network(ANN) to solve partial differential equations (PDEs) in the last few years, viz., K. Rudd and S. Ferrari [13] have solved PDEs by developing the method through the combination of classical Galerkin methods with a constrained backpropagation training approach in order to obtain the ANN representation of PDE solution. J. Berg and K. Nyström [1] have solved PDEs in complex geometries by using deep feed-forward ANN and modified the back-propagation algorithm to update the weights. M. Raissi [11] overcome the limitation of classical numerical methods on higher-dimensional PDEs by solving the Black-Scholes-Barenblatt and HamiltonJacobi-Bellman equations over 100 dimension with ANN. J. Berner et al. [2] have solved higher dimensional Black-Scholes PDEs with ANN and showed that curse of dimensionality of in numerical approximation can be solved easily with ANN. L. M. N. Tawfiq and O. M. Sailh [20] have developed the ANN with a decomposition approach to solve PDEs. Sun et al. [16] have used Bernstein neural network and extreme learning machine algorithms to solve PDEs. S. Mall and S. Chakraverty [6] solved the two-dimensional elliptic PDEs with the help of Chebyshev Neural Network and update the weights with the help of gradient decent method. Yang et al. [23] have used Legendre improved extreme learning machine based ANN to solve PDEs. S. Ghasemi and S. Effati [4] have solved distributed optimal control of Poisson's equation with Dirichlet boundary condition with the help of single layer ANN. The discussed approach in the present article can be used to find the numerical solution of any class of PDEs, but for convenience of discussion, the authors have taken the following FRDE.

$$\frac{\partial^{\alpha} u(x,t)}{\partial t^{\alpha}} = k(u,x,t)\frac{\partial^{\beta} u(x,t)}{\partial x^{\beta}} + f(u,x,t), \tag{1}$$

with the following types of initial and boundary conditions

$$\begin{aligned} u(x,0) &= f_0(x) \\ u(0,t) &= f_1(t) \\ u(1,t) &= f_2(t) \end{aligned} \tag{2}$$

where, $\frac{\partial^{\alpha} u(x,t)}{\partial t^{\alpha}}$ and $\frac{\partial^{\beta} u(x,t)}{\partial x^{\beta}}$ are fractional order derivative of $u(x,t)$ of order α and β in Caputo sense. Fractional diffusion equation plays a key role in many field such as statistical physics [7], neuroscience [5], economy [15], control theory [22] and combustion science [9]. As many phenomena are being described with fractional diffusion equations, finding an accurate solution of the fractional diffusion model (FDM) is a major challenge. Finding an exact solution is not always possible due to complexity in the FDM, so developing a numerical method that can provide accurate solutions is of great interest to today's researchers.

The article is arranged in the following order. In Sect. 2, the authors have discussed Caputo order fractional derivative, Fibonacci polynomial, the fractional derivative of a Fibonacci polynomial. In Sect. 3, the authors have discussed the structure of Fibonacci Neural Network (FNN) and methodology to solve the considered model (1). In Sect. 4, the authors have used the discussed method in Sect. 3 on four examples and showed that the present method works much efficiently than previously existing methods. In Sect. 5, the authors used the discussed method t solve the unsolved model and observed the change in solution due to change in different parameters graphically. The overall is concluded in Sect. 6.

2 Definitions

2.1 Caputo Fractional Order Derivative

In Caputo sense the fractional differential operator is defined as [10]

$$(D^{\alpha} f)(t) = \frac{1}{\Gamma(n-\alpha)} \int_0^t (t-\tau)^{(n-\alpha-1)} f^{(n)}(\tau) d\tau, \quad \alpha > 0, \tau > 0,$$

where $n-1 < \alpha < n, \quad n \in N,$

The operator D^{α} satisfies the following properties for $n-1 < \alpha < n$:

$$D^{\alpha} t^k = \begin{cases} 0 & k \in 0, 1, 2, ..., \lceil \alpha \rceil, \\ \frac{\Gamma(k+1)}{\Gamma(k+1-\alpha)} t^{k-\alpha}, & k \in N, \quad k \geq \lceil \alpha \rceil, \end{cases} \tag{3}$$

where the notation $\lceil \cdot \rceil$ denotes the ceiling function, which is defined by the set of equation $\lceil \alpha \rceil = \min\{n \in Z | n \geq \alpha\}$, Z is the set of integers.

2.2 Properties of Fibonacci Polynomial

It is known that Fibonacci Polynomial can be constructed by the following recurrence relation

$$F_{m+2}(x) = x F_{m+1}(x) + F_m(x), \quad m \geq 0,$$

with initial conditions as

$$F_0(x) = 0, \quad F_1(x) = 1.$$

From above relation, the explicit form of the series is obtained as

$$F_m(x) = \sum_{r=0}^{\lfloor \frac{m-1}{2} \rfloor} \binom{m-r-1}{r} x^{m-2r-1} \quad , \tag{4}$$

where $\lfloor \cdot \rfloor$ denotes the floor function, which is defined by $\lfloor \alpha \rfloor = \max\{n \in Z | n \leq \alpha\}$, Z is the set of integers.

Above equation can be rewritten as

$$F_i(x) = \sum_{j=0}^{i} \frac{(\frac{i+j-1}{2})!}{j!(\frac{i-j-1}{2})!} x^j, \quad (j+i) = odd, \quad i \geq 0. \tag{5}$$

The Caputo order derivative of Fibonacci polynomial of degree i can be obtained by using (3) in equation (5) as

$$D^{\alpha} F_i(x) = \begin{cases} 0 & i \in 0, 1, 2, ..., \lceil \alpha \rceil, \\ \sum_{\substack{j=\lceil \alpha \rceil \\ (j+i)=odd}}^{i} \frac{(\frac{i+j-1}{2})!}{(\frac{i-j-1}{2})!(j-\alpha)!} x^{j-\alpha}, & i \in N, \quad i \geq \lceil \alpha \rceil, \end{cases} \tag{6}$$

3 FNN and Method to Apply to Solve Considered Model

In this section, the authors discuss the FNN and how to apply it to solve the fractional order partial differential equation like model (1).

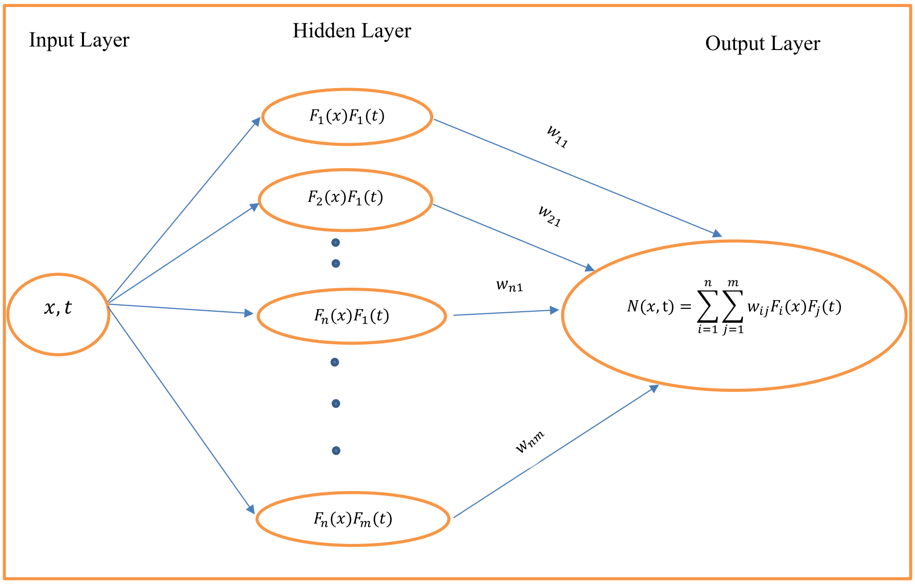

Fig. 1 Architecture of Fibonacci Neural Network to solve partial differential equation

3.1 Structure of Fibonacci Neural Network (FNN) Model

Figure 1 depicts an FNN, which consists of the input-output layer with one perceptron and a hidden layer of $n \times m$ perceptrons. Required numbers of perceptrons in the hidden layer will depend on the problem which is going to solve. $F_i(x)$ denotes the Fibonacci polynomial of degree i. The FNN will operate in the following way

- Inputs x and t will be given from the input layer.
- The inputs will be activated with different degree of Fibonacci polynomial in x and t in the hidden layer.
- The output layers is linear combinations of perceptrons of the hidden layer with weights(w).

3.2 Method to Apply the FNN to Solve FRDE

In this subsection we illustrate the method to apply the FNN to solve the considered model (1). Let us consider the trial solution of model (1) with initial and boundary conditions (2) is $u_t(x,t) = N(x,t)$. Where $N(x,t)$ is the output of FNN and is defined as

$$N(x,t) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} F_i(x) F_j(t).$$

Then the model (1) will be transformed as

$$\frac{\partial^\alpha N(x,t)}{\partial t^\alpha} = k(N,x,t)\frac{\partial^\beta N(x,t)}{\partial x^\beta} + f(N,x,t), \tag{7}$$

with the following types of initial and boundary conditions

$$\begin{aligned} N(x,0) &= f_0(x), \\ N(0,t) &= f_1(t), \\ N(1,t) &= f_2(t). \end{aligned} \tag{8}$$

where,

$$\begin{aligned} \frac{\partial^\alpha N(x,t)}{\partial t^\alpha} &= \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} F_i(x) \frac{\partial^\alpha F_j(t)}{\partial t^\alpha}, \\ \frac{\partial^\beta N(x,t)}{\partial x^\beta} &= \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} \frac{\partial^\alpha F_i(x)}{\partial t^\beta} F_j(t). \end{aligned} \tag{9}$$

$\frac{\partial^\alpha F_j(t)}{\partial t^\alpha}$ and $\frac{\partial^\alpha F_i(x)}{\partial t^\beta}$ can be obtained from equation (6). Now above model (7) with conditions (8) can be transformed in to following non-constrained minimization problem

$$\begin{aligned} &\min_w \left\{ \sum_{x_p,t_q} \left(\left(\frac{\partial^\alpha N(x_p,t_q)}{\partial t^\alpha} - k(N,x_p,t_q)\frac{\partial^\beta N(x_p,t_q)}{\partial x^\beta} \right.\right.\right. \\ &\quad - f(N,x_p,t_q))^2 + (N(x_p,0) - f_0(x_p))^2 + (N(0,t_q) - f_1(t_q))^2 \\ &\left.\left. (N(1,t_q) - f_1(t_q))^2 \right) \right\}, \end{aligned} \tag{10}$$

where, x_p and x_q are the training points. Now the only thing left is to find the values of $w's$ that minimize the equation (10). This process is called training the neural network. In the next subsection the authors have discussed the Marquardt's method to update the values of $w's$ that will minimize the equation (10).

3.3 Learning Algorithm of Fibonacci Neural Network (FNN)

Marquardt's [12] method is actually a combination of Cauchy's and Newton's optimization method. This is also a gradient based method. The process of using Marquardt's method to minimize equation (10) as follows

```
Step 1. Define $w^{(0)}$ (Randomly chose but not identical)
$M$ =maximum number of iteration
$\epsilon$ = convergence criteria
Step 2. Set $k = 0, \quad \lambda^{(0)} = 10^4$.
Step 3. Calculate $\nabla E(w^{(k)})$
Step 4. Is $E(w^{(k)}) < \epsilon$?
Yes: Go to Step 11
No: Continue
Step 5. Is $k \geq M$? Yes: Go to Step 11
No: Continue
Step 6. Calculate $s(w^{(k)}) = \left[H^{(k)} + \lambda^{(k)} I\right]^{-1} \nabla E(w^{(k)})$
Step 7. $w^{(k+1)} = w^{(k)} - s(w^{(k)})$
Step 8. Is $E(w^{(k+1)}) < E(w^{(k)})$?
Yes: Go to Step 9
No: Go to Step 10
```

Step 9. Set $\lambda^{(k+1)} = \lambda^{(k)}/4$ and $k = k + 1$. Go to **Step 3**
Step 10. $\lambda^{(k)} = 2\lambda^{(k)}$. Go to **Step 6**
Step 11. Print the result and stop.
where w^k, $E(w^{(k)})$, $\nabla E(w^{(k)})$ and $H^{(k)}$ denotes the values at k^{th} iteration and is equal to

$$E(w^{(k)}) = \sum_{x_p,t_q} \Bigg((\frac{\partial^\alpha N(x_p,t_q)}{\partial t^\alpha} - k(N,x_p,t_q)\frac{\partial^\beta N(x_p,t_q)}{\partial x^\beta} - f(N,x_p,t_q))^2 + (N(x_p,0) - f_0(x_p))^2 + (N(0,t_q) - f_1(t_q))^2 (N(1,t_q) - f_1(t_q))^2 \Bigg),$$

$$w^{(k)} = \begin{bmatrix} w_{11} \\ w_{21} \\ \vdots \\ w_{n1} \\ w_{n2} \\ \vdots \\ w_{nm} \end{bmatrix}_{(n\times m)} \qquad \nabla E(w^{(k)}) = \begin{bmatrix} \frac{\partial E(w^{(k)})}{\partial w_{11}} \\ \frac{\partial E(w^{(k)})}{\partial w_{21}} \\ \vdots \\ \frac{\partial E(w^{(k)})}{\partial w_{n1}} \\ \frac{\partial E(w^{(k)})}{\partial w_{n2}} \\ \vdots \\ \frac{\partial E(w^{(k)})}{\partial w_{nm}} \end{bmatrix}_{(n\times m)},$$

$$H^{(k)} = \begin{bmatrix} \frac{\partial^2 E(w^{(k)})}{\partial w_{11}^2} & \frac{\partial^2 E(w^{(k)})}{\partial w_{11}\partial w_{21}} & \cdot & \frac{\partial^2 E(w^{(k)})}{\partial w_{11}\partial w_{nm}} \\ \vdots & \vdots & \cdot & \vdots \\ \frac{\partial^2 E(w^{(k)})}{\partial w_{n1}\partial w_{11}} & \frac{\partial^2 E(w^{(k)})}{\partial w_{n1}\partial w_{21}} & \cdot & \frac{\partial^2 E(w^{(k)})}{\partial w_{n1}\partial w_{nm}} \\ \vdots & \vdots & \cdot & \vdots \\ \frac{\partial^2 E(w^{(k)})}{\partial w_{nm}\partial w_{11}} & \frac{\partial^2 E(w^{(k)})}{\partial w_{nm}\partial w_{21}} & \cdot & \frac{\partial^2 E(w^{(k)})}{\partial w_{nm}^2} \end{bmatrix}_{(nm\times nm)}.$$

In the next section the authors apply the discussed method in this section on some of the examples that have exact solutions and compare the numerical results with previously existing methods to show the accuracy of the method.

4 Numerical Examples

The authors have considered L_∞ error in the following examples which is defined as

$$L_\infty = ||u_{exact} - u_{numerical}||_\infty = \max_{i,j} \left|u_{exact}(x_i,t_j) - u_{numerical}(x_i,t_j)\right|.$$

Example 1 The following nonlinear diffusion equation

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

with the following initial and boundary conditions

$$u(x,0) = e^x, \quad 0 \le x \le 1,$$
$$u(0,t) = e^t, \quad t > 0,$$
$$u(1,t) = e^{1+t}, \quad t > 0,$$

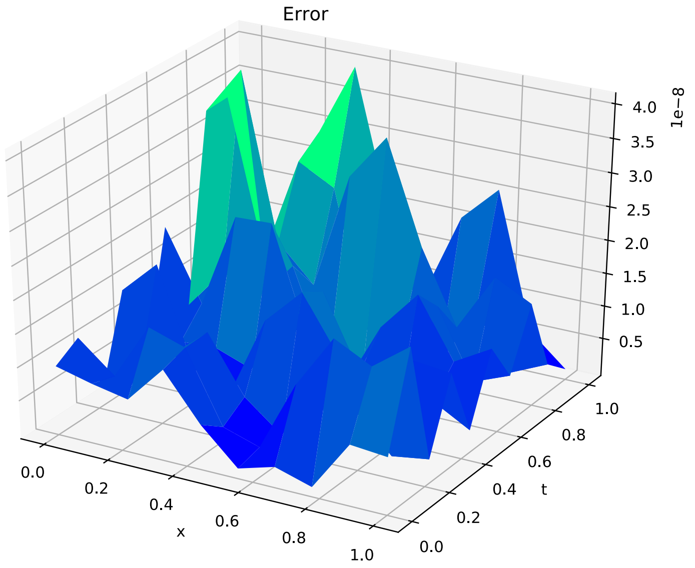

Fig. 2 Absolute error of Example 1

have the exact solution e^{x+t}. Now on solving this problem with discussed method for $n = m = 8$ over 35 iteration, we successfully get error of order $e-8$ which is shown in Fig. 2.

Momani et al. [8] have also solved this problem and end up with an error of order $e-4$ only. From the above Fig. 2, it can be seen that our method performs much better than this.

Example 2 Let us consider the partial differential equation

$$\frac{\partial^\alpha u(x,t)}{\partial t^\alpha} = \frac{\partial^2 u(x,t)}{\partial x^2} - \frac{\partial u(x,t)}{\partial x} + \frac{2t^{2-\alpha}}{\Gamma(3-\alpha)} + 2x - 2$$

with the following initial and boundary conditions

$$u(x,0) = x^2$$
$$u(0,t) = t^2$$
$$u(1,t) = 1 + t^2$$

have the exact solution $u(x,t) = x^2 + t^2$.

Many authors viz, Uddin and Haq [21] have solved Example 2 with radial basis functions (RBFs) approximation method for $\alpha = 0.5$ and end up with minimum error of order $e-2$. Ghandehari et al. [3] have solved above Example 2 by converting it into a nonlinear programming problem and end up with the minimum error of order $e-5$. From the Table 1, we can easily say that our proposed method is far more efficient than previously existing methods for $n = m = 3$ in very few iterations.

Example 3 The following diffusion equation

$$\frac{\partial u(x,t)}{\partial t} = p(x)\frac{\partial^{1.8} u(x,t)}{\partial x^{1.8}} + q(x,t),$$

Table 1 Maximum absolute error of example 2

α	Iteration	$\lVert u_{exact} - u_{numerical} \rVert_\infty$
0.1	26	6.6613e−16
0.3	25	1.1102e−15
0.5	25	8.8817e−16
0.7	25	1.3322e−15
1.0	26	7.2164e−16

where,

$$p(x,t) = \Gamma(1.2)x^{1.8}, \quad q(x,t) = 3x^2(2x-1)e^{-t},$$

with initial condition and boundary conditions

$$\begin{aligned} u(x,0) &= x^2(1-x), \quad 0 \le x \le 1, \\ u(0,t) &= 0, \quad t > 0, \\ u(1,t) &= 0, \quad t > 0, \end{aligned}$$

have the exact solution $u(x,t) = x^2(1-x)e^{-t}$. The authors have compared the absolute error obtained on solving this problem with the proposed method for $n = 4$ and $m = 10$ for 29 iterations with the absolute errors obtained by the previously existing method in Tables 2 and 3.

From the Tables 2, 3 and Fig. 3 it is very clear that presented method is very much accurate than previously existing methods.

Example 4 Let us consider the following diffusion equation

$$\frac{\partial u(x,t)}{\partial t} = p(x)\frac{\partial^{1.8} u(x,t)}{\partial x^{1.8}} + q(x,t),$$

where,

$$p(x,t) = \frac{\Gamma(2.2)}{6}x^{2.8}, \quad q(x,t) = -(1-x)x^3e^{-t},$$

Table 2 Comparison of absolute error obtained by different methods of example. 3 at T=1

x	Method [14]	Method [17]	Presented method
0.0	0.00	0.00	1.66e−13
0.1	4:66e−5	5.46e−6	1.60e−12
0.2	7.74e−5	8.51e−6	9.52e−12
0.3	5.00e−5	9.60e−6	2.12e−11
0.4	2.30e−5	9.18e−6	3.45e−11
0.5	2.74e−5	7.69e−5	4.70e−11
0.6	4.38e−5	5.60e−6	5.65e−11
0.7	3.87e−5	3.33e−6	6.05e−11
0.8	1.01e−5	1.34e−6	5.69e−11
0.9	3.35e−5	8.39e−8	4.32e−11
1.0	0.00	0.00	1.73e−11

Table 3 Comparison of absolute error obtained by different methods of example. 3 at T=2

x	Method [17]	Method [18]	Presented method
0.0	0.00	0.00	1.30e−11
0.1	3.33e−6	3.79e−10	2.97e−12
0.2	5.65e−6	6.26e−10	2.54e−12
0.3	7.05e−6	7.61e−10	8.98e−12
0.4	7.64e−6	7.99e−10	1.95e−11
0.5	7.52e−6	7.60e−10	3.14e−11
0.6	6.80e−6	6.59e−10	4.20e−11
0.7	5.59e−6	5.16e−10	4.58e−11
0.8	3.98e−6	3.46e−10	4.81e−11
0.9	2.08e−6	1.68e−10	3.81e−11
1.0	0.00	0.00	1.58e−11

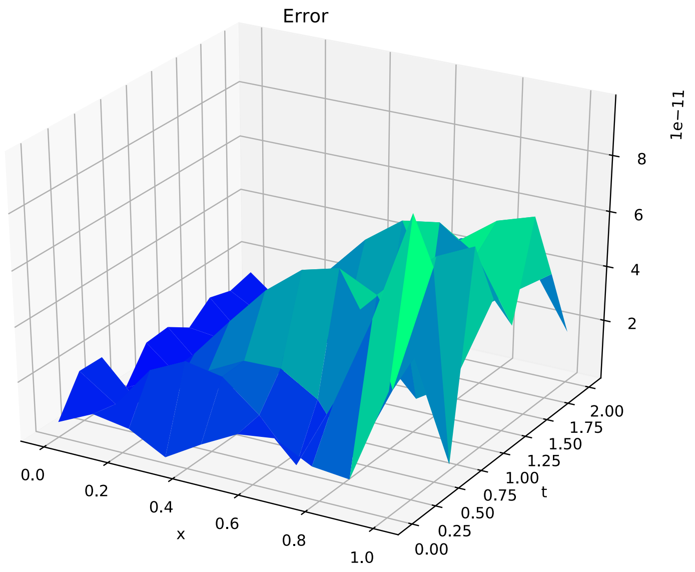

Fig. 3 Absolute error of Example 3

with initial condition and boundary conditions

$$u(x,0) = x^3, \quad 0 \le x \le 1,$$
$$u(0,t) = 0, \quad t > 0,$$
$$u(1,t) = e^{-t}, \quad t > 0,$$

which have the exact solution $x^3 e^{-t}$. The authors have compared the maximum absolute error obtained in solving the problem with the proposed method for $n = 4$ and $m = 9$ and previously existing methods in the Table 4.

So from the above table it is clear that our method works better than previously existing methods and takes very few number of iterating (56 for this problem) to reach this accuracy (Fig. 4).

Table 4 Comparison of maximum absolute error obtained by different methods of example. 4 at T=1

Max error CN [19]	Max error-ext CN [19]	Mex error [17]	Max error [18]	Max error Presented method
6:84895 e−4	2:82750e−5	8.3830−10	1.008e−10	2.31754e−11

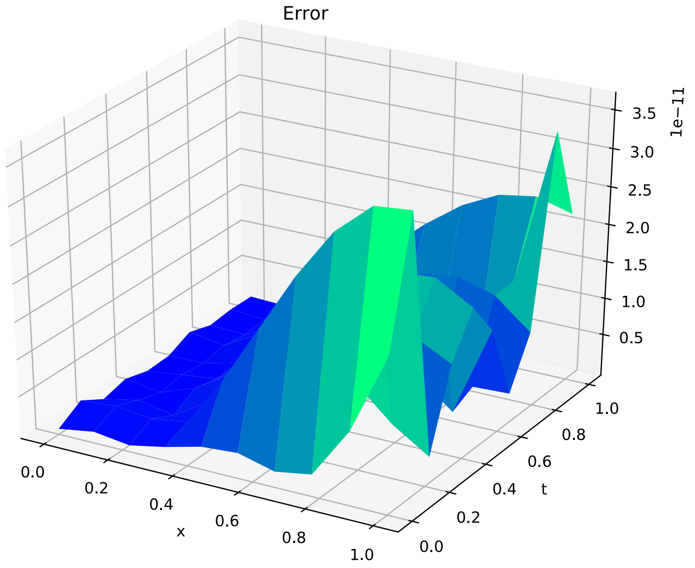

Fig. 4 Absolute error of Example 4

5 Solution of FRDE

The following problem has the exact solution for fractional-order space derivative $\beta = 1.8$, but we don't have the exact solution for different orders β. So in this section, the authors have solved this problem for different fractional-order derivative and showed the change in the value of $u(x, t)$ at different time levels.

$$\frac{\partial u(x,t)}{\partial t} = p(x)\frac{\partial^{\beta} u(x,t)}{\partial x^{\beta}} + q(x,t),$$

where,

$$p(x,t) = \frac{\Gamma(2.2)}{6}x^{2.8}, \quad q(x,t) = -(1-x)x^3 e^{-t},$$

with initial condition and boundary conditions

$$u(x,0) = x^3, \quad 0 \le x \le 1,$$
$$u(0,t) = 0, \quad t > 0,$$
$$u(1,t) = e^{-t}, \quad t > 0,$$

From Figs. 5, 6, 7 and 8, we can easily observe that on increasing in the order of fractional order space derivative, the value of $u(x, t)$ is increasing after a certain distance in the medium. On comparing Figs. 5, 6, 7 and 8, we can observe that on increasing the time, the change in $u(x, t)$ is increasing with the order of space derivative.

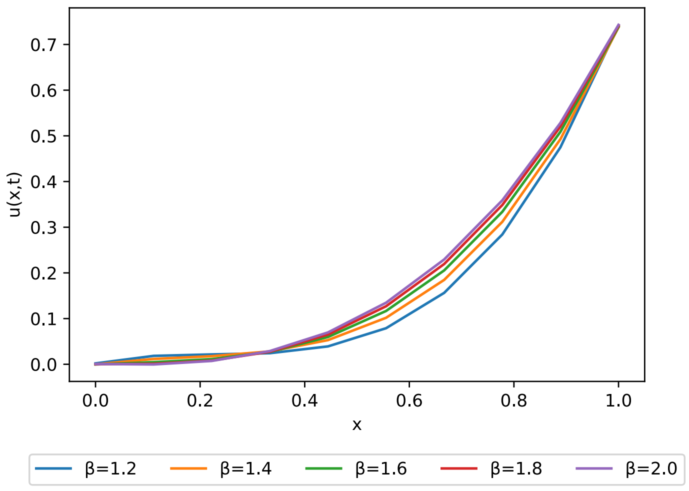

Fig. 5 Solution of FRDE at $T = 0.3$ for different fractional order β

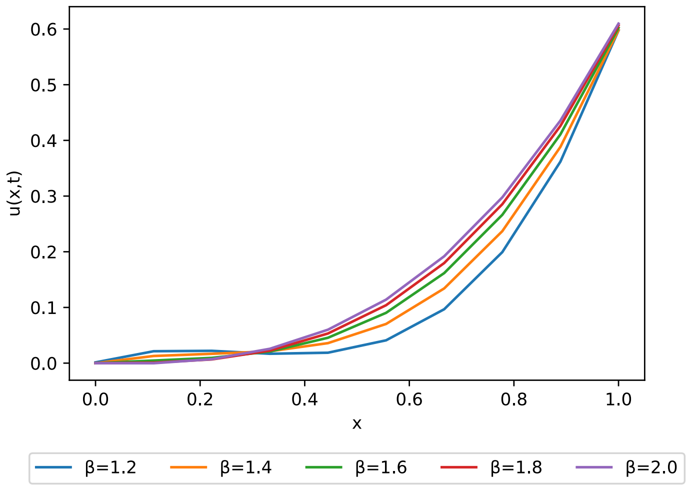

Fig. 6 Solution of FRDE at $T = 0.5$ for different fractional order β

6 Conclusion

In the last few decades, AI has done some extraordinary things that are nearly impossible to do for humans. So the authors believe that AI could help in finding the solution of the differential equations more accurately and easily. The author developed the present method with the help of a very basic ANN with one hidden layer and showed the proposed method is performing far better than the previously solved methods by applying it on four problems having an exact solution. As we have seen that only one hidden layer in the neural network is performing better than previously existing methods, so think of adding more hidden layers and more non-linearity in hidden layers. There are various things that can be done to FNN to increase its accuracy, like dropout, normalization, etc. for the method's future scope. Furthermore, the

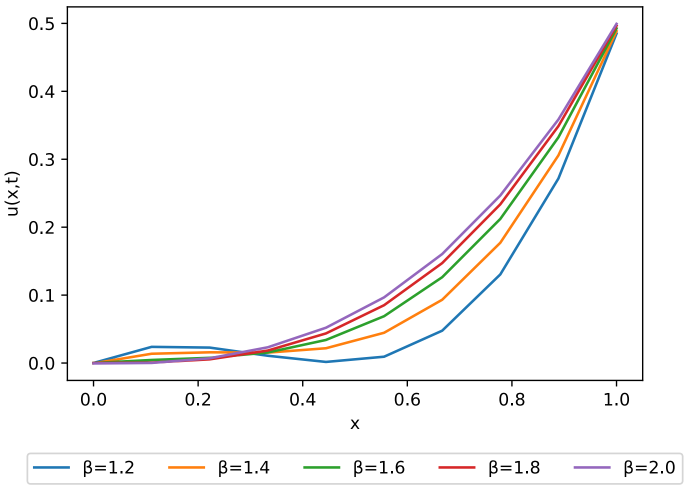

Fig. 7 Solution of FRDE at $T = 0.7$ for different fractional order β

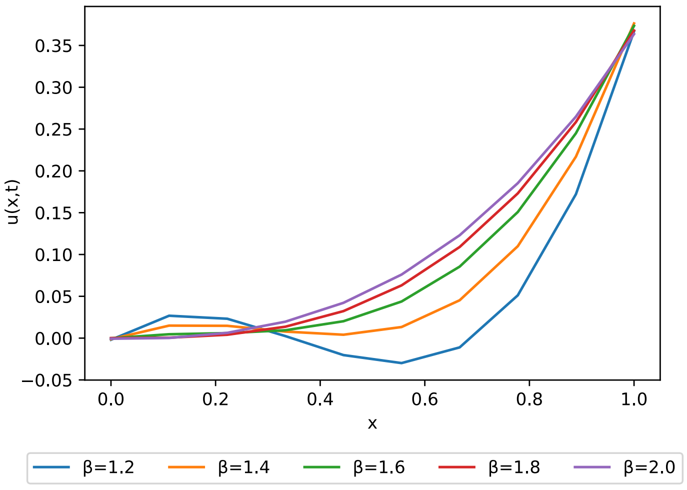

Fig. 8 Solution of FRDE at $T = 1.0$ for different fractional order β

authors used the proposed method to an unsolved fractional-order differential problem and observed change in the solution due to changes in different parameters of the problem.

Declaration

Conflicts of interest We have no conflict of interest to declare.

References

1. Berg J, Nyström K (2018) A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing 317:28–41
2. Berner J, Grohs P, Jentzen A (2018) Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black-scholes partial differential equations. arXiv preprint arXiv:1809.03062
3. Ghandehari MAM, Ranjbar M (2013) A numerical method for solving a fractional partial differential equation through converting it into an nlp problem. Comput Math Appl 65(7):975–982
4. Ghasemi S, Effati S (2019) An artificial neural network for solving distributed optimal control of the poisson's equation. Neural Process Lett 49(1):159–175
5. Lundstrom BN, Higgs MH, Spain WJ, Fairhall AL (2008) Fractional differentiation by neocortical pyramidal neurons. Nat Neurosci 11(11):1335
6. Mall S, Chakraverty S (2017) Single layer chebyshev neural network model for solving elliptic partial differential equations. Neural Process Lett 45(3):825–840
7. Metzler R, Klafter J (2004) The restaurant at the end of the random walk: recent developments in the description of anomalous transport by fractional dynamics. J Phys A Math General 37(31):R161
8. Momani S, Yıldırım A (2010) Analytical approximate solutions of the fractional convection-diffusion equation with nonlinear source term by he's homotopy perturbation method. Int J Comput Math 87(5):1057–1065
9. Pagnini G (2011) Nonlinear time-fractional differential equations in combustion science. Fract Calculus Appl Anal 14(1):80–93
10. Podlubny I (1998) Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications. Elsevier, New York
11. Raissi M (2018) Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. arXiv preprint arXiv:1804.07010
12. Ravindran A, Reklaitis GV, Ragsdell KM (2006) Engineering optimization: methods and applications. Wiley, London
13. Rudd K, Ferrari S (2015) A constrained integration (cint) approach to solving partial differential equations using artificial neural networks. Neurocomputing 155:277–285
14. Saadatmandi A, Dehghan M (2011) A tau approach for solution of the space fractional diffusion equation. Comput Math Appl 62(3):1135–1142
15. Scalas E (2006) The application of continuous-time random walks in finance and economics. Phys A Stat Mech Appl 362(2):225–239
16. Sun H, Hou M, Yang Y, Zhang T, Weng F, Han F (2019) Solving partial differential equation based on bernstein neural network and extreme learning machine algorithm. Neural Process Lett 50(2):1153–1172
17. Sweilam N, Nagy A, El-Sayed AA (2015) Second kind shifted chebyshev polynomials for solving space fractional order diffusion equation. Chaos Solit Fract 73:141–147
18. Sweilam N, Nagy A, El-Sayed AA (2016) On the numerical solution of space fractional order diffusion equation via shifted chebyshev polynomials of the third kind. J King Saud Univ Sci 28(1):41–47
19. Tadjeran C, Meerschaert MM, Scheffler HP (2006) A second-order accurate numerical approximation for the fractional diffusion equation. J Comput Phys 213(1):205–213
20. Tawfiq LNM, Salih OM (2019) Design neural network based upon decomposition approach for solving reaction diffusion equation. In: Journal of Physics: Conference Series, vol. 1234, p. 012104. IOP Publishing
21. Uddin M, Haq S (2011) Rbfs approximation method for time fractional partial differential equations. Commun Nonlinear Sci Numer Simul 16(11):4208–4214
22. Vinagre B, Podlubny I, Hernandez A, Feliu V (2000) Some approximations of fractional order operators used in control theory and applications. Fract Calcul Appl Anal 3(3):231–248
23. Yang Y, Hou M, Sun H, Zhang T, Weng F, Luo J (2020) Neural network algorithm based on legendre improved extreme learning machine for solving elliptic partial differential equations. Soft Comput 24(2):1083–1096