Iğdır Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 13(2), 1266-1274, 2023
Journal of the Institute of Science and Technology, 13(2), 1266-1274, 2023
ISSN: 2146-0574, eISSN: 2536-4618
Mathematics **DOI: 10.21597/jist.1230287**
Research Article
Received: 06.01.2023 **Accepted: 23.02.2023**
To Cite: Bulut, S. & Yiğider, M. (2023). Investigation of solutions of β −conformable fractional ordinary differential equation with artificial neural network. *Journal of the Institute of Science and Technology*, 13(2), 1266-1274.

Investigation of Solutions of β −conformable Fractional Ordinary Differential Equation With Artificial Neural Network

Sadullah BULUT[1*], Muhammed YİĞİDER[1]

Highlights:

- The neural network method was applied to systems of both fractional single ODEs and fractional coupled ODEs.
- Obtained results can be compared with different fractional derivative definitions.
- For numerical solutions of fractional differential equations, the topological structure of the feed forward artificial neural network was obtained.

Keywords:

- NeuralNetwork Method
- Numerical Solutions
- Fractional Differential Equations

ABSTRACT:

In this study, we present a method in order to get initial value fractional differential equations with artificial neural networks. On the basis of the function approach of feedforward neural networks, this method is a general method that is written in an implicit analytical form and results in the creation of a differentiable solution. The first part of the created trial solution which is stated as the sum of the two parts, with no controllable parameters, gives the initial conditions. The second part, unaffected by the initial conditions, consists of a feedforward neural network with controllable parameters (weights). The applicability of this approach is demonstrated in systems of both fractional single ODEs and fractional coupled ODEs.

[1] Sadullah BULUT (**Orcid ID:** 0000-0001-5026-6534), Muhammed YİĞİDER (**Orcid ID:** 0000-0003-4255-5760), Erzurum Teknik University Faculty of Science, Department of Mathematics, Erzurum, Türkiye

Corresponding Author: Sadullah BULUT, e-mail: sadullah.bulut@erzurum.edu.tr

INTRODUCTION

Fractional differential equations are widely benefited as generalizations of traditional differential equations in order to describe different complex phenomena in many fields such as diffusion of chemical kinematics, biological populations, plasma physics, solid state physics, signal processing, optical fiber, electricity etc. Therefore, a lot of efficient methods including generalized darboux transformation (Yang et al., 2022), galerkin finite element (Esen et al., 2013), exponential multistep methods (Zhan et al., 2022), variational iteration (Ain et al., 2022), sumudu transform (Karatas Akgül & Akgül, 2022), first integral method (Akinyemi et al., 2022), Wronskian determinant (Tang et al., 2021), modified exponential function (Tian & Liu, 2021), the modified auxiliary expansion (Akram et al., 2022), Adomian's decomposition (Kumar, 2022), inverse scattering (Gao et al., 2022), homotopy perturbation (Kocak et al., 2014), generalized Exp-function method (Shakeel et al., 2022) or modified trial equation method (Aderyani et al., 2022) have been given in previous studies in the literature.

Due to the wide range of usage areas, in this article we look at many of the methods presented so far for the solution of fractional differential equations from a different perspective. In our study, we present a general method based on the function approach of feedforward neural networks, which is written in a closed analytical form and results in the creation of a differentiable solution. This form uses a feedforward neural network whose parameters (weights and biases) are adjusted to minimize the error value as the basic approximation element (Lee & Kang, 1990). We use optimization techniques to train the network calling for the calculation of the gradient of the error according to the network parameters, respectively. Here the solution is stated as the sum of two parts: the first part gives the initial conditions and does not include any controllable parameters. The second part includes a feedforward neural network to be trained to give the differential equation. For being conscious of that a multilayer perceptron with a hidden layer can draw up any function to arbitrary accuracy, it is logical to think such a network architecture as a prospective model for handling differential equations.

When the literature on artificial neural networks is examined; it has been seen that neural networks are used to get approximate serial solutions of initial value ordinary differential equations with fractional degrees over a limited area (Jafarian et al., 2017), to parameterize the derivative of the hidden state using a neural network instead of specifying a separate set of hidden layers (Chen et al., 2018), to develop the legendre neural network method to solve linear and non-linear ordinary differential equations and the system of equations (Yang et al., 2018) and to create a partial differential equation solver based on collocation points and for the function approximation (Liu et al., 2019). In some of the studies, it has been mentioned that deep artificial neural networks can provide better accuracy for fewer network points to solve systems of ordinary differential equations, the moment minimization method and a vectorized algorithm are demonstrated applying it by Python (Dufera, 2021) and the neural network approach that simulates the behavior of partial differential equation systems using neural networks (Omidi et al., 2022).

Since artificial neural networks learn through examples, no prior knowledge is needed to understand the relationship of parameters in a complex problem. The relationships between the parameters on which the problem depends may not be linear. However, when the traditional method is used to solve the problem, certain points are approached linearly. This increases the fault tolerance for the system whose behavior we want to determine. Whether the relationships in artificial neural networks are linear or not is not a situation that needs to be focused in order to reach the desired system. The non-linearity of processes in artificial neural networks spreads this feature to the entire network structure, and this feature of artificial neural networks provides convenience for solving complex nonlinear problems. Neural networks can be retrained when new information emerges and data changes. Even if

some cells of artificial neural networks break down and become inoperable, the network continues to work. However, it can be said that artificial neural networks have fault tolerance, since the performance of the network may decrease depending on the importance of the damaged cells.

In the article, the important properties of the β −conformable fractional derivative will be mentioned, the structure of the artificial neural network will be explained and its applicability to the initial value problem for the β −order β −conformable fractional differential equation will be examined.

MATERIALS AND METHODS

The beginning of artificial neural networks has started with people's interest in neurobiology and their application to computer science. Artificial neural networks (Neural Networks-NNs) are computer systems inspired by biological nervous systems, developed with the aim of automatically performing abilities such as deriving new information and creating new information through learning being one of the features of the human brain without any assistance.

Artificial nerve cells are also called as process elements in engineering science. Each process element has five basic components. These elements are;

Inputs: The information that artificial neurons (processing elements) receive as inputs, is defined by the instances that the neural network purposes to learn from.

Weights: Weights show the effect of information coming into an artificial cell on the cell. The fact that the weights are large or relatively small does not indicate that the information is important or unimportant. Weights can change or take fixed values.

Addition Function: The net input to a cell is determined by the addition function. This is accomplished using several functions. The most frequent method is to compute the weighted sum. Here, each incoming input value is multiplied by its respective weight to determine the network's net input.

Activation Function: This function determines the net input to the cell and determines the corresponding output that the cell will produce as a result of this input. Various functions are used to calculate the output, however one of the most frequently used as activation function is the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$
$$\sigma' = -\sigma^2 + \sigma$$
$$\sigma'' = 2\sigma^3 - 3\sigma^2 + s$$
$$\sigma''' = -6\sigma^4 + 12\sigma^3 - 7\sigma^2 + \sigma$$
$$\sigma^{(4)} = 24\sigma^5 - 60\sigma^4 + 50\sigma^3 - 15\sigma^2 + s$$
$$.$$
$$.$$
$$.$$

Output of the Cell: The value determined by the activation function, the output produced is transmitted to the outside world or to another cell.

For $x = x_0 = a$, where $x \in [a, b]$,

$$ {}_0^A D_x^\beta y = f\big(x, y(x)\big),$$
$$ {}_0^A D_x^\beta (x_0) = y_0$$

Is the initial value problem for the β −order β −conformable fractional differential equation. Let the parameters of the feedforward artificial neural network be $\vec{p} = \vec{p}(\vec{\alpha},\vec{\beta},\vec{w})$ with $\vec{\alpha}$, $\vec{\beta}$ and $\vec{w} \in R$, where n is the total number of inputs and m is the number of neurons in the interlayer. For $j = 1, 2, \ldots, n$ the output of the neural network at the point $x_j \in [a, b]$ is shown as $NN(x_j, \vec{p})$ for the $\vec{p}$ parameter values. In general, the points acquired from the fragmentation of the interval $[a, b]$ such that $x_j = a + j.h$ for a fixed step length of $h > 0$ are used for training the network.

For the solution of the given initial value problem, the trial function that satisfies the initial conditions depending on the artificial neural network's output can be expressed as

$$y_T(x, \vec{p}) = y_0 + (x - x_0)NN(x; \vec{p})$$

As seen in Figure 1; for $i = 1, 2, 3, \ldots, m$, z_i represents i. neuron in the middle layer, with m indicating the number of neurons in the middle layer. For input x_i, the output of the z_i neuron is calculated as $z_i = w_i x_i + \beta_i$ when w_i is the weight and β_i is the threshold. The output of each neuron is weighted after processing with the activation function. The sum of the weighted outputs is determined as the output of the neural network. That is, when $\vec{p} = \vec{p}(\vec{\alpha},\vec{\beta},\vec{w})$ represents the unknown parameters of the network, the weighted sum value is attained with

$$NN(t_j, \vec{p}) = \sum_{i=1}^{m} \alpha_i \sigma(z_i)$$

This constitutes the output of the feedforward neural network. The error function that must be diminished is as follows:

$$E(p) = \sum_{i=1}^{m} \left\{ {}_0^A D_x^\beta (y_T(x, \vec{p}) - f[x_i, y_T(x, \vec{p})]) \right\}^2$$

Here, $\{x_i\}_{i=1}^m$ are some arbitrary discrete points in the range $[a, b]$. Our purpose is to minimize the error function to solve the differential equation. μ is the learning coefficient, which usually takes a value between 0 and 1; Gradient Descent Algorithm is used to update α_i, w_i ve β_i values for $i = 1, 2, \ldots, n$. In this approach, parameter values are updated to

$$\alpha_i = \alpha_i - \mu \frac{\partial E}{\partial \alpha_i}$$
$$w_i = w_i - \mu \frac{\partial E}{\partial w_i}$$
$$\beta_i = \beta_i - \mu \frac{\partial E}{\partial \beta_i}$$

respectively.

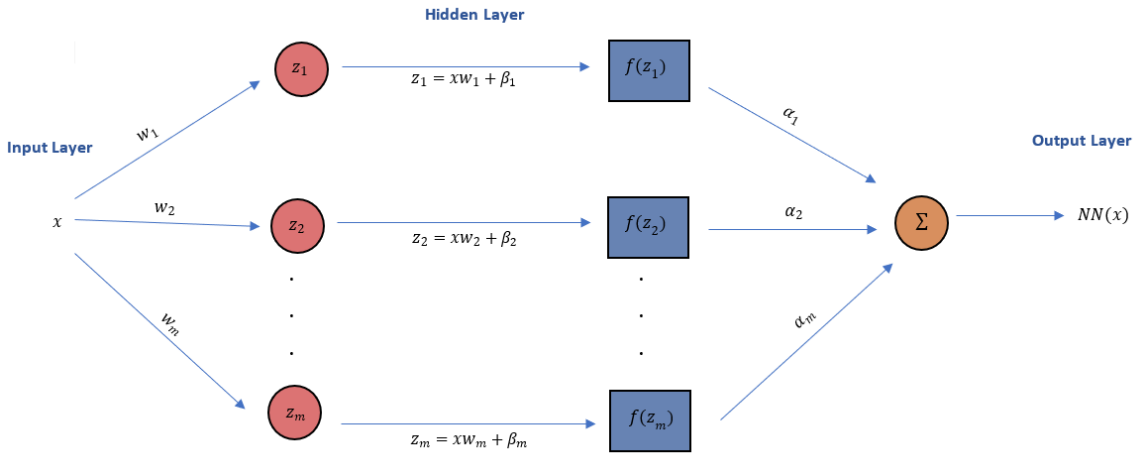

Figure 1. Topological structure of feedforward artificial neural network for numerical solutions of fractional differential equations

Definition 1. Let $a \in R$ and f be a function, such that, $f: [a, \infty) \to R$. Then, the β −derivative of f is defined as:

$$ {}_0^A D_x^\beta \{f(x)\} = \begin{cases} \lim_{\varepsilon \to 0} \dfrac{f\left(x + \varepsilon \left(x + \dfrac{1}{\Gamma(\beta)}\right)^{1-\beta}\right) - f(x)}{\varepsilon}, & 0 \le x \quad 0 < \beta \le 1 \\ f(x), & 0 \le x \qquad \beta = 0 \end{cases} $$

where f is a function such that $f: [0, \infty) \to R$ and the gamma-function

$$ \Gamma(\beta) = \int_0^\infty x^{\beta - 1} e^{-x} dx $$

If the above limit of exists, then f is said to be β −differentiable. Note that for $\beta = 1$, we have ${}_0^A D_x^\beta f(x) = \frac{d}{dx} f(x)$ (Atangana, 2015). Moreover, unlike other fractional derivatives, the β −derivative of a function can be locally defined at a certain point, the same way like first-order derivative.

Some important properties of the β −conformable fractional derivative are:

$$ {}_0^A D_x^0 f(x) = 0 $$

$$ {}_0^A D_x^\beta \big(\alpha f(x) + \mu g(x)\big) = \alpha\, {}_0^A D_x^\beta f(x) + \mu\, {}_0^A D_x^\beta g(x) $$

$$ {}_0^A D_x^\beta \big((fog)(x)\big) = {}_0^A D_x^\beta \Big(f\big(g(x)\big)\Big) g'(x) $$

$$ {}_0^A D_x^\beta (f^{-1}(x)) = -\frac{{}_0^A D_x^\beta (f(x))}{f^2(x)} $$

$$ {}_0^A D_x^\beta \big(f(x)g(x)\big) = {}_0^A D_x^\beta \big(f(x)\big) g(x) + {}_0^A D_x^\beta (g(x)) f(x) $$

$$ {}_0^A D_x^\beta \left(\frac{f(x)}{g(x)}\right) = \frac{{}_0^A D_x^\beta (f(x)) g(x) - {}_0^A D_x^\beta \big(g(x)\big) f(x)}{g^2(x)} $$

Definition 2. Given that $f:[a,b]\to R$ is a continuous function on the closed interval $[a,b]$, then, the 2α −derivative of f is described as the following:

$$ {}_0^A D_x^{2\beta}\big(f(x)\big) = {}_0^A D_x^{\beta}\Big({}_0^A D_x^{\beta}\big(f(x)\big)\Big), \qquad 0\le\beta\le 1 $$

The $n\beta$ −derivative of f is typically stated as:

$$ {}_0^A D_x^{2\beta}\big(f(x)\big) = {}_0^A D_x^{\beta}\left({}_0^A D_x^{(n-1)\beta}\big(f(x)\big)\right), \quad 0\le\beta\le 1 $$

Remark 1. Pointing out that the $n\beta$ −derivative of a given function gives information of the $(n-1)$ −derivatives coming before it, is very crucial.

As an example of:

$$ {}_0^A D_x^{2\beta}\big(f(x)\big) = \left(x+\frac{1}{\Gamma(\beta)}\right)^{1-\beta}\left[(1-\beta)\left(x+\frac{1}{\Gamma(\beta)}\right)^{-\beta} f' + \left(x+\frac{1}{\Gamma(\beta)}\right)^{1-\beta} f''\right] $$

As a result, it provides this derivative an unsurpassed memory characteristic that no other derivative has. Furthermore, it is simple to illustrate that we attain the second derivative of f if $\beta=1$.

RESULTS AND DISCUSSION

In this section, an application problem is solved in an attempt to illustrate the technique recommended in the previous section. With the enema solution acquired by the method, it converges to the exact solution and is completely stable. In order to demonstrate the behavior and features of the recommended method, we perform experiments on the following problem. All of the programs are written by Jupyter Notebook 6.4.8.

$$ {}_0^A D_x^{\beta} y + y = x^2 + 2x\sqrt{x+\frac{1}{\pi}} $$

with

$$ {}_0^A D_x^{\beta} y(0) = 0 \qquad \beta = \frac{1}{2} $$

and $x\in[0,1]$. The analytic solution which is depicted in Figure 2 is $y(x)=x^2$. The solution's trial neural form is presumed to be as follows, in accordance with equation (1):

$$ u_T(x,\vec{p}) = xNN(x;\vec{p}) $$

Values in the range $[0,1]$ consisting of 10 equal points are used to perform the training of the network. Also, the deviation of the approximate solution from the exact solution is presented in Figure 2. Despite the fact that only a few points have been utilized in the training, it is obvious that the solution is excellent accuracy. Additionally, for locations near to the equation field, the extrapolation error continues to be minimal.

The results attained with an ANN method in which the parameters are optimized using a gradient descent backpropagation algorithm to solve the β −order β −conformable fractional differential equation are visualized below.

Table 1. Comparison of the numerical solution got by artificial neural networks with the exact solution

Input Data	$u_{T_1}(x,\vec{p})$	$u_{T_1}(x,\vec{p})$	$u_{T_2}(x,\vec{p})$	$u_{T_2}(x,\vec{p})$	$y(x)$
0	0	0	0	0	0
0.01	0.0006702795	$0.67x10^{-3}$	0.00022911071	$0.22x10^{-3}$	$0.1x10^{-3}$
0.02	0.0014052297	$1.40x10^{-3}$	0.0005400422	$0.54x10^{-3}$	$0.4x10^{-3}$
0.03	0.00221075	$2.21x10^{-3}$	0.00094771583	$0.94x10^{-3}$	$0.9x10^{-3}$
0.04	0.003093045	$3.09x10^{-3}$	0.0014672052	$1.46x10^{-3}$	$1.6x10^{-3}$
0.05	0.0040586004	$4.05x10^{-3}$	0.0021132813	$2.11x10^{-3}$	$2.5x10^{-3}$
0.06	0.0051141502	$5.11x10^{-3}$	0.0028998812	$2.89x10^{-3}$	$3.6x10^{-3}$
0.07	0.006266749	$6.26x10^{-3}$	0.003839688	$3.83x10^{-3}$	$4.9x10^{-3}$
0.08	0.007523658	$7.52x10^{-3}$	0.0049438733	$4.94x10^{-3}$	$6.4x10^{-3}$
0.09	0.0088923415	$8.89x10^{-3}$	0.0062217684	$6.22x10^{-3}$	$8.1x10^{-3}$
0.10	0.010380539	$10.3x10^{-3}$	0.0076808045	$7.68x10^{-3}$	$10x10^{-3}$

The graphs of the exact solution $y(x) = x^2$ and the numerical solution of $u_T(x,\vec{p})$ for different training steps (10^3 and 10^6) in the [0,1] range are demonstrated in Figure 2.

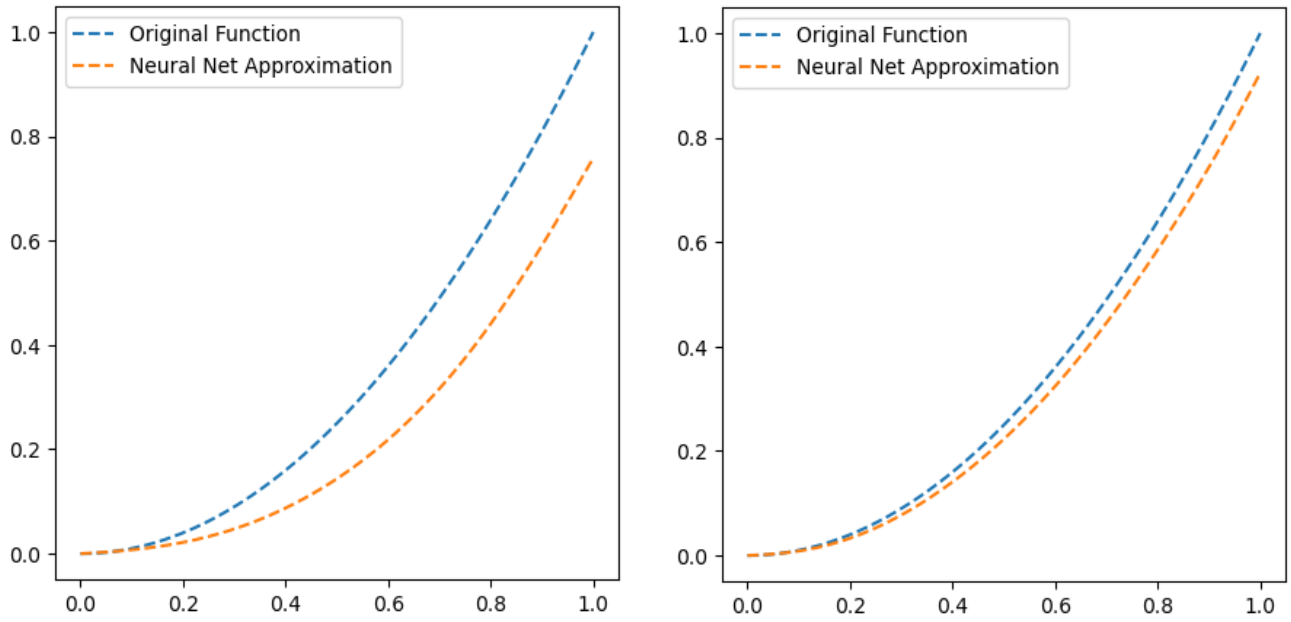

Figure 2. Comparison of the numerical solution acquired with artificial neural networks and the exact solution in the range of [0,1]

The biggest disadvantage of classical methods is that the solution sought in a certain interval is found only at the nodes attained from the fragmentation of the specified interval. Interpolation techniques are generally used to have the numerical solution of the differential equation at points other than the nodal points. However, this approach results in increased cumulative error. On the contrary, in the neural network in our example, the nodes got from the fragmentation of the gap are used only for the training of the network. In addition, after completing the network training in Figure 2, it is seen that it produces a solution at every point on the range.

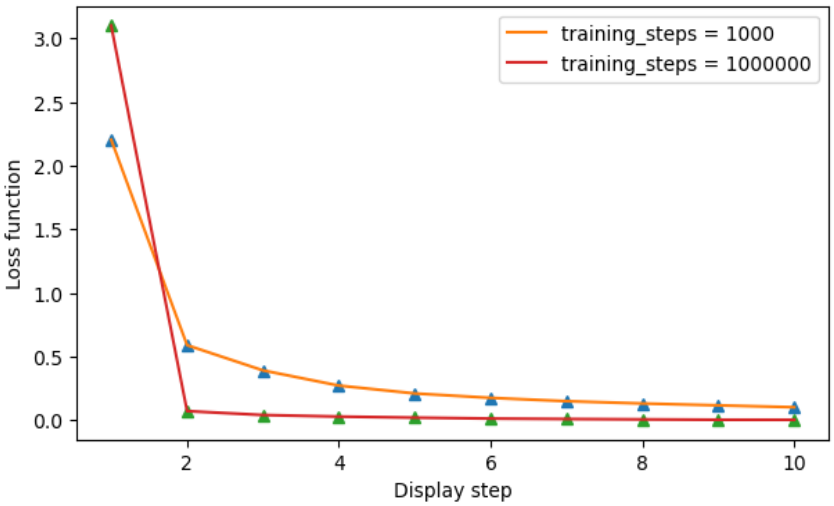

Figure 3. Loss function change according to different training steps values

CONCLUSION

In the article, the β −conformable fractional derivative, the structure of the artificial neural network and to get the solution of the initial value problem for the β −order β −conformable fractional differential equation were discussed. The originality of the study was achieved by getting the solutions of differential equations intuitively with computers without using theoretical methods. It was observed that there is no common artificial neural network structure in the numerical solutions of the differential equations, since different types of trial functions must be used, including the solution of the artificial neural network and satisfying the initial or boundary conditions of the differential equation. It is thought that the results of the current study got with different fractional derivative definitions can be compared and transferred to deep learning, which is a sub-branch of machine learning.

Conflict of Interest

The article authors declare that there is no conflict of interest between them.

Author's Contributions

The authors declare that they have contributed equally to the article.

REFERENCES

Aderyani, S. R., Saadati, R., Vahidi, J., & Allahviranloo, T. (2022). The exact solutions of the conformable time-fractional modified nonlinear Schrödinger equation by the Trial equation method and modified Trial equation method. *Advances in Mathematical Physics*, 2022.

Ain, Q. T., Nadeem, M., Karim, S., Akgül, A., & Jarad, F. (2022). Optimal variational iteration method for parametric boundary value problem. *AIMS Mathematics*, 7(9), 16649-16656.

Akinyemi, L., Mirzazadeh, M., Amin Badri, S., & Hosseini, K. (2022). Dynamical solitons for the perturbated Biswas–Milovic equation with Kudryashov's law of refractive index using the first integral method. *Journal of Modern Optics*, 69(3), 172-182.

Akram, G., Sadaf, M., & Zainab, I. (2022). The dynamical study of Biswas–Arshed equation via modified auxiliary equation method. *Optik*, 255, 168614.

Atangana, A. (2015). Derivative with a new parameter: Theory, methods and applications. Academic Press.

Chen, T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations, in 'Advances in neural information processing systems La Jolla.

Dufera, T. T. (2021). Deep neural network for system of ordinary differential equations: Vectorized algorithm and simulation. *Machine Learning with Applications*, 5, 100058.

Esen, A., Ucar, Y., Yagmurlu, N., & Tasbozan, O. (2013). A Galerkin finite element method to solve fractional diffusion and fractional diffusion-wave equations. *Mathematical Modelling and Analysis*, 18(2), 260-273.

Gao, Y., Liu, H., Wang, X., & Zhang, K. (2022). On an artificial neural network for inverse scattering problems. *Journal of Computational Physics* 448, 110771.

Jafarian, A., Mokhtarpour, M., & Baleanu, D. (2017). Artificial neural network approach for a class of fractional ordinary differential equation. *Neural Computing and Applications*, 28(4), 765-773.

Karatas Akgül, E., & Akgül, A. (2022). New applications of Sumudu transform method with different fractional derivatives. *International Journal of Applied and Computational Mathematics*, 8(5), 1-12.

Kocak, Z. F., Bulut, H., & Yel, G. (2014). The solution of fractional wave equation by using modified trial equation method and homotopy analysis method. *In AIP Conference Proceedings*, 1637(1), 504-512.

Kumar, M. (2022). Recent development of Adomian decomposition method for ordinary and partial differential equations. *International Journal of Applied and Computational Mathematics*, 8(2), 1-25.

Lee, H., & Kang, I. S. (1990). Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1), 110-131.

Liu, Z., Yang, Y., & Cai, Q. (2019). Neural network as a function approximator and its application in solving differential equations. *Applied Mathematics and Mechanics*, 40(2), 237-248.

Omidi, M., Arab, B., Rasanan, A. H., Rad, J. A., & Parand, K. (2022). Learning nonlinear dynamics with behavior ordinary/partial/system of the differential equations: looking through the lens of orthogonal neural networks. *Engineering with Computers*, 38(2), 1635-1654.

Shakeel, M., El-Zahar, E. R., Shah, N. A., & Chung, J. D. (2022). Generalized Exp-Function Method to Find Closed Form Solutions of Nonlinear Dispersive Modified Benjamin–Bona–Mahony Equation Defined by Seismic Sea Waves. *Mathematics*, 10(7), 1026.

Tang, Y., Ma, J., Zhou, B., & Zhou, J. (2021). From 2Mth-order wronskian determinant solutions to Mth-order lump solutions for the (2+1)-Dimensional Kadomtsev–Petviashvili I equation. *Wave Motion*, 104, 102746.

Tian, Y., & Liu, J. (2021). A modified exp-function method for fractional partial differential equations. *Thermal Science*, 25(2 Part B), 1237-1241.

Yang, D. Y., Tian, B., Hu, C. C., & Zhou, T. Y. (2022). The generalized Darboux transformation and higher-order rogue waves for a coupled nonlinear Schrödinger system with the four-wave mixing terms in a birefringent fiber. *The European Physical Journal Plus*, 137(11), 1-11.

Yang, Y., Hou, M., & Luo, J. (2018). A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods. *Advances in Difference Equations*, 2018(1), 1-24.

Zhan, R., Chen, W., Chen, X., & Zhang, R. (2022). Exponential Multistep Methods for Stiff Delay Differential Equations. *Axioms*, 11(5), 185.