Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Feedforward Neural Network for Solving Caputo Fractional Differential Equations with Application to Tumor Dynamics

Ng Chor Hong[1], Mohd Rashid Admon[1,*], Mohd Ariff Admon[1], Ali Ahmadian[2], Mohamad Shahiir Saidin[1], Noorehan Yaacob[1]

[1] Department of Mathematical Sciences, Faculty of Science, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia
[2] Decision Lab, Mediterranea University of Reggio Calabria, Reggio Calabria, Italy

ARTICLE INFO

Article history:
Received 27 March 2026
Received in revised form 21 May 2026
Accepted 10 June 2026
Available online 7 July 2026

Keywords:
Fractional Differential Equations; Feedforward Neural Network; Caputo fractional derivative; tumor dynamics

ABSTRACT

Fractional differential equations (FDEs) play an important role in modeling complex dynamical systems because they capture memory and hereditary effects more effectively than classical ordinary differential equations (ODEs). However, FDEs are generally more difficult to solve due to their nonlocal characteristics and higher computational complexity. Traditional numerical methods such as the predictor-corrector (PC) and Euler methods have been widely used but may face limitations in terms of computational cost and accuracy. Recently, artificial neural network (ANN) has emerged as a promising alternative for solving differential equations due to their strong approximation capability. This study proposes a neural network-based framework for solving FDEs by implementing a feedforward neural network (FNN) integrated with the Broyden-Fletcher-Goldfarb-Shanno(BFGS) optimization algorithm. The neural network parameters are initialized and iteratively updated using the BFGS algorithm by minimizing a loss function defined from the residual of the governing equation. The proposed method is evaluated on several linear and nonlinear FDE problems and further applied to a fractional tumor dynamics model. Numerical results show that the proposed method achieves higher accuracy and lower computational cost compared to the Adomian Decomposition Method (ADM) and previously published methods, while yielding a continuous and differentiable solution throughout the domain.

1. Introduction

1.1 Research Background

Fractional differential equations (FDEs) are important in physics, mathematics and engineering, for modelling dynamical systems in our daily lives. FDEs are necessary because they provide higher accuracy in representing these dynamical systems compared to ordinary differential equations (ODEs). There are traditional methods such as the predictor-corrector (OC) and Euler method to evaluate FDEs but they fall short in terms of accuracy and require high computation cost. Evidence suggests that traditional numerical techniques are not completely reliable to solve FDEs, leading researchers to explore artificial neural networks (ANN).

* *Corresponding author.*
E-mail address: m.rashid@utm.my

https://doi.org/10.37934/araset.61.5.6679

Artificial Neural Network (ANN) is a computational model that is based off of the human brain. It falls into the category of machine learning whereby it learns how to perform tasks by processing data repeatedly through interconnected neurons, much like how biological neurons function. Using mathematical equations, electronic circuits and even software, ANN can mimic the human brain [1]. The structure of an ANN is shown below in Figure 1.

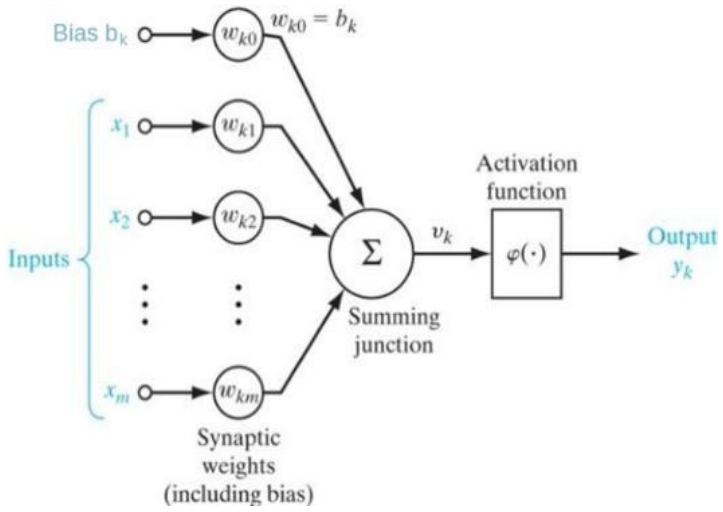

Fig. 1. Structure of ANN [2]

The neurons and input flow which are represented by circles and arrows respectively make up the body of the ANN. The neurons receive inputs x_1, x_2, … , x_m and multiplies with the weight connectors w_1, w_2, … , w_m. Next, the collection of weights results in a weighted sum which is added to a predetermined bias. The activation function is then applied to the weighted sum and an output value is produced to another connected neuron. At the end of one iteration, it compares its approximation with the actual output and learns by adjusting the weights to minimize the error function. Through this practice, ANN becomes reliable in evaluating unseen data, proving to be a strong contender against traditional numerical methods in solving FDEs.

This research aims to develop a feedforward neural network (FNN) in MATLAB software that utilizes Broyden-Fletcher-Goldfarb-Shanno Optimization method to solve Caputo FDEs. The neural network initializes weight parameters, goes through backpropagation and applies BFGS optimization method to minimize the error function. This FNN model is tested on various examples. Iterations are repeated until desired accuracy is achieved and the findings are compared with other traditional methods to determine accuracy and computation cost of the proposed FNN model.

1.2 Literature Review

1.2.1 History of differential equations

Differential equations (DEs) are equations which relate unknown functions and their derivatives. Since the mid 17th century, DEs have been the building blocks of pure and applied mathematics, even to this day. Further exploration of these equations has led to the discovery of partial differential equations (PDEs) in the 18th century and general and particular solutions as well as existence theorems in the 19th century which contributed to the invention of the general theory of relativity by Einstein in the 20th century. As a result, many applications in mechanics, optics and quantum mathematics have appeared to name a few [3]. Since then, many solving techniques have been

created either through exact or approximation methods. There was even an effort to compile code for solving different DEs to be used by softwares such as MATLAB, Python and R [4]. Recently, DEs have been successfully solved using quantum computers, efficiently simulating classical dynamical systems with very high dimensions [5].

1.2.2 History of fractional calculus

Fractional calculus (FC) is an extension of classical calculus (CC) as it focuses on arbitrary orders of derivatives and integrals. When CC was developed back in 17th century Europe by Isaac Newton and Gottfried Wilhelm Leibniz, they only considered the integer orders for derivatives and integrals. This prompted French mathematician Marquis de L'Hopital to ponder about derivatives of half order and thus wrote a letter to Leibniz asking about it to which Leibniz responded in a letter on 30th September 1695 stating that it may contribute to valuable findings in the future [6]. This incident is referred to as the birth of FC.

Ever since, the development of FC gradually grew, starting from 1730 when Euler's curiosity on whether derivatives can be algebraically represented for non-integer orders led to the development of the gamma function. This provided the groundwork for other Mathematicians such as Laplace (1812) who successfully created a definition for fractional derivatives using integrals and Lacroix (1819) who managed to calculate the derivative of order ½ of function x using the formula as presented in Eq. (1) [7].

$$\frac{d^n y}{dx^n} = \frac{m!}{(m-1)!} x^{m-n} = \frac{\Gamma(m+1)}{\Gamma(m-n+1)} x^{m-n} \tag{1}$$

where m and n are powers of x and order of the derivative respectively.
Substituting $m = 1$ and $n = \frac{1}{2}$ into Eq. (1) obtains,

$$\frac{d^{\frac{1}{2}} y}{dx^{\frac{1}{2}}} = D_x^{1/2} x = \frac{\Gamma(2)}{\Gamma(\frac{3}{2})} x^{1/2} = 2\sqrt{\frac{x}{\pi}}, \tag{2}$$

End result of Eq. (2) is the half order derivative of the function $y = x$ discovered by Lacroix.

Three years later, Joseph Fourier (1822), indirectly hinted at fractional derivatives and fractional calculus being a possibility. He noticed that the power of the frequency parameter, $i\omega$ in Fourier transform of the complex plane could be any real number, leading him to believe that the order of differentiation could consist of non-integer values as well [8]. Through advance development of AI, implementing fractional derivatives (FDs) into computer systems has led to a massive contribution in applications such as edge detection, optical flow, image recognition and object detection to name a few [9]. There was even a study about overcoming uncertainties in control systems by improving optimization and control using FC which was conducted by Singh and Bingi [10]. It is no doubt that FC has many advantages over traditional integer-order models in robotics because of its ability to describe dynamic systems with high accuracy, providing significant advancements in areas such as motion planning, stability and adaptive control.

1.2.3 Artificial neural network in solving fractional differential equations

One of the earliest contributions of using ANN to solve FDEs was the development of a numerical simulation algorithm for fractional order dynamics model [11]. Besides that, a stochastic method for solving a nonlinear fractional-order Riccati differential equation has been developed [12]. In 2017, Pakdaman *et al.*, [13] approximated FDE solutions using ANN but this time using BFGS optimization method. After testing on linear and nonlinear FDEs, results show that ANN with BFGS optimization method is superior to the predictor-corrector (PC) method and fractional Euler (FE) method in terms of accuracy and convergence speed. as the solution of the FDE returns an entire function which is more accurate instead of just selected points.

Curious about new training methods of the ANN, Sivalingam et al. presented a physics-informed neural network based on extreme learning to solve generalized Caputo FDEs [14]. The ANN in the paper made use of L-1 finite difference scheme and Volterra integral equation for the generation of the error function. Currently, the authors are extending their research into solving fractional PDEs with the same model. It is evident that solving FDEs using numerical systems are still limited compared to ODEs.

1.3 Motivation

When ANN is applied in solving problems, it is commonly used alongside optimization methods whether they are first order or second order. In recent findings, it has been proven that ANN with second order optimization methods obtains more accurate results as well as less iterations, resulting in faster convergence speeds compared to first order optimization methods. One of these second order optimization methods is the BFGS optimization method. It is also important to note that many traditional methods to solve ODEs have been modified to solve FDEs to suit the underlying theory in ODEs. However, there are few numerical solutions that involve FNN with second order optimization method being applied to solve not only ODEs but also FDEs. Thus, the idea of using FNN with BFGS to solve Caputo FDEs has not been explored in much detail yet. Existing numerical methods such as fractional Euler and fractional Runge-Kutta 4 method require many iterations and it can be time-consuming whereas FNN uses less iterations to obtain these approximations, only requiring a short computing time compared to other existing numerical methods. Moreover, the results obtained are discrete because they are only limited to the step-size being used unlike FNN which is able to produce a continuous function containing all values in between as well. However, the accuracy and effectiveness of FNN in solving Caputo FDEs compared to them is not investigated to its full extent. Hence, this study aims to develop an FNN with BFGS to perform numerical simulations to solve Caputo FDEs and make comparisons with other numerical methods based on accuracy and effectiveness.

2. Methodology

2.1 Mathematical Formulation

The general form of an FDE is as follows

$$({}_a^C D_x^\alpha y)\,(x) = f\big(x, y(x)\big),$$
$$a \le x \le b, \tag{3}$$

with initial conditions

$$y^{(l)}(0) = c_l, \qquad l = 0, \dots, m-1.$$

where $m-1 < \alpha \le m, m \in \mathbb{N} := \{1,2,\dots\},\ c_l \in \mathbb{R} : {}_a^C D_x^\alpha = (-\infty, \infty)$, is the Caputo-type derivative and $f: [0,b] \times \mathbb{R} \to \mathbb{R}$ is a given continuous function [13].

2.2 Trial Solution

To ensure the solution effectively satisfies initial condition, the approximate solution is defined. Mathematical formulation of approximation solution to Caputo FDE in Eq. (3) can be created as such:

$$y_T(x, \boldsymbol{w}) = a + xN(x, \boldsymbol{w}). \tag{4}$$

First term in Eq. (4) satisfies the initial condition while the other contains the FNN output, $N(x,w)$. Substituting Eq. (4) into original problem of Caputo FDE in Eq. (3) obtains

$$({}_a^C D_x^\alpha y_T)\,(x, \boldsymbol{w}) = f\big(x, y_T(x, \boldsymbol{w})\big). \tag{5}$$

Eq. (5) is the trial solution implemented in this study.

2.3 Error Function

Once the problem and approximation solution is defined, the error function is next. Before this can be done, discretization of domain $[0, x_f]$ is needed. It is discretized using ρ subintervals with constant step size of $h = \frac{x_f}{\rho}$, resulting in updated error function of

$$E(w) = \sum_{d=1}^{m_t} \left[({}_a^C D_x^\alpha y_T)\big(x^{(d)}, w\big) - f\left(x^{(d)}, y_T\big(x^{(d)}, w\big)\right)\right]^2. \tag{6}$$

where the goal is to minimize Eq. (6) to obtain optimal network parameters w.

2.4 FNN Structure for Solving FDE

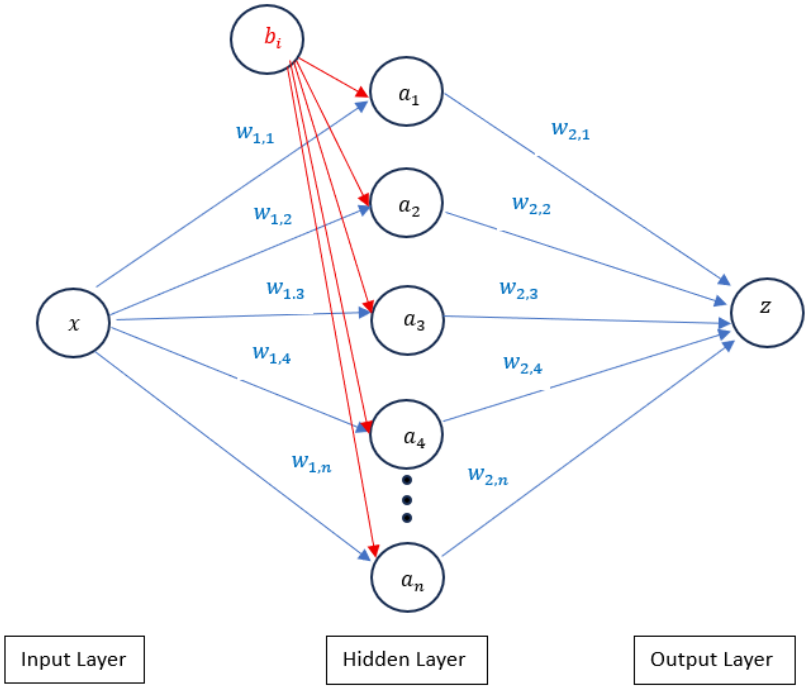

Fig. 2. Structure of proposed FNN

Figure 2 shows the structure of a feedforward neural network that is used in this study to solve Caputo FDEs. The neural network is divided into three parts, namely the input layer, hidden layer and output layer. Each layer is represented by circles also known as neurons which make up the entire neural network structure. In the input layer, x is the input value from a determined initial point that is received. This is followed by the multiplication of the first set of n weight values and addition of bias value in the hidden layer to form:

$$a_i = w_{1,i}x + b_i, \tag{7}$$

where $i = 0,1, \dots, n$

After that, a_i from Eq. (7) multiplies by the second set of n weight values to form :

$$z = \sum_{i=1}^{n} w_{2,i}a_i. \tag{8}$$

where z in Eq. (8) is the output value produced at the end of the FNN structure.

Going through the entire network once and obtaining an initial approximation is considered as one iteration. The trained model goes through many iterations attempting to minimize the error function until the parameters (weights and biases) have little to no change with increasing number of iterations. At this point, the parameters are optimal and the model has approximated a solution with high accuracy to the given problem.

2.5 Broyden-Fletcher-Goldfarb-Shanno Optimization Method

When dealing with unconstrained optimization problems, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method proves to be very effective. BFGS optimization method is effective in solving FDEs by reducing the error function to a minimum as defined in Eq. (6) as

$$\min_{w} E(w) \tag{9}$$

MATLAB software can be used to create an optimization solver known as $\mathrm{fminunc}$. Here, the proposed FNN will make use of BFGS method to learn the network parameters helped by the built-in solver in MATLAB to solve Caputo FDE. In this neural network, all mentioned network parameters, w are involved in the learning process. The basic command of this solver is $\mathrm{fminunc}(\mathrm{fun}, x_0, \mathrm{options})$, where fun is the specific objective function or error function as represented in Eq. (9) that needs to be minimized. A vector or an array is accepted and a scalar vector is returned, while x_0 is the initial point as a vector or array. When solving Caputo FDE, the input for fun is the error function and the initial values of network parameters are randomly taken from uniform random distribution denoted by x_0. The biases are initialized to zero.

The remaining input argument in $\mathrm{fminunc}$ that needs to be fulfilled is options. This input argument specifies output of other syntax known as $\mathrm{optimset}$. Summarized in Table 1 are the many arguments of $\mathrm{optimset}$, most with the modified settings for the FNN-BFGS on solving Caputo FDE while others are set to default.

Table 1
Modified settings for the argument in $optimset$ used in FNN-BFGS scheme

Argument	Description	Settings
Algorithm	Type of algorithm used to minimize objective function	'quasi-newton'
HessianApproximation	Indicates how Hessian is calculated	'bfgs'
StepTolerance	Stopping criteria of the solver	'1×10^{-8}'
MaxIterations	Maximum number of iterations allowed	'1000'
MaxFunctionEvaluations	Maximum number of functions evaluations allowed	'1×10^{7}'
OutputFcn	To generate graphical output or record the generated algorithm data history	'OutputFcn'

3. Numerical Results

3.1 Problem 1 (Composite Fractional Oscillation Equation, [15])

$$D^{\alpha} y(x) + y(x) = f(x), \quad 0 < \alpha < 1, \;\; y(0) = 0, \tag{10}$$

where

$$f(x) = x^2 + \frac{2x^{2-\alpha}}{\Gamma(3-\alpha)}.$$

The exact solution of the problem as represented in Eq. (10) is $y(x) = x^2$.

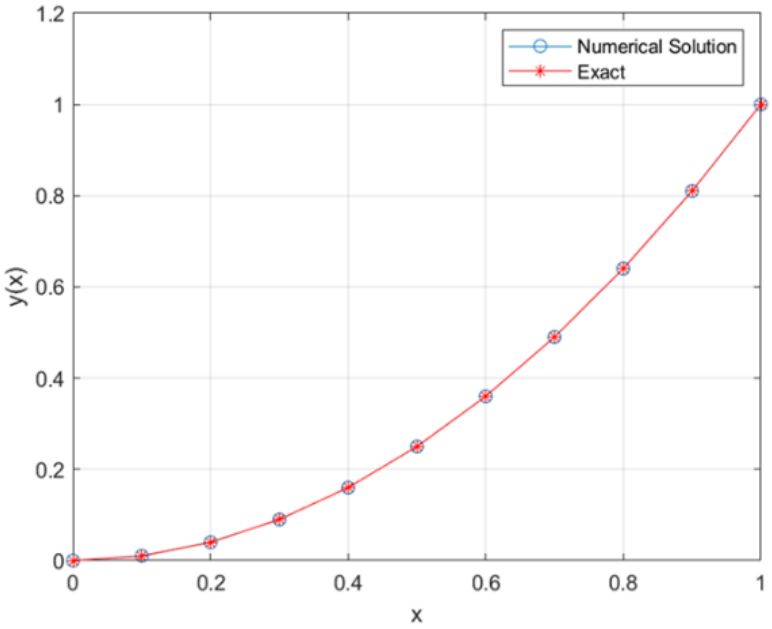

Fig. 3. Comparison of numerical solution with exact solution $y(x)$ against x for Problem 1 with α = 0.75

Table 2
Comparison of approximation of FNN model with Adomian Decomposition method (ADM) and Spectral Collocation method (SCM) and their numerical errors for Problem 1 with α = 0.75

x	Exact	Approximation			Numerical Error		
		FNN model	ADM	SCM	FNN model	ADM	SCM
0	0.000000	0	0	0	0.000000	0.000000	0.000000
0.1	0.010000	0.010398	0.216866	0.2321153	0.000398	0.206866	0.222115
0.2	0.040000	0.040307	0.428892	0.4961556	0.000307	0.388892	0.456156
0.3	0.090000	0.090091	0.654614	0.7523005	0.000091	0.564614	0.662301
0.4	0.160000	0.159925	0.891404	0.9998683	-0.000075	0.731404	0.839868
0.5	0.250000	0.249867	1.132763	1.2372036	-0.000133	0.882763	0.987204
0.6	0.360000	0.359914	1.370240	1.4604023	-0.000086	1.010240	1.100402
0.7	0.490000	0.490037	1.594278	1.6619744	0.000037	1.104278	1.171974
0.8	0.640000	0.640181	1.794879	1.8278045	0.000181	1.154879	1.187805
0.9	0.810000	0.810255	1.962239	1.9347648	0.000255	1.152239	1.124765

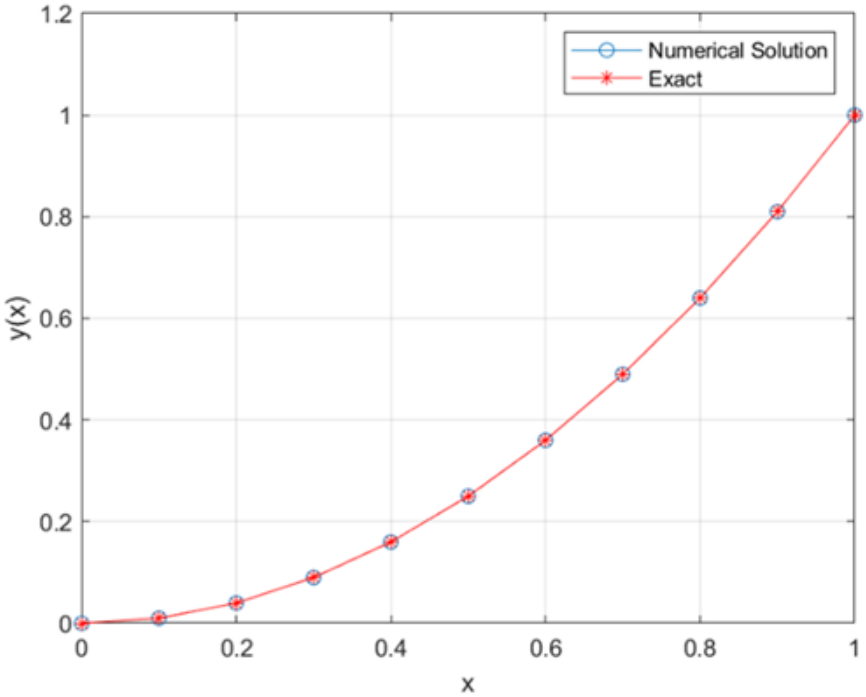

Fig. 4. Comparison of numerical solution with exact solution $y(x)$ against x for Problem 1 with α = 0.9

Table 3
Comparison of numerical error in approximation by FNN model for Problem 1 using α = 0.75 and α = 0.9

Approximation		Numerical Error	
α = 0.75	**α = 0.9**	**α = 0.75**	**α = 0.9**
0	0	0	0
0.010398	0.010045	3.98×10^{-4}	4.52×10^{-5}
0.040307	0.039993	3.07×10^{-4}	6.93×10^{-6}
0.090091	0.089981	9.10×10^{-5}	1.88×10^{-5}
0.159925	0.160012	7.49×10^{-5}	1.16×10^{-5}
0.249867	0.250041	1.33×10^{-4}	4.10×10^{-5}
0.359914	0.360040	8.57×10^{-5}	3.99×10^{-5}
0.490037	0.490013	3.69×10^{-5}	1.34×10^{-5}
0.640181	0.639991	1.81×10^{-4}	8.80×10^{-6}
0.810255	0.809997	2.55×10^{-4}	2.90×10^{-6}
1.000107	1.000013	1.07×10^{-4}	1.28×10^{-5}

The alpha value for Problem 1 is $\alpha = 0.75$. Figure 3 shows that the approximation solution by the proposed FNN model visually fits the exact solution nicely. Comparing with ADM and SCM for $\alpha = 0.75$ as in Table 2, the FNN model's approximation has the least numerical error out of the three. It is obvious that the FNN model is highly accurate and preferred over the other two. In addition to that, the computational time taken for FNN model is 3.27s, proving it to be quite efficient.

The alpha value for Problem 1 is $\alpha = 0.9$. Figure 4 shows that the numerical solution approximated by the proposed model closely fits the exact solution. Based on Table 3, numerical error of FNN model for Problem 1 with $\alpha = 0.9$ is smaller when compared with that of $\alpha = 0.75$. Therefore, it can be concluded from here that a higher α value leads to a more accurate approximation.

$$D^2 y(x) + ({}_0^C D_x^{1.5} y)(x) + y(x) = 1 + x, \quad x \in [0,1] \tag{11}$$

3.2 Problem 2 (Inhomogeneous Bagley-Torvik equation, [16])

with initial conditions $y(0) = 1, y'(0) = 1$.

The exact solution of Problem 2 as represented in Eq. (11) is $y(x) = 1 + x$.

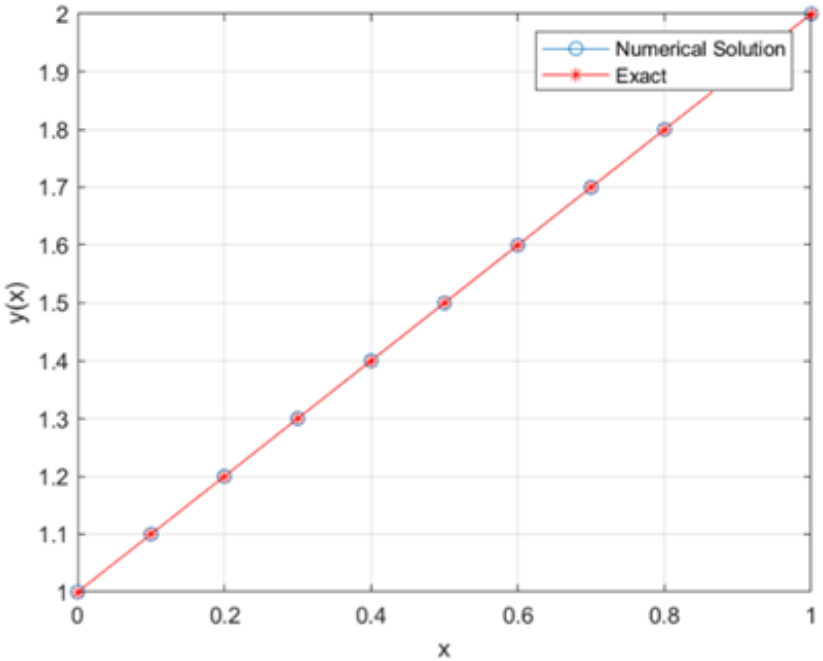

Fig. 5. Comparison of numerical solution with exact solution $y(x)$ against x for Problem 2 with α = 1.5

Table 4
Approximations by the FNN model and the comparison of its numerical error with that of Genetic Algorithm-Pattern Search Technique (GA-PS) for Problem 2 with α = 1.5

Exact	Error	
	Approximation	GA-PS [15]
1.00	0.0000	1.60×10^{-2}
1.10	1.02×10^{-5}	4.73×10^{-3}
1.20	1.01×10^{-5}	1.95×10^{-4}
1.30	6.68×10^{-5}	6.66×10^{-4}
1.40	1.23×10^{-4}	1.62×10^{-3}
1.50	1.48×10^{-4}	4.97×10^{-3}
1.60	1.36×10^{-4}	7.42×10^{-3}
1.70	1.09×10^{-4}	6.70×10^{-3}
1.80	9.85×10^{-5}	1.27×10^{-5}
1.90	1.24×10^{-4}	1.62×10^{-2}
2.00	1.68×10^{-4}	4.62×10^{-2}

The alpha value for Problem 2 is $\alpha = 1.5$. From Figure 5, it is clear that the numerical solution by FNN suitably fits the exact solution. Not only that, Table 4 shows the numerical error by the FNN

model is less than that of GA-PS. Furthermore, computation time used by FNN is 0.45 seconds, shorter than GA-PS which needed 2297 seconds.

3.3 Problem 3 (Composite Fractional Oscillation equation immersed in Newtonian fluid,[17])

$$D^{\alpha}y(x) + y(x) = x^4 - \frac{1}{2}x^3 - \frac{3}{\Gamma(4-\alpha)}x^{3-\alpha} + \frac{24}{\Gamma(5-\alpha)}x^{4-\alpha}, x \in [0,1] \quad (12)$$

where $y(0) = 0$ and $0 < \alpha < 1$.
The exact solution for Problem 3 as represented in Eq. (12) is $y(x) = x^4 - \frac{1}{2}x^3$. The results are shown below.

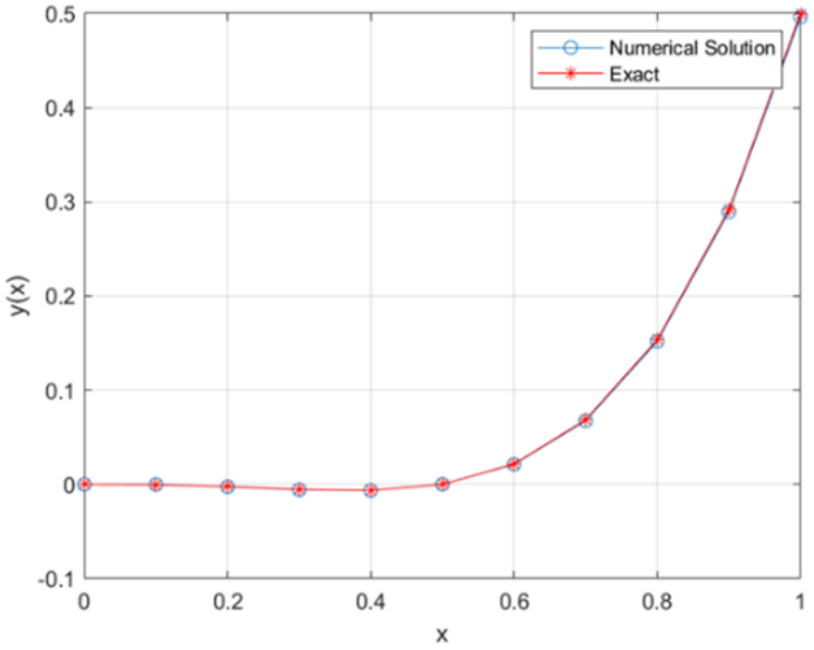

Fig. 6. Comparison of numerical solution with exact solution $y(x)$ against x for Problem 3 with $\alpha = 0.25$

Table 5
Approximations by the FNN model and its numerical error for Problem 3 with α = 0.25

x	Exact	Approximation	Numerical Error
0.000000	0.000000	0.000000	0.0000
0.100000	–0.000400	–0.000160	2.40×10^{-4}
0.200000	–0.002400	–0.002517	1.17×10^{-4}
0.300000	–0.005400	–0.005594	1.94×10^{-4}
0.400000	–0.006400	–0.006411	1.14×10^{-5}
0.500000	0.000000	0.000041	4.07×10^{-5}
0.600000	0.021600	0.021268	3.32×10^{-4}
0.700000	0.068600	0.067548	1.05×10^{-3}
0.800000	0.153600	0.151868	1.73×10^{-3}
0.900000	0.291600	0.289386	2.21×10^{-3}
1.000000	0.500000	0.496356	3.64×10^{-3}

The alpha value for Problem 3 is $\alpha = 0.25$. In Figure 6, it is clear that the numerical solution approximated by the proposed model almost perfectly fits the exact solution. Besides that, Table 5

shows the small numerical error obtained. The FNN model is also efficient in reaching the solution quickly as its computation time is 6.6457 seconds.

3.4 Problem 4 (Tumor System Without Immune Response,[18])

Consider a system of fractional differential equations:

$$D^{\alpha}T_I = 2a_4T_M - (d_2 + a_1)T_I,$$
$$D^{\alpha}T_M = a_1T_I - dT_M. \tag{13}$$

where $\alpha_4 = 0.8,\ \alpha_1 = 1,\ d = 1.9,\ d_2 = 0.11$ and initial conditions used in Eq. (13) are $T_I(0) = 1.3,\ T_M(0) = 1.2$.

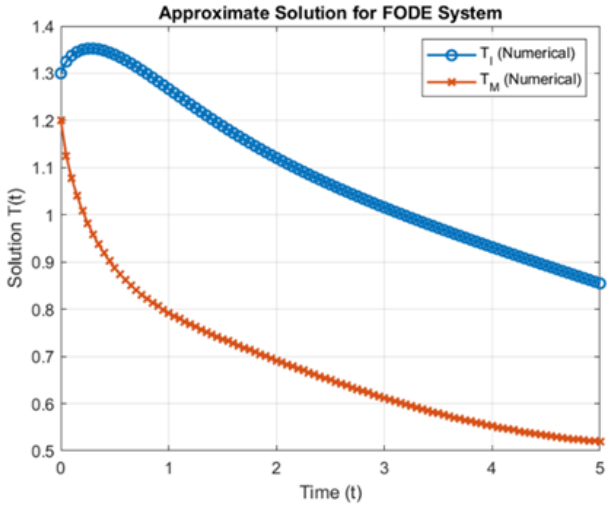

Fig. 7. Graph of numerical solution of T_I and T_M against time for Problem 4 with $\alpha = 0.75$

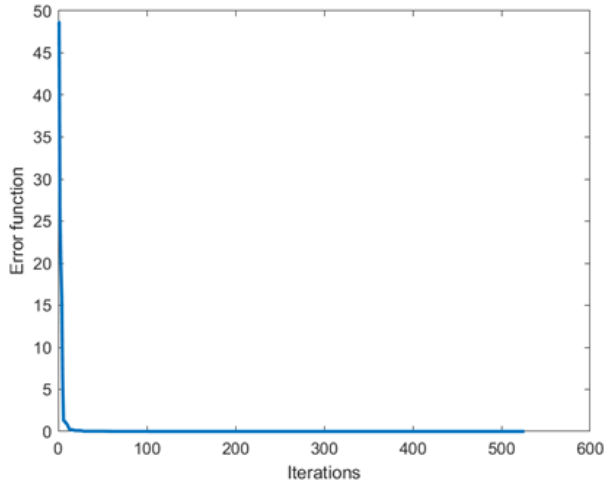

Fig. 8. Graph of error function against number of iterations for Problem 4 with $\alpha = 0.75$

The alpha value for Problem 4 is $\alpha = 0.75$. From Figure 7, the FNN model is able to approximate good solutions to the FDE system. To support this result, Figure 8 illustrates that the error function is

close to 0 when number of iterations increases. This reflects the potential of the FNN to solve complex problems as it can even handle the FDE tumor system.

4. Conclusions

This study has developed an efficient FNN model with BFGS optimization method in MATLAB to solve Caputo Fractional Differential Equations. The proposed FNN model with BFGS obtains high accuracy approximations to the solutions for all three problems of Caputo FDEs as well as the FDE tumor system. The proposed FNN model performed better in terms of accuracy and effectiveness when solving for Caputo FDEs compared to traditional numerical methods because it can overcome discretization errors in step-based methods by generating a continuous, differentiable solution over the entire domain. BFGS method has shown that it converges rapidly and provides stability not only for both linear and nonlinear problems but also for tumor systems. As a result, the proposed FNN model has demonstrated that it is a powerful tool to handle Caputo FDEs due to its high accuracy and short computation time. Experimenting with optimization methods other than BFGS, swapping out activation functions and implementing different definitions of fractional derivatives should be considered in future studies for higher accuracy.

Acknowledgement

This study was supported by the Geran Dana Penyelidik Berpotensi Dana DTD (Vot No. 4J724) awarded by Universiti Teknologi Malaysia.

References

[1] Shao, Feng, and Zheng Shen. "How can artificial neural networks approximate the brain?." *Frontiers in psychology* 13 (2023): 970214. https://doi.org/10.3389/fpsyg.2022.970214

[2] Papadomichelakis, G. *Machine Learning Techniques for the Estimation of the Operating Parameters of Solar Cells*. 2022. https://doi.org/10.13140/RG.2.2.28584.24326.

[3] Archibald, Tom, Craig Fraser, and Ivor Grattan-Guinness. "The history of differential equations, 1670–1950." *Oberwolfach reports* 1, no. 4 (2005): 2729-2794. https://doi.org/10.14760/OWR-2004-51.

[4] Zwillinger, Daniel, and Vladimir Dobrushkin. *Handbook of differential equations*. Chapman and Hall/CRC, 2021.https://doi.org/10.1201/9780429286834.

[5] Jennings, David, Matteo Lostaglio, Robert B. Lowrie, Sam Pallister, and Andrew T. Sornborger. "The cost of solving linear differential equations on a quantum computer: fast-forwarding to explicit resource counts." *Quantum* 8 (2024): 1553. https://doi.org/10.22331/q-2024-12-05-1553.

[6] Loverro, Adam. "Fractional calculus: history, definitions and applications for the engineer." *Rapport technique, Univeristy of Notre Dame: Department of Aerospace and Mechanical Engineering* (2004): 1-28.

[7] Domingues, J. C. *Lacroix and the Calculus*. Vol. 35. Springer Science & Business Media, 2008. https://doi.org/10.1007/978-3-7643-8638-2.

[8] Kilbas, A. A., Yu F. Luchko, H. Martinez, and J. J. Trujillo. "Fractional Fourier transform in the framework of fractional calculus operators." *Integral Transforms and Special Functions* 21, no. 10 (2010): 779-795. https://doi.org/10.1080/10652461003676099.

[9] Arora, Sugandha, Trilok Mathur, Shivi Agarwal, Kamlesh Tiwari, and Phalguni Gupta. "Applications of fractional calculus in computer vision: a survey." *Neurocomputing* 489 (2022): 407-428. https://doi.org/10.1016/j.neucom.2021.10.122.

[10] Singh, Abhaya Pal, and Kishore Bingi. "Applications of fractional-order calculus in robotics." *Fractal and Fractional* 8, no. 7 (2024): 403. https://doi.org/10.3390/fractalfract8070403.

[11] Zhou, Shangbo, Hua Li, and Zhengzhou Zhu. "Chaos control and synchronization in a fractional neuron network system." *Chaos, Solitons & Fractals* 36, no. 4 (2008): 973-984. https://doi.org/10.1016/j.chaos.2006.07.034.

[12] Raja, Muhammad Asif Zahoor, Junaid Ali Khan, and Ijaz Mansoor Qureshi. "A new stochastic approach for solution of Riccati differential equation of fractional order." *Annals of mathematics and artificial intelligence* 60, no. 3 (2010): 229-250. https://doi.org/10.1007/s10472-010-9214-4.

[13] Pakdaman, Morteza, Ali Ahmadian, Sohrab Effati, Soheil Salahshour, and Dumitru Baleanu. "Solving differential equations of fractional order using an optimization technique based on training artificial neural network." *Applied mathematics and computation* 293 (2017): 81-95. https://doi.org/10.1016/j.amc.2016.07.021.

[14] Sivalingam, S. M., Pushpendra Kumar, and Venkatesan Govindaraj. "A neural networks-based numerical method for the generalized Caputo-type fractional differential equations." *Mathematics and Computers in Simulation* 213 (2023): 302-323. https://doi.org/10.1016/j.matcom.2023.06.012.

[15] Shah, Firdous A., R. Abass, and Lokenath Debnath. "Numerical solution of fractional differential equations using Haar wavelet operational matrix method." *International Journal of Applied and Computational Mathematics* 3, no. 3 (2017): 2423-2445. https://doi.org/10.1007/s40819-016-0245-y.

[16] Admon, Mohd Rashid, Norazak Senu, Ali Ahmadian, Zanariah Abdul Majid, and Soheil Salahshour. "A new efficient algorithm based on feedforward neural network for solving differential equations of fractional order." *Communications in nonlinear science and numerical simulation* 117 (2023): 106968. https://doi.org/10.1016/j.cnsns.2022.106968.

[17] Talaei, Younes, and M. J. N. C. Asgari. "An operational matrix based on Chelyshkov polynomials for solving multi-order fractional differential equations." *Neural Computing and Applications* 30, no. 5 (2018): 1369-1376. https://doi.org/10.1007/s00521-017-3118-1.

[18] Admon, Mohd Rashid, and Normah Maan. "Modelling tumor growth with immune response and drug using ordinary differential equations." *Jurnal Teknologi (Sciences & Engineering)* 79, no. 5 (2017). https://doi.org/10.11113/jt.v79.10303.