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 with BFGS Optimization for Solving Ordinary Differential Equations with Application to Tumor-Immune Dynamics

Wan Muhammad Shahmi Wan Muhammad Shahrizal[1], Mohd Rashid Admon[1,*], Noorehan Yaacob[1], Ali Ahmadian[2], Mohd Ariff Admon[1], Mohamad Shahiir Saidin[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:
Ordinary Differential Equations; Feedforward Neural Network; BFGS optimization; tumor-immune-drug interaction

ABSTRACT

Ordinary Differential Equations (ODEs) are fundamental for modeling dynamic systems in science and engineering. Traditional numerical methods, such as Euler and Runge-Kutta, are widely used but may face limitations in accuracy and computational efficiency, particularly for nonlinear problems. Recently, one of the branches in Neural Network known as feedforward neural network (FNN) have emerged as an alternative approach for solving differential equations due to their universal approximation capability. This study proposes a framework for solving ODEs by integrating a FNN with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization algorithm. The method constructs a trial solution that automatically satisfies the initial conditions and trains the network by minimizing a loss function defined by the residual of the differential equation. The proposed method is tested on several linear and nonlinear ODE problems and is further applied to a tumor-immune-drug interaction model to illustrate its capability in handling biologically motivated dynamical systems. Numerical results demonstrate that the ANN-BFGS approach achieves higher accuracy than Euler and fourth-order Runge-Kutta (RK4) methods while providing a continuous and differentiable solution across the domain.

1. Introduction

Ordinary differential equations (ODEs) serve as the primary mathematical framework for modelling dynamic systems in fields ranging from physics and engineering to biology and economics. Accurately solving these equations is critical for understanding system behavior and predicting future states. While analytical solutions exist for specific classes of ODEs, many real-world problems involve complex, nonlinear relationships that require numerical approximation.

Traditional numerical methods, such as Euler's method and the Runge-Kutta, have long been the standard for solving ODEs. The fourth-order Runge-Kutta (RK4) method, in particular, balances accuracy and computational efficiency and is widely used in engineering applications. However, these

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

https://doi.org/10.37934/araset.61.5.5565

step-based methods produce solutions only at discrete points and can suffer from accumulating discretization errors. Furthermore, for stiff or highly nonlinear problems, explicit methods may become unstable or require prohibitively small step sizes.

In recent years, Artificial Neural Networks (ANNs) have gained attention as a mesh-free, continuous alternative for solving differential equations. Leveraging the universal approximation theorem, ANNs can theoretically approximate any smooth function and its derivatives to arbitrary accuracy. Early work by Meade and Fernandez [1] demonstrated that neural networks could reduce errors through backpropagation while handling nonlinear behaviors effectively.

This paper presents a numerical framework that combines a feedforward ANN with the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm. Unlike standard gradient descent methods, BFGS is a quasi-Newton method that approximates the Hessian matrix, utilizing information about the curvature of the loss function to achieve faster and more stable convergence. This study aims to develop a neural network-based framework utilizing the BFGS optimization algorithm to solve ordinary differential equations and evaluate its accuracy and effectiveness through comparisons with exact solutions and traditional methods.

2. Methodology

2.1 Mathematical Formulation

Consider a general first-order ODE:

$$\frac{dy}{dx} = f(x, y), \qquad x \in [a, b]. \tag{1}$$

subject to the initial condition $y(x_0)$=y_0.

2.2 Trial Solution Construction

To ensure the solution effectively satisfies the initial condition, I construct a trial solution $y_t(x, \boldsymbol{p})$:

$$y_t(x, \boldsymbol{p}) = y_0 + (x - x_0)N(x, \boldsymbol{p}), \tag{2}$$

where:

- y_0 is the initial value.
- $(x - x_0)$ acts as a weighting function that vanishes at x_0, forcing $y_t(x_0) = y_0$.
- $N(x, \boldsymbol{p})$ is the neural network output with parameters $\boldsymbol{p}$ (weights and biases).

2.3 Loss Function

The network is trained by minimizing the squared residual of the ODE at n discrete collocation points x_i:

$$E(\boldsymbol{p}) = \frac{1}{n}\sum_{i=1}^{n} e(x_i, \boldsymbol{p})^2 , \tag{3}$$

where

$$e(x_i, \boldsymbol{p}) = \frac{dy_t(x_i, \boldsymbol{p})}{dx} - f\big(x_i, y_t(x_i, \boldsymbol{p})\big). \tag{4}$$

2.4 BFGS Optimization Implementation

The minimization of $E(p)$ is performed using the BFGS algorithm. The algorithm iteratively updates the network weights by approximating the inverse Hessian matrix (H_k):

$$p_{k+1} = p_k - \alpha_k H_k \nabla E(p_k). \tag{5}$$

This approach allows the network to escape local minima more effectively than standard backpropagation, ensuring a high-precision approximation of the ODE solution.

The BFGS method is an effective unconstrained solving method of optimization. BFGS is a quasi-Newton second order optimisation technique that takes advantage of data provided by the Hessian matrix to provide optimal results at a faster rate than other first-order optimization methods. Matlab software contains an optimization solver called fminunc. substituted the formula which was applied manually. Initialization of the network parameters is made using random numbers of uniform distribution of w_i and v_i. Then the biases of the network are made to zero. The other input in fminunc that has to be satisfied is options. This input determined the output of other approach known as optimset. This There are a lot of arguments of optimset and only a few settings of this approach are altered in ANN when solving ODEs using BFGS. This may be summed up in Table 1.

Table 1
Settings for argument in optimset used in ANN with BFGS method

Argument	Description	Setting
Display	Monitoring objective function value and iteration count at each iteration.	'iter'
HessUpdate	Specifies how Hessian is calculated.	'bfgs'
MaxIter	Maximum number of iterations allowed.	10000
MaxFunEval	Maximum number of functions evaluations allowed.	1000000
OutputFcn	Produce graph output and record the data history of generated algorithm.	@outputFcn_global

2.5 Algorithm for ANN with BFGS Method

The general process of solving ODEs using ANN with BFGS is summarized in Algorithm 1.

Algorithm 1 Solving ODE using ANN with BFGS Method

Input: Differential equation, $f(x, y)$; Initial condition, $y(a)$; Domain, [a, b]; Number of hidden layers, ℓ; optimset: 'iter', 'bfgs', 10000, 1000000, @outputFcn_global.

Output: Approximate solution $y(x)$ along [a, b]

1. Select collocation points x_i, uniform distributed in the interval [a, b].
2. Initialize weights and biases randomly to form parameter.
$$\boldsymbol{P} = \{w_i, b_i, v_i | \ i = 0,1,2, \dots\}. \tag{6}$$
3. Define the trial solution as $y_t(x, \boldsymbol{p}) = y_0 + (x - a).N(x, \boldsymbol{p})$ where $N(x, \boldsymbol{p})$ is the ANN output.
4. Compute the ANN output
$$N(x, \boldsymbol{p}) = \left(\sum_{i=1}^{n} v_i f(w_i x_i + b_1) \right). \tag{7}$$

5. Compute the derivative of the ANN output and trial solution

$$\frac{dN(x_i,\boldsymbol{p})}{dx} = \left(\sum_{i=1}^{n} v_i f'(w_i x_i + b_1) w_i\right), \tag{8}$$

$$\frac{dy_t(x_i,\boldsymbol{p})}{dx} = N(x_i,\boldsymbol{p}) + (x_i - a)\frac{dN(x_i,\boldsymbol{p})}{dx}. \tag{9}$$

6. Compute residual error at each point

$$e(x,\boldsymbol{p}) = \frac{dy_t(x,\boldsymbol{p})}{dx} - f\big(x, y_t(x,\boldsymbol{p})\big). \tag{10}$$

7. Compute loss function

$$E(\boldsymbol{p}) = \sum_{i=1}^{n} e(x_i,\boldsymbol{p})^2 . \tag{11}$$

8. Initialization of settings through optimset function for the execution of the BFGS through fminunc.

9. Execution of the BFGS through fminunc.

10. Generates results from the @outputFcn_global in optimset: Initial and final optimal network parameters and output:

3. Results

The ANN-BFGS framework was benchmarked on Matlab against exact analytical solutions, Euler's method, and the RK4 method across linear and nonlinear problems. Error is the difference between the exact solution and the numerical solution.

3.1 Problem 1 [8]

$$\frac{dy}{dx} = y - x, \qquad x \in [0,1]. \tag{12}$$

subject to $y(0) = 1$ and the exact solution is

$$y(x) = x + 1 - 0.5e^{x}. \tag{13}$$

The trial solution is

$$y_t(x,p) = 1 + x\big(N(x,\boldsymbol{p})\big). \tag{14}$$

Table 2
Numerical solution and absolute error obtained by ANN with BFGS for Example 1

x	Exact	ANN	Error ANN for Problem 1	Euler Method	Error
0.000000	0.500000	0.500000	$0.00\text{x}10^{-5}$	0.500000	0.000000
0.100000	0.547415	0.547415	$6.56\text{x}10^{-5}$	0.525000	0.000615
0.200000	0.589299	0.589290	$9.00\text{x}10^{-5}$	0.548750	0.001169
0.300000	0.625071	0.625052	$1.86\text{x}10^{-5}$	0.571313	0.001667
0.400000	0.654088	0.654065	$2.24\text{x}10^{-5}$	0.592747	0.002112
0.500000	0.675639	0.675620	$1.91\text{x}10^{-5}$	0.613110	0.002510
0.600000	0.688941	0.688929	$1.17\text{x}10^{-5}$	0.632454	0.002863
0.700000	0.693124	0.693118	$6.06\text{x}10^{-6}$	0.650831	0.003175
0.800000	0.687230	0.687221	$8.36\text{x}10^{-6}$	0.668290	0.003450
0.900000	0.670198	0.670179	$1.96\text{x}10^{-5}$	0.684875	0.003689
1.000000	0.640859	0.640832	$2.71\text{x}10^{-5}$	0.700632	0.003897

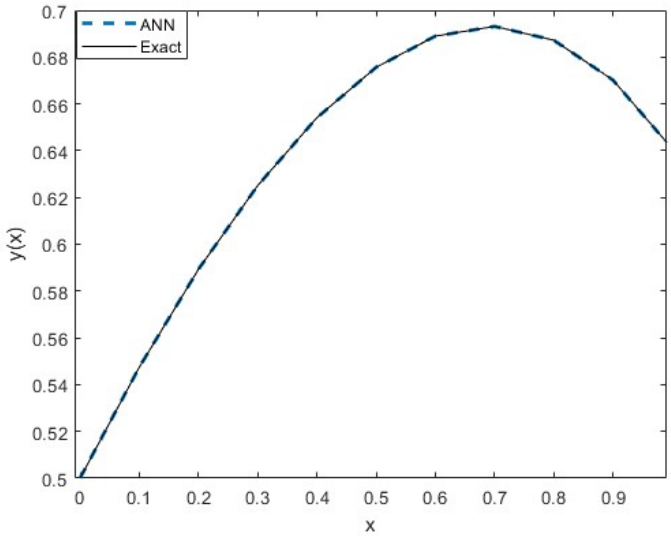

Fig. 1. Comparison numerical solutions between ANN with BFGS using exact solutions for Problem 1

In Problem 1, the linear ordinary differential equation was solved using the ANN-BFGS framework. The results were compared against Euler's method to evaluate relative performance. The ANN-BFGS method demonstrated superior accuracy across the entire domain. As shown in Table 2, absolute errors for the ANN solutions. In contrast, Euler's method produced significantly larger errors. The maximum error for Euler's method at $x = 1.0$ was approximately 0.062270, while the ANN-BFGS with tanh achieved an error of only 2.71×10^{-5}.

Figure 1. displays the comparison between the numerical solutions produced by the ANN with BFGS and the exact analytical solution. The visual alignment in Figure 1 demonstrates that the ANN-BFGS framework accurately tracks the exact solution across the domain.

The comparison clearly demonstrates that ANN-BFGS provides a substantial improvement in accuracy over Euler's method for this linear ODE, achieving error reduction of approximately three orders of magnitude while offering a continuous solution representation.

3.2 Problem 2 [14]

$$\frac{dy}{dx} = xy^2 - xy, \qquad x \in [0,1]. \tag{15}$$

subject to $y(0) = 0.5$ and the exact solution is

$$y(x) = \frac{1}{1 + e^{0.5x^2}} \ . \tag{16}$$

The trial solution is

$$y_t(x,p) = 1 + x\big(N(x,\boldsymbol{p})\big). \tag{17}$$

Table 3
Numerical nonlinear solution and absolute error obtained by ANN with BFGS for Problem 2

x	Exact	ANN	Error ANN for Problem 2	Euler Method	Error
0.000000	0.500000	0.500000	0.00	0.500000	0.000000
0.100000	0.498750	0.498753	$3.40\text{x}10^{-6}$	0.550000	0.002585
0.200000	0.495000	0.495033	$3.27\text{x}10^{-5}$	0.595000	0.005701
0.300000	0.488752	0.488804	$5.16\text{x}10^{-5}$	0.634500	0.009430
0.400000	0.480011	0.480057	$4.63\text{x}10^{-5}$	0.667950	0.013862
0.500000	0.468791	0.468812	$2.18\text{x}10^{-5}$	0.694745	0.019106
0.600000	0.455121	0.455116	$5.29\text{x}10^{-5}$	0.714219	0.025278
0.700000	0.439055	0.439039	$1.56\text{x}10^{-5}$	0.725641	0.032518
0.800000	0.420676	0.420677	$1.69\text{x}10^{-5}$	0.728205	0.040976
0.900000	0.400112	0.400148	$3.62\text{x}10^{-5}$	0.721026	0.050828
1.000000	0.377541	0.377584	$4.37\text{x}10^{-5}$	0.703129	0.062270

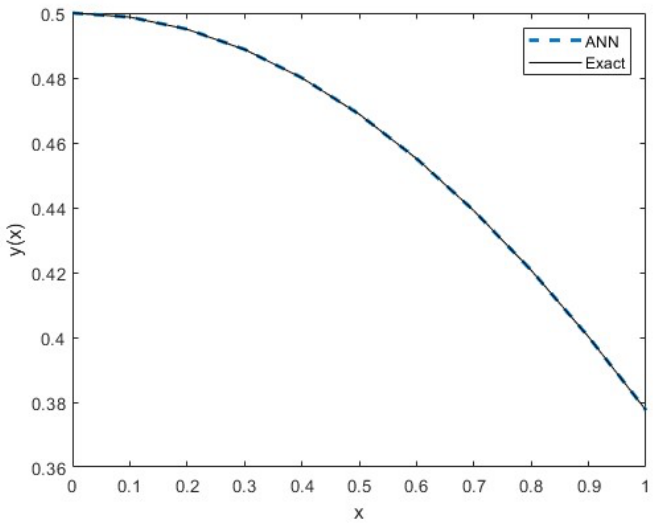

Fig. 2. Comparison numerical solutions between ANN with BFGS and exact solutions for Problem

Problem 2 presented the nonlinear ODE, ANN-BFGS method was compared against Euler's method to assess performance on nonlinear problems.

The ANN-BFGS approach again demonstrated significant accuracy advantages. As shown in Table 3, while Euler's method produced errors approximately 100 to 300 times larger. At $x = 1.0$, Euler's method achieved an error of 0.0114 compared to 4.37×10^{-5} for ANN-BFGS.

Figure 2 illustrates the comparison between the proposed ANN-BFGS method and the exact solution for the nonlinear case. The figure shows a high degree of overlap between the proposed method and the exact solution

This comparison confirms that ANN-BFGS provides superior accuracy for nonlinear ODEs, with error reduction of two to three orders of magnitude compared to the first-order Euler method.

3.3 Problem 3 [15]

$$\frac{dy}{dx} = \frac{1}{2}(1 - y), \qquad x \in [0,1]. \tag{18}$$

subject to $y(0) = 0.5$ and the exact solution is

$$y(x) = 1 - \frac{1}{2}e^{-0.5x}. \tag{19}$$

The trial solution is

$$y_t(x,p) = 1 + x\big(N(x,\boldsymbol{p})\big). \tag{20}$$

Table 3
Numerical nonlinear solution and absolute error obtained by ANN with BFGS for Problem 3

x	Exact	ANN	Error ANN for Problem 3	Euler Method	Error
0.000000	0.500000	0.500000	$0.00\text{x}10^{-5}$	0.500000	0.000000
0.100000	0.498750	0.498753	$3.40\text{x}10^{-6}$	0.525000	0.000615
0.200000	0.495000	0.495033	$3.27\text{x}10^{-5}$	0.548750	0.001169
0.300000	0.488752	0.488804	$5.16\text{x}10^{-5}$	0.571313	0.001667
0.400000	0.480011	0.480057	$4.63\text{x}10^{-5}$	0.592747	0.002112
0.500000	0.468791	0.468812	$2.18\text{x}10^{-5}$	0.613110	0.002510
0.600000	0.455121	0.455116	$5.29\text{x}10^{-5}$	0.632454	0.002863
0.700000	0.439055	0.439039	$1.56\text{x}10^{-5}$	0.650831	0.003175
0.800000	0.420676	0.420677	$1.69\text{x}10^{-6}$	0.668290	0.003450
0.900000	0.400112	0.400148	$3.62\text{x}10^{-5}$	0.684875	0.003689
1.000000	0.377541	0.377584	$4.37\text{x}10^{-5}$	0.700632	0.003897

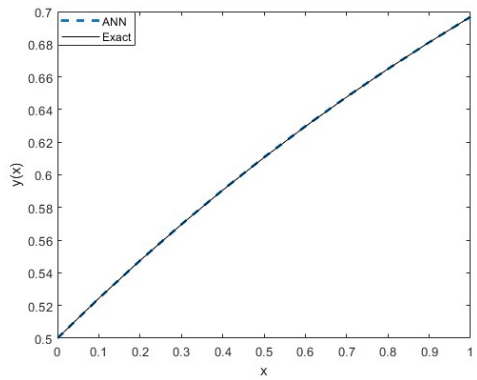

Fig. 3. Comparison numerical solutions between ANN with BFGS using exact solutions for Problem 3

Problem 3 presented the nonlinear ODE, ANN-BFGS method was compared against Euler's method to assess performance on nonlinear problems.

The results in Table 3 show that ANN-BFGS maintained its accuracy advantage, while Euler's method produced errors approximately 100 times larger. At $x = 1.0$, Euler's error was 0.003897 compared to 4.37×10^{-5} for ANN-BFGS.

Figure 3. provides a visual representation of the numerical solutions for ANN-BFGS versus the exact solution. The plot confirms that even for straightforward ODEs where Euler's method is stable, the ANN-BFGS method is preferable for applications requiring high precision.

This problem demonstrates that even for straightforward linear ODEs where Euler's method is stable, ANN-BFGS provides substantially improved accuracy, making it preferable for applications requiring high precision.

3.4 Problem 4 [16]

$$\frac{dT_I}{dt} = 2a_4 T_M - (c_1 I + d_2) T_I - a_1 T_I, \tag{21}$$

$$\frac{dT_M}{dt} = a_1 T_I - d_3 T_M - a_4 T_M I - c_3 T_M I, \tag{22}$$

$$\frac{dI}{dt} = k + \frac{\rho I (T_I + T_M)^n}{\beta + (T_I + T_M)^n} - c_2 I T_I - c_4 T_M I - d_1 I - k_3 (1 - e^{-k_4 u}) I, \tag{23}$$

$$\frac{du}{dt} = -\gamma u. \tag{24}$$

where $a_4 = 0.8$, $d_2 = 0.11$, $a_1 = 0.98$, $c_1 = c_3 = 0.9$, $d_3 = 0.4$,$k_3 = 0.49$, $\rho = 0.1$, $n = 3$, $\beta = 0.2$, $c_2 = c_4 = 0.085$, $d_1 = 0.29$, ɣ=0.85, $k_1 = 0.47, k_2 = 0.57, , k_4 = 0.061, , k = 0.029$

subject to

$$T_I(0) = 1.3,$$

$$T_M(0) = 1.2,$$

$$I(0) = 0.9,$$

$$u(0) = 0.1.$$

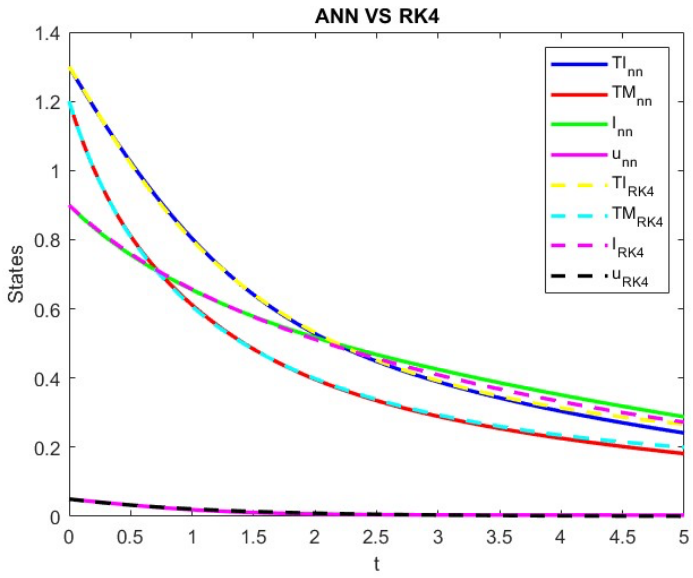

Fig. 4. Comparison between ANN-BFGS with RK4 for Problem 4

Problem 4 presented a challenging coupled system of four nonlinear ODEs modeling tumor-immune-drug interactions. Given the complexity and stiffness of this system, the ANN-BFGS solution was compared against the fourth-order Runge-Kutta (RK4) method as a more appropriate benchmark than Euler's method.

The ANN-BFGS framework successfully approximated the behavior of all four state variables, interphase tumor cells (T_I), metaphase tumor cells (T_M), immune response (I), and drug concentration (u) with solutions closely matching the RK4 benchmark. Both methods captured the essential dynamics: the initial dominance of T_I rapid decline of T_M under treatment, and saturating immune drug response.

While RK4 provided a reliable discrete reference solution with high accuracy, the trained ANN offered several distinct advantages. First, it provided a continuous, differentiable approximation across the entire time domain, enabling analytical operations without interpolation. Second, the neural network representation allows for efficient parameter studies and optimization without re-solving the ODE system from scratch.

Figure 4 compares the ANN-BFGS results against the fourth-order RK4 method for four variables T_I, T_M, I, and u. The figure demonstrates a close agreement between the ANN-BFGS and RK4 benchmarks, capturing complex dynamics such as the rapid decline of T_M under treatment.

The close agreement between ANN-BFGS and RK4, despite the system's nonlinearity and coupling, demonstrates the neural network's capability to approximate complex, multi-variable dynamics. This suggests that ANN-BFGS is not merely an alternative to basic methods like Euler's, but can compete with established high-order methods like RK4 for complex systems, while offering additional benefits of continuous representation and potential computational advantages for parametric analysis.

4. Conclusions

This research has successfully established a robust and efficient numerical framework for solving ordinary differential equations by integrating Artificial Neural Networks with the BFGS optimization algorithm. The study demonstrates that the proposed ANN-BFGS method not only addresses the limitations of traditional discrete solvers but also achieves a level of accuracy comparable to the high-

precision fourth-order Runge-Kutta method, while significantly outperforming Euler's method by reducing errors by several orders of magnitude. A key advantage of this framework is its ability to generate a continuous, differentiable solution over the entire domain, eliminating the discretization errors inherent in step-based methods and facilitating analytical operations without the need for interpolation. Furthermore, the application of the BFGS algorithm ensured rapid convergence and stability, proving effective for both linear and nonlinear problems, as well as complex coupled systems like tumor dynamics. These findings confirm that the ANN-BFGS approach is a promising, computationally efficient alternative for scientific and engineering applications, offering a flexible pathway for future extensions into partial differential equations and higher-order boundary value problems.

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

References

[1] Meade, R. J., & Fernandez, F. (2001). Neural networks for solving differential equations. *Journal of Computational Physics, 171*(1), 77-99.
[2] Paudel, P., et al. (2023). Comparative analysis of numerical methods for ODEs. *Journal of Applied Mathematics*.
[3] Lagaris, Isaac E., Aristidis Likas, and Dimitrios I. Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations." *IEEE transactions on neural networks* 9, no. 5 (1998): 987-1000. https://doi.org/10.1109/72.712178
[4] Okereke, R. N., et al. (2019). Solving first order ordinary differential equations using artificial neural networks. *Journal of Scientific Research*.
[5] Li, H., et al. (2020). Optimization of neural networks using BFGS algorithm. *SIAM Journal on Optimization*.
[6] Mahata, S., et al. (2021). Neural network based solution for first order differential equations. *Applied Soft Computing*.
[7] Carpenter, M. H., & Kennedy, C. A. (2005). Fourth-order Runge-Kutta methods for solving ODEs. *NASA Technical Reports*.
[8] Burden, R. L., & Faires, J. D. (2011). *Numerical Analysis* (9th ed.). Brooks/Cole. (Standard Reference for Linear ODE benchmarks).
[9] Shen, Xing, Xiaoliang Cheng, and Kewei Liang. "Deep Euler method: solving ODEs by approximating the local truncation error of the Euler method." *arXiv preprint arXiv:2003.09573* (2020).
[10] Xie, Yuchen, Richard H. Byrd, and Jorge Nocedal. "Analysis of the BFGS method with errors." *SIAM Journal on Optimization* 30, no. 1 (2020): 182-209. https://doi.org/10.1137/19M1240794
[11] Xie, Yuchen, Richard H. Byrd, and Jorge Nocedal. "Analysis of the BFGS method with errors." *SIAM Journal on Optimization* 30, no. 1 (2020): 182-209. https://doi.org/10.1137/19M1240794
[12] Bakar, A., Shamsuddin, N., & Aziz, M. (2019). Neural network-based solution of complex ODEs. *Journal of Engineering and Applied Sciences*
[13] Sharma, A., Rani, R., & Singh, P. (2020). Modeling drug dynamics using artificial neural networks for pharmacokinetics. *Computers in Biology and Medicine*
[14] Malek, Alaeddin, and R. Shekari Beidokhti. "Numerical solution for high order differential equations using a hybrid neural network—optimization method." *Applied mathematics and computation* 183, no. 1 (2006): 260-271. https://doi.org/10.1016/j.amc.2006.05.068
[15] Butcher, John Charles. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016. https://doi.org/10.1002/9781119121534
[16] 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.v79.9791
[17] Raissi, Maziar, Paris Perdikaris, and George E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *Journal of Computational physics* 378 (2019): 686-707. https://doi.org/10.1016/j.jcp.2018.10.045

[18] Diederik, Kingma. "Adam: A method for stochastic optimization." *(No Title)* (2014).