SEMARAK ILMU PUBLISHING

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

Artificial Neural Network Solution of Ordinary Differential Equations using BFGS Optimization: Activation Function Comparison

Siti Nurzahirah Mohd Ramli[1], Mohd Rashid Admon[1,*], Mohamad Shahiir Saidin[1], Ali Ahmadian[2], Noorehan Yaacob[1], Mohd Ariff Admon[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 31 May 2026
Accepted 30 June 2026
Available online 6 July 2026

Keywords:

Artifial neural network; ordinary differential equations; Runge-Kutta method; Euler's Method; Broyden-Fletcher-Goldfarb-Shanno

ABSTRACT

Ordinary Differential Equations (ODEs) are fundamental tools in science and engineering. Existing numerical methods such as Euler's method and Runge-Kutta require discretization and repeated computations. This study develops Artificial Neural Network (ANN) with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) optimization for solving ODEs. Second aim is to perform numerical simulations using ANN with BFGS using hyperbolic tangent function and sigmoid to solve ODEs and compares the efficiency and accuracy of ANN with existing numerical methods. the neural network created a trial solution to satisfy the initial conditions and is trained using the BFGS optimization that updates weights by approximating the inverse Hessian. After training, the ANN with BFGS solutions were compared with existing numerical results obtained from Euler's method and RK4 method. The findings show that the ANN with BFGS method give accuracy better than Euler and similar with RK4 methods. The hyperbolic tangent activation function generally shows better approximation accuracy than sigmoid.

1. Introduction

Through the years, numerical methods such as Euler's method and Runge- Kutta methods have become preferred methods for solving ODEs. According to Paudel and Batta [1] Euler's method approximates solutions in discrete stages, but RK methods such as the fourth-order variation, offer more accuracy through analyze many variables for every step. The rapid growth of modern applications such as advanced engineering systems and real-time simulation demands more adaptive solution methods.

Okereke *et al.,* [2] develop the ANN for first-order ODE improve the accuracy leading to a thoroughly guided method for the field of computational mathematics. According to Hatcher and Yu [3] the flow of an ANN is a continuous procedure when data is passed on, modified, and learned within the network. As mention by Li *et al.*, [4] the accuracy of these neural network models has been

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

https://doi.org/10.37934/araset.61.5.1123

greatly improved by the use of optimization techniques like the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method.

All neural network requires at least one activation function to produce accurate estimations. Sharma *et al.*, [5] highlight that the selection of an activation function may affect the training speed, convergence, and accuracy of a neural network. Different activation functions give advantages and disadvantages depending on the system.

The main objective of this research is to develop an ANN framework capable of solving ODEs efficiently and accurately. The results will be compared against existing numerical methods such as Euler's method and Runge Kutta methods. These comparisons include accuracy, iteration convergence, and behaviour when dealing with nonlinear and linear systems. Through this comparison, the study aims to demonstrate the potential of ANN as another alternative tool for differential equation solving.

1.1 Artificial Neural Network

ANN have become known as effective tools for solving ordinary differential equations (ODEs), providing an alternative to existing numerical methods. The Universal Approximation Theorem was first shown by Cybenko [6], who was one of the first to prove that every continuous function could be approximated by a feedforward neural network with a single hidden layer and a sigmoid activation function. Then, Lagaris *et al.,* [7] have been among the first to introduce ANN-based methods for dealing with both initial and boundary value problems through implementing the ODE conditions into the loss function, allowing the network to learn solutions.

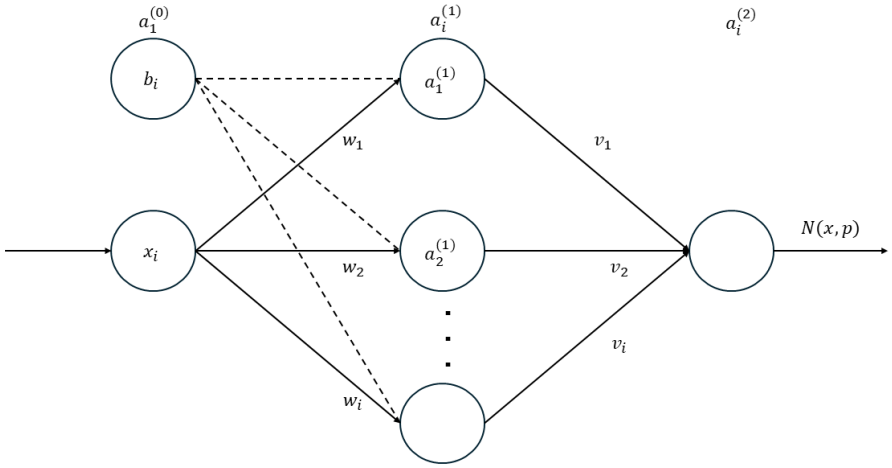

Fig. 1. Architecture of ANN

The network consists of an input layer one or more hidden layers and an output layer.

Layers: The layer number will be denoted by the upper script in the following ℓ. The input layer is $\ell = 0$ and the first hidden layer is $\ell = 1$. The ANN in this study consists of one hidden layer and five neurons. A single hidden layer is sufficient for approximating the solution of the given ordinary differential equations. The output layer is $\ell = 2$. The symbol $a_i^{(\ell)}$ represents the progress of input in every layer.

Weights: The system of weight from $a_1^{(0)}$ is denoted by w_i with $i = 1,2,3, \ldots$ The weight w_i connect the input and bias to the hidden layer. The system of weight from $a_i^{(1)}$ also denoted by v_i with $i = 1,2,3, \ldots$For v_i, it connects the hidden layer to the output layer $a_1^{(2)}$. Initial weights w_i and v_i from $a_1^{(0)}$ and $a_i^{(1)}$ are taken randomly. Next, a bias b_i provides a constant value unlike the input. All node in hidden layer is directly connected to the bias node.

1.2 Broyden-Fletcher-Goldfarb-Shanno (BFGS) Optimization Method

The BFGS algorithm, a quasi-Newton method, improves the training efficiency of ANN resulting in faster convergence than traditional gradient descent methods. Nocedal and Wright [8] mentioned that BFGS combines approximations of second-order information, which often results in faster convergence compared to traditional gradient descent, which mainly uses first-order derivative data. The algorithm starts with initial guess θ_0 and an initial approximation H_0 of the inverse Hessian matrix. Each iteration, the gradient $g_k = \nabla\mathcal{L}(\theta_k)$ is computed and search direction $p_k = -H_k G_k$ is determined. Next, choose the step size α_k to satisfy certain conditions and the parameter vector is updated

$$\theta_{k+1} = \theta_k + \alpha_k p_k. \tag{1}$$

Then, the algorithm calculated the step taken

$$s_k = \theta_{k+1} - \theta_k. \tag{2}$$

and the change in gradient

$$y_k = \nabla\mathcal{L}(\theta_{k+1}) - \nabla\mathcal{L}(\theta_k). \tag{3}$$

These are used to update the inverse Hession approximation using BFGS formula

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \left(\frac{s_k y_k^T}{y_k^T s_k} \right). \tag{4}$$

This will maintain that H_k remains symmetric and positive definite, provided $y_k^T s_k > 0$, which is normally implemented using a line search. The algorithm iteratively continues this procedure until convergence which is referred to the gradient become extremely small. This process can be carried out in MATLAB by using the fminunc function with the 'Quasi-Newton' algorithm option which applies BFGS automatically.

1.3 Activation Function

Sharma *et al.,* [9] discovered that the selection of the activation function can significantly affect the ability of the network to converge from training data. The activation functions that are used in the hidden layers are hyperbolic tangent (tanh) and sigmoid functions. These two functions were chosen because of their strong approximation capabilities. *Pratiwi et al.,* [10] studied the accuracy of the sigmoid activation function when choosing the most effective artificial neural network (ANN). The sigmoid function transfers input values into the range (0,1) remains relevant for its effectiveness.

Chen and Cao [11] studied the development and approximation of feedforward neural networks applying the hyperbolic tangent (tanh) function. The tanh activation function is preferred because of its zero-centered output which range from -1 to 1 allowing faster convergence during training.

1.4 Runge Kutta Fourth-Order (RK4)

Islam [12] found that the Runge-Kutta method performed better than the more basic Euler method in solving initial value problems (IVPs) in ODEs. The author worked for the fourth-order Runge-Kutta (RK4) method, that achieves high accuracy by evaluating intermediate slopes and approximating the solution effectively. The RK4 method approximate solution at each discrete point. The time interval $[t_0, T]$ is divided into N equal step size:

$$h = \frac{T - t_0}{N}. \tag{5}$$

Next, four intermediate slopes are calculated as follow:

$$k_1 = f(t_n\,, y_n), \tag{6}$$

$$k_2 = f\left(t_n + \frac{h}{2}\,, y_n + \frac{h}{2}k_1\right), \tag{7}$$

$$k_3 = f\left(t_n + \frac{h}{2}\,, y_n + \frac{h}{2}k_2\right), \tag{8}$$

$$k_4 = f(t_n + h\,, y_n + hk_3). \tag{9}$$

These slopes will be combined to obtain the updated value of the next solution for the next point.

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \tag{10}$$

By repeating this process for the entire domain, RK4 will produces stable and accurate approximation. The step size used for RK4 methods in this study is h = 0.1 for all examples.

1.5 Euler's Method

Salleh *et al.,* [13] studied applying of Euler's method to solve ODEs, that concentrate on how it is used in MATLAB. The authors used various examples to show how the method approximates results through each step and how it could possibly be effectively programmed. They ended up notice the limitations of Euler's method, especially its inaccuracy and possible unstable when used for large intervals.

The step begins by dividing the time interval into equal step size. The step size used for Euler methods in this study is h = 0.1 for all examples.

The time interval $[t_0, T]$ is divided into N equal step size:

$$h = \frac{T - t_0}{N}. \tag{11}$$

The slope was computed at each time step:

$$f(t_n\,, y_n). \tag{12}$$

These slopes will be used to calculated the next solution:

$$y_{n+1} = y_n + hf(t_n, y_n). \quad (13)$$

This process is repeated across the entire time domain to generate a sequence of approximate solution values.

2. Methodology

2.1 Solving ODE using ANN with BFGS Method

For the first step, the proposed solution known as trial solution is constructed to satisfies the initial condition of first order ODEs. The trial solution associated with ANN is constructed as

$$y_t(x, p) = y_0 + (x - a)N(x, p). \quad (14)$$

Next, Traditional neural networks optimize their parameters to minimize a specific objective function which measure how close the network's predictions match the exact solution. This objective function is commonly referred to as the error function.

$$e(x, \boldsymbol{p}) = \frac{dy_t(x,\boldsymbol{p})}{dx} - f\big(x, y_t(x, \boldsymbol{p})\big). \quad (15)$$

The overall error is then calculated by adding the squared residuals at each discrete collocation point within the domain. The domain [0,1] is discretized, resulting in 11 collocation points This approach ensures that the ANN solution satisfies the ODE as closely as possible across the entire interval.

$$E(\boldsymbol{p}) = \sum_{i=1}^{n} e(x_i, \boldsymbol{p})^2. \quad (16)$$

Now, it is ready to design the learning algorithm. The ANN is trained with BFGS method to find approximate solution for ODEs. Before the training process start, the network parameters are need to be set up. The network parameters are set to suitable initial values at the start of the learning process. The weights are set with random values that taken from uniform random distribution while the biases are initialized to zero. The output layer produces continuous-valued outputs that match the ODE solution while the hidden layer uses either the hyperbolic tangent or sigmoid activation function.

$$N(x, \boldsymbol{p}) = \sum_{i=1}^{n} v_i f(w_i x + b_1). \quad (17)$$

After that, a single input from the domain is ready to be sent into the network through forward propagation until the ANN produces an output. During this phase, the value of error function is calculated. Using optimization technique, the learning process is performed by minimizing the error function. This error function is minimized to make sure that the neural network solution closely matches the ODE across the entire domain. Second order optimization method is used to solve ODE. This gradient-based optimization technique uses slope information to identify the necessary path to find minimum error function. Finding the error function's first order derivative results in the gradient of the error function.

$$\frac{dN(x,\boldsymbol{p})}{dx} = \left(\sum_{i=1}^{n} v_i f'(w_i x + b_i) w_i\right). \quad (18)$$

The algorithm updates the approximation with each iteration considering changes in parameter vector along with gradients. The learning process is repeated until convergence conditions are reached. Once convergence is reached, optimized parameters are applied to give an estimated solution. Finally, a final output of ANN is obtained, thus final the final approximate solution in trial solution is achieved.

2.2 Learning Solver: Broyden-Fletcher-Goldfarb- Shanno Method

An efficient technique for solving unconstrained optimization is the BFGS approach. BFGS is a quasi-Newton second-order optimization method that uses data from the Hessian matrix to achieve optimal results faster than other methods in first-order optimization. The optimization solver in Matlab software known as fminunc can be used instead of manually applying the formula. Network parameters are initialized with random numbers from a uniform distribution for w_i and v_i. Then, the biases in the network are set to zero. The remaining input in fminunc that need to be fulfilled is options. This input defined the output of other approach called optimset. This optimset has many arguments but only several settings for this approach are modified in ANN with BFGS on solving ODEs. This can be summarized 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

3. Results

3.1 Problem 1 [14]

A first-order linear ODE is

$$\frac{dy}{dx} = 2x + 1, \qquad x \in [0,1]. \tag{19}$$

subject to $y(0) = 1$. The exact solution for this problem is

$$y(x) = x^2 + x + 1. \tag{20}$$

The trial solution is

$$y_t(x, p) = 1 + x(N(x, p)). \tag{21}$$

Table 2
Absolute error obtained by ANN with BFGS using sigmoid, ANN with BFGS using tanh and Euler's Method for Problem 1

x	ANN with BFGS using sigmoid	ANN with BFGS using tanh	Euler's Method
0.0	0.00	0.00	0.00
0.1	4.75×10^{-5}	9.49×10^{-7}	0.01
0.2	1.10×10^{-4}	7.13×10^{-6}	0.02
0.3	9.03×10^{-5}	1.06×10^{-5}	0.03
0.4	3.03×10^{-5}	8.78×10^{-6}	0.04
0.5	4.62×10^{-7}	3.15×10^{-6}	0.05
0.6	2.70×10^{-5}	2.59×10^{-6}	0.06
0.7	8.20×10^{-5}	4.46×10^{-6}	0.07
0.8	1.02×10^{-4}	5.15×10^{-7}	0.08
0.9	5.33×10^{-5}	6.64×10^{-6}	0.09
1.0	1.49×10^{-5}	7.14×10^{-6}	0.10

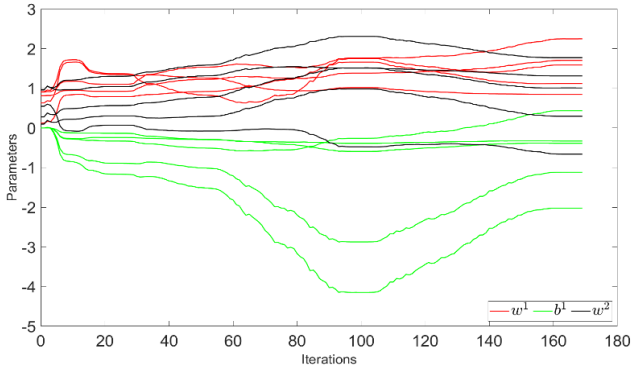

Fig. 2. Convergence of network parameters of ANN with BFGS using sigmoid activation function for Problem 1

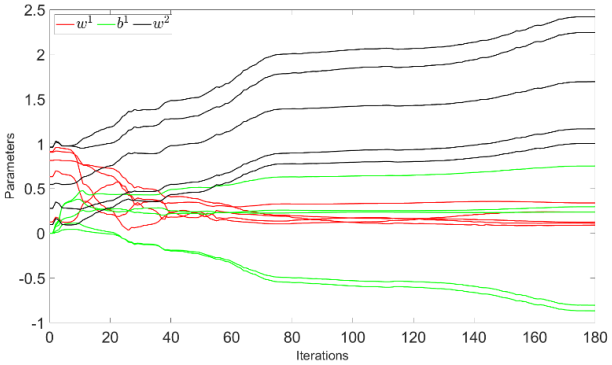

Fig. 3. Convergence of network parameters of ANN with BFGS using tanh activation function for Problem 1

3.2 Problem 2 [15]

A first order nonlinear ODE is

$$\frac{dy}{dx} = -y - y^2, \tag{22}$$

subject to $y(0) = 1$. The exact solution for this problem is

$$y(x) = \frac{1}{-1+2e^x}. \tag{23}$$

The trial solution is

$$y_t(x,p) = 1 + x(N(x,p)). \tag{24}$$

Table 3
Absolute error obtained by ANN with BFGS using sigmoid, ANN with BFGS using tanh and Euler's Method for Problem 2

x	ANN with BFGS using tanh	ANN with BFGS using sigmoid	Euler's Method
0.0	0.00	0.00	0.00
0.1	5.01×10^{-4}	7.56×10^{-4}	9.09×10^{-3}
0.2	9.19×10^{-4}	1.44×10^{-3}	1.43×10^{-2}
0.3	5.95×10^{-4}	9.01×10^{-4}	1.73×10^{-2}
0.4	2.68×10^{-5}	7.96×10^{-5}	1.89×10^{-2}
0.5	3.31×10^{-4}	6.96×10^{-4}	1.96×10^{-2}
0.6	3.23×10^{-4}	6.25×10^{-4}	1.98×10^{-2}
0.7	3.08×10^{-5}	2.45×10^{-6}	1.97×10^{-2}
0.8	3.34×10^{-4}	7.80×10^{-4}	1.93×10^{-2}
0.9	5.24×10^{-4}	1.19×10^{-3}	1.89×10^{-2}
1.0	3.18×10^{-4}	7.04×10^{-4}	1.83×10^{-2}

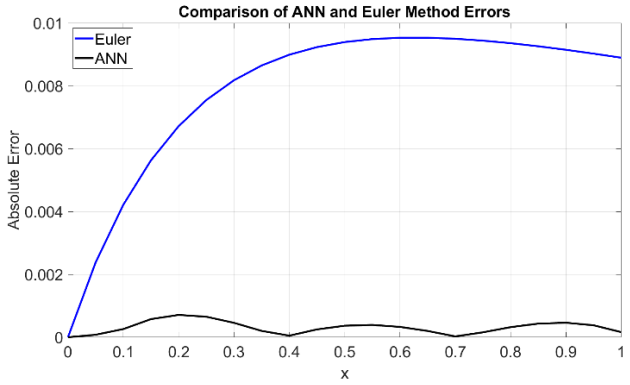

Fig. 4. Comparison absolute error between ANN with BFGS using hyperbolic tangent tanh activation function with Euler's Method for Problem 2

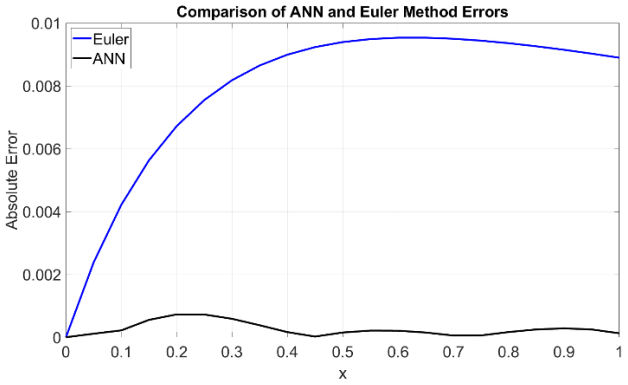

Fig. 5. Comparison absolute error between ANN with BFGS using sigmoid activation function with Euler's Method for Problem 2

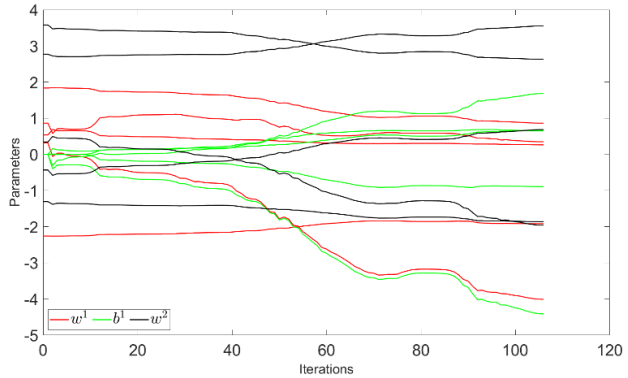

Fig. 6. Convergence of network parameters of ANN with BFGS using hyperbolic tangent tanh activation function for Problem 2

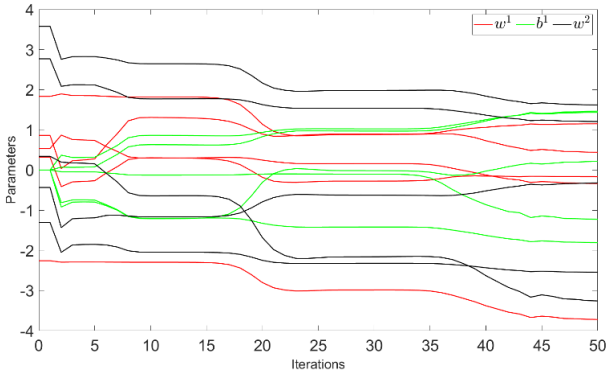

Fig. 7. Convergence of network parameters of ANN with BFGS using sigmoid activation function for Problem 2

3.3 Problem 3 [16]

A first order nonlinear ODE is

$$\frac{dy}{dx} = -y^2, \tag{25}$$

subject to $y(0) = 1$. The exact solution for this problem is

Table 4
Absolute error obtained by ANN with BFGS using sigmoid, ANN with BFGS using tanh and Euler's Method for Problem 3

x	ANN with BFGS using tanh	ANN with BFGS using sigmoid	Euler's Method
0.0	0.00	0.00	0.00
0.1	2.66×10^{-4}	3.10×10^{-4}	9.09×10^{-3}
0.2	7.15×10^{-4}	7.23×10^{-4}	1.43×10^{-2}
0.3	6.43×10^{-4}	6.22×10^{-4}	1.73×10^{-2}
0.4	2.96×10^{-4}	1.73×10^{-4}	1.89×10^{-2}
0.5	2.53×10^{-5}	2.57×10^{-4}	1.96×10^{-2}
0.6	3.21×10^{-5}	3.99×10^{-4}	1.98×10^{-2}
0.7	6.45×10^{-5}	1.85×10^{-4}	1.97×10^{-2}
0.8	1.81×10^{-4}	2.43×10^{-4}	1.93×10^{-2}
0.9	2.17×10^{-4}	5.83×10^{-4}	1.89×10^{-2}
1.0	1.59×10^{-4}	4.32×10^{-4}	1.83×10^{-2}

$$y(x) = \frac{1}{1+x}. \tag{26}$$

The trial solution is

$$y_t(x, p) = 1 + x(N(x, p)). \tag{27}$$

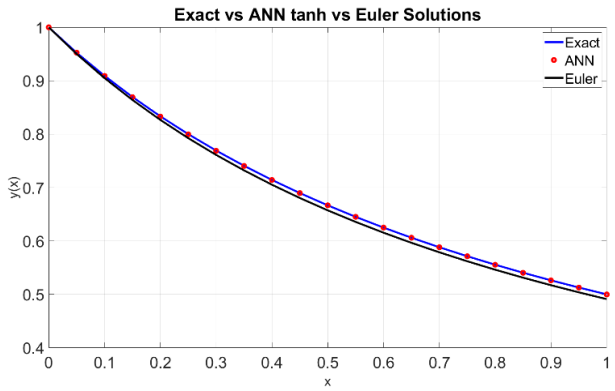

Fig. 8. Comparison numerical solutions between ANN with BFGS using hyperbolic tangent tanh activation function with Euler's Method and exact solutions for Problem 3

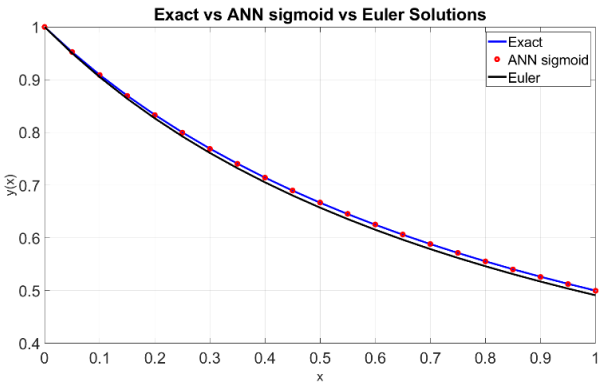

Fig. 9. Comparison numerical solutions between ANN with BFGS using sigmoid activation function with Euler's Method and exact solutions for Problem 3

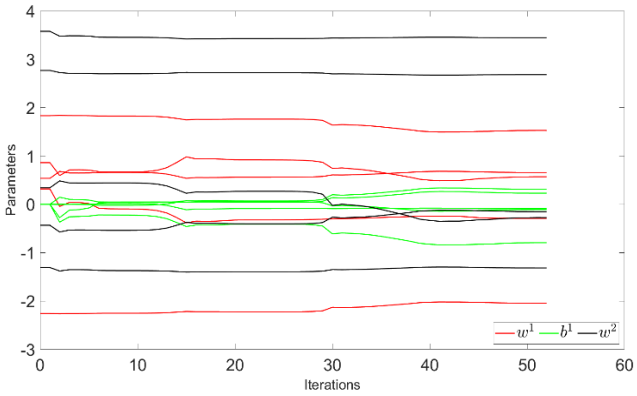

Fig. 10. Convergence of network parameters of ANN with BFGS using tanh activation function for Problem 3

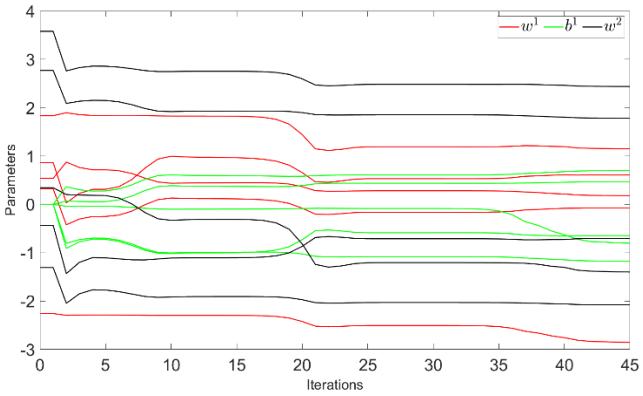

Fig. 11. Convergence of network parameters of ANN with BFGS using sigmoid activation function for Problem 3

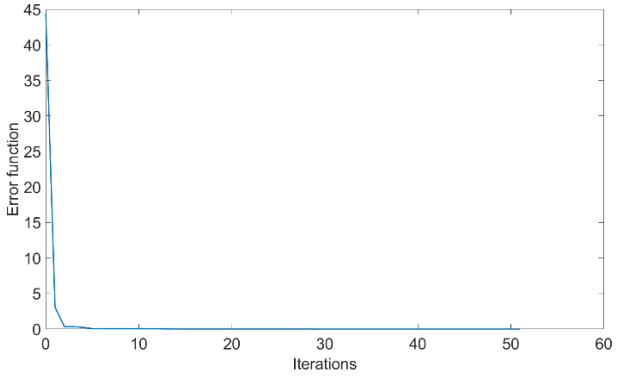

Fig. 12. Error function of ANN with BFGS using tanh activation function for Problem 3

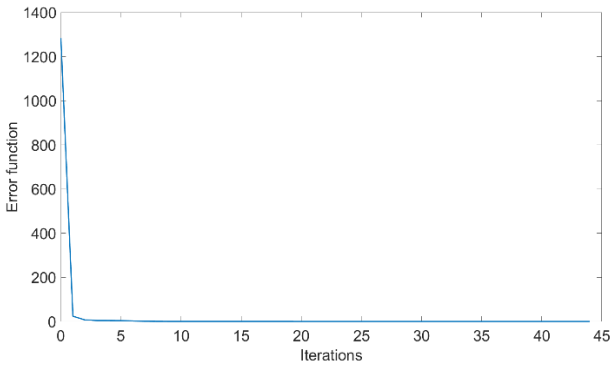

Fig. 13. Error function of ANN with BFGS using sigmoid activation function for Problem 3

3.4 Problem 4 [17]

$$\frac{dT_1}{dt} = 2a_4T_M - c_1I - (d_2 + a_1)T_1,$$

$$\frac{dT_M}{dt} = a_1T_I - dT_M - c_3T_MI,$$

$$\frac{dI}{dt} = k + \frac{\rho I(T_I+T_M)^n}{\beta+(T_I+T_M)^n} - c_2IT_I - c_4T_MI - d_1I.$$

where $a_4 = 0.8, d_2 = 0.11, a_1 = 1, c_1 = c_3 = 0.9, d = 1.15, k = 0.47, \rho = 0.1, n = 3, \beta = 0.2, c_2 = c_4 = 0.085, d_1 = 0.29.$

subject to

$$T_I(0) = 1.3,\ T_M(0) = 1.2,\ I(0) = 0.9.$$

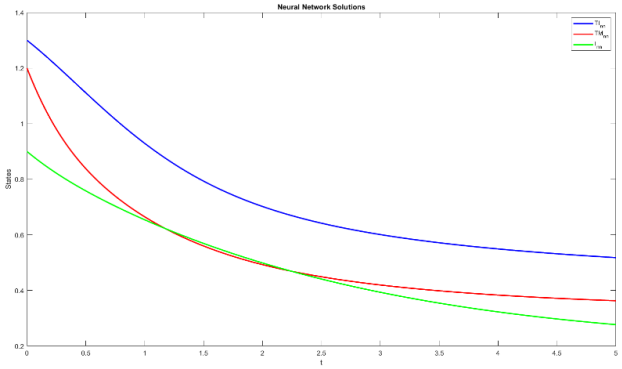

Fig. 14. Numerical solution of Problem 4 using ANN with BFGS

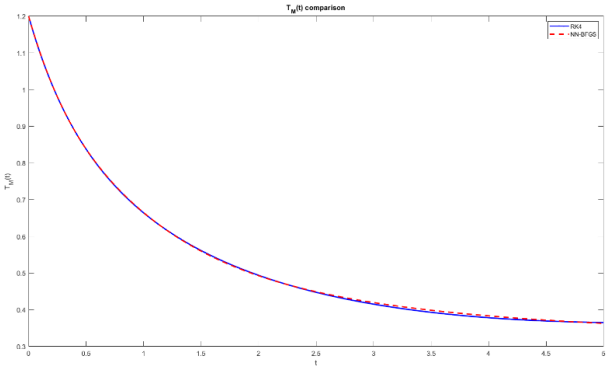

Fig. 15. Comparison of RK4 and ANN with BFGS for T_M in Problem 4

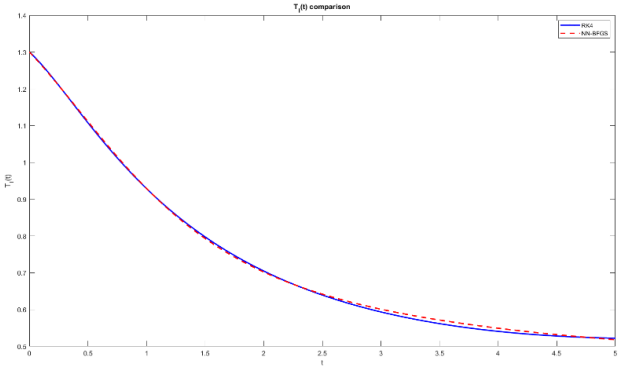

Fig. 16. Comparison of RK4 and ANN with BFGS for T_I in Problem 4

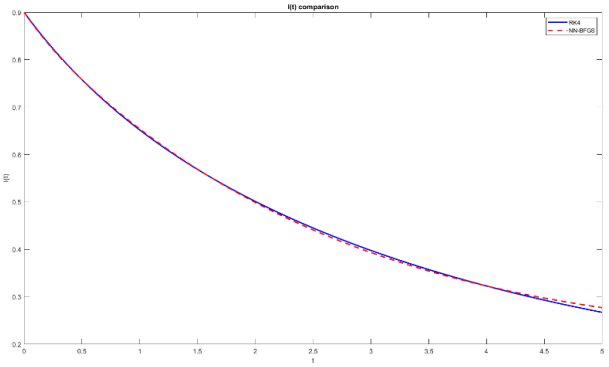

Fig. 17. Comparison of RK4 and ANN with BFGS for *I* in Problem 4

3.5 Discussions

For Problem 1, there was a significant difference in the accuracy of the results. The results indicate that both sigmoid and hyperbolic tangent activation functions produce accurate approximations. While the tanh function generally shows lower errors across most points, sigmoid performs better at certain points indicating that performance is based on the problem rather than universally superior. This can be seen through the absolute error of sigmoid in Table 4.1. For comparison, Euler's method is a simple first-order numerical method for solving ODEs was also applied to this problem. Euler's method can provide a quick approximate solution but larger absolute errors compared to ANN with BFGS results. In terms of convergence, sigmoid activation function converged faster than tanh activation function. However, the error of sigmoid is higher than tanh. In contrast, tanh activation function produced high accuracy than sigmoid even converged slower than sigmoid. This can be seen through Figure 4.1 and Figure 4.2. Based on the parameters, the differences on iterations for convergence between sigmoid and tanh is not much which is between 170 to 180.

For Problem 2, based on Table 3 the sigmoid activation function shows stable performance across all test points and the tanh activation function provides improved accuracy in most regions but may vary at certain points depending on the solution behaviour. In addition, Figure 4.3 and Figure 4.4 shows that Euler's method produces large absolute error compared to ANN with BFGS. In terms of convergence in Figure 4.5 and Figure 4.6, tanh activation function converged slower than sigmoid activation function. The plot displays a better error range with continuously reduced error. This indicates that tanh approximated the solution more effectively and accurately.

For Problem 3, the numerical results for both cases are shown with the exact solution. Based on the Figure 4.7 and Figure 4.8, there was a significant difference in the accuracy of the results. The ANN with BFGS is better than Euler's method leading to lower absolute errors. In contrast, tanh activation function produced high accuracy than sigmoid even converged slower than sigmoid. Table 4 shows the details of all methods error. Figures 4.9 and 4.10 show that the sigmoid activation function converged more quickly than the tanh activation function. On the other hand, tanh has a lower error than sigmoid. The tanh activation function was more accurate than the sigmoid and even converged more slowly. Compared to the sigmoid error function in Figure 4.12, the tanh activation function in Figure 4.11 produces faster reduction in the error function due to its zero-centered output.

For Problem 4, the results of simulations from the three tumours system show the importance of immune response and drugs in tumour growth control. The figure shows the interphase tumour population as T_I ,metaphase tumour population as T_M and the immune drug population as I. The graph highlights the role of immune response and drug treatment in controlling tumour growth. The stability analysis shows that the tumour growth is reduced compared to the system without immune response. This can be seen in Figure 4.13 when T_I remains the highest but steadily became lower and T_M decrease rapidly. The combination of immune response and drug provides a strong control mechanism against tumour growth than immunity alone leading to a more reduction in tumour cell populations over time.

Comparing the RK4 results with the ANN with BFGS in Figure 4.14, Figure 4.15 and Figure 4.16 successfully show the overall results of tumour and immune drug response. The RK4 method also provide stable reference solution satisfying the initial conditions. While ANN with BFGS able to approximate the overall system and provide continuous solution which can be advantageous for optimization.

In conclusion, RK4 remains reliable method for direct numerical integration while ANN shows potential in nonlinear system behaviour and the combined immune and drug even if the exact solution does not provided.

4. Conclusions

The results show that the ANN with BFGS optimization can accurately solve ODEs and performs better than the Euler Method that has larger errors. Both sigmoid and tanh activation functions are effective but tanh generally produces lower errors while sigmoid performs better at certain points. It shows that the training performance depends on the problem. The BFGS method improves convergence speed and stability during training. Comparison with the RK4 method is limited in this study since the results are only presented for Example 4, where the results appear similar with ANN. However, this is insufficient to conclude across all cases. Overall, ANN with BFGS is a promising method for solving ODEs, but further comparisons with RK4 across all examples are needed for stronger conclusions.

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

References

[1] Paudel, Dharma Raj, and Mohan Raj Bhatta. "Comparative study of Euler's method and Runge-Kutta method to solve an ordinary differential equation through a computational approach." *Academic Journal of Mathematics Education* 6, no. 1 (2023): 81-85. https://doi.org/10.3126/ajme.v6i1.63802

[2] Okereke, R. N. "A New Perspective to the Solution of Ordinary Differential Equations using Artificial Neural Networks." *Ph. D. dissertation* (2019).

[3] Hatcher, William Grant, and Wei Yu. "A survey of deep learning: Platforms, applications and emerging research trends." *IEEE access* 6 (2018): 24411-24432.https://doi.org/10.1109/ACCESS.2018.2830661

[4] Li, C., & Wei, X. (2020). Adaptive Memory Multi-Batch L-BFGS for Efficient Neural Network Training. Computer Science Review, 35, 17-29.

[5] Sharma, Sagar, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks." *Towards Data Sci* 6, no. 12 (2017): 310-316. https://doi.org/10.33564/IJEAST.2020.v04i12.054

[6] Cybenko, George. "Approximation by superpositions of a sigmoidal function." *Mathematics of control, signals and systems* 2, no. 4 (1989): 303-314. https://doi.org/10.1007/BF02551274

[7] 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

[8] Nocedal, J., and S. Wright. "Numerical optimization. 2nd edn Springer." New York (2006).

[9] Sharma, S., Sharma, S., & Athaiya, A. (2020). Activations Function in Neural Networks. International Journal of Engineering Applied Sciences and Technology, 04(12), 310–316. https://doi.org/10.33564/IJEAST.2020.v04i12.054

[10] Pratiwi, Heny, Agus Perdana Windarto, S. Susliansyah, Ririn Restu Aria, Susi Susilowati, Luci Kanti Rahayu, Yuni Fitriani, Agustiena Merdekawati, and Indra Riyana Rahadjeng. "Sigmoid activation function in selecting the best model of artificial neural networks." In *Journal of physics: conference series*, vol. 1471, no. 1, p. 012010. IOP Publishing, 2020. https://doi.org/10.1088/1742-6596/1471/1/012010

[11] Chen, Zhi-xiang, and Fei-long Cao. "The construction and approximation of feedforward neural network with hyperbolic tangent function." *Applied Mathematics-A Journal of Chinese Universities* 30, no. 2 (2015): 151-162. https://doi.org/10.1007/s11766-015-3000-9

[12] Islam, Md Amirul. "A Comparative Study on numerical solutions of initial value problems (IVP) for ordinary differential equations (ODE) with Euler and Runge-Kutta methods." *American Journal of computational mathematics* 5, no. 03 (2015): 393-404. https://doi.org/10.4236/ajcm.2015.53034

[13] Salleh, Z., et al. (2010). Ordinary Differential Equations (ODE) Using Euler's Technique and MATLAB Implementation. Proceedings of the 12th WSEAS International Conference on Mathematical Methods, Computational Techniques, and Intelligent Systems (MAMECTIS '10).

[14] Chakraverty, Snehashish, and Susmita Mall. *Artificial neural networks for engineers and scientists: solving ordinary differential equations*. CRC Press, 2017. https://doi.org/10.1201/9781315155265

[15] Al-Omari, Ahmad, Heinz-Bernd Schüttler, Jonathan Arnold, and Thiab Taha. "Solving nonlinear systems of first order ordinary differential equations using a Galerkin finite element method." *Ieee Access* 1 (2013): 408-417. https://doi.org/10.1109/ACCESS.2013.2269192

[16] John, Sabo, Adewale A. James, Ayinde M. Abdullahi, Umar Medina Osuma, Akanbi Muhammed Bello, and Belgore Ahmad Kawu. "Numerical Methods for Solving First-Order Nonlinear Differential Equations Using a Linear Block Approach." In *AUN INTERNATIONAL CONFERENCE*, vol. 2, no. 1, pp. 27-37. 2024.

[17] 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