RADIAL BASIS NEURAL NETWORK FOR THE SOLUTION OF OPTIMAL CONTROL PROBLEMS VIA SIMULINK

[a]Aduroja, O. O.
[b*]Adamu, S. (https://orcid.org/0000-0003-1690-4269)
[c]Ajileye, A. M.

[a,c] Department of Mathematics, University of Ilesa, Ilesa, Osun State, Nigeria
[b] Department of Mathematics, Nigerian Army University Biu, Borno State, Nigeria
*Corresponding Author: malgwisa@gmail.com (Adamu, S.)

How to Cite

Adamu, S., Aduroja, O. O., & Ajileye, A. M. (2024). RADIAL BASIS NEURAL NETWORK FOR THE SOLUTION OF OPTIMAL CONTROL PROBLEMS VIA SIMULINK. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *15*(2), 291–308. https://doi.org/10.61841/turcomat.v15i2.14728

Abstract: This study uses radial basis neural network to solve optimum control problems via simulink. Using Pontryaginí's principle, the optimal control problem's optimum system is constructed. MATLAB is used to simulate the optimality system and generate the simulink architecture for the trial value. A radial basis neural network is then used to train the system and produce the optimal solution. This approach's effectiveness is evaluated using a few control problems, and it is shown to be effective because of the reliable, accurate, and consistent results that are produced. The performance of this strategy is superior to that of other approaches.

Keywords: Numerical solution, Optimal control problem, Radial basis neural network

1. Introduction

The goal of optimal control is to identify a dynamic system's input that maximizes a given performance index while satisfying all applicable constraints [1, 2, 3]. Since 1950, there has been a significant development in the generalization of the calculus of variation to optimal control theory, driven mostly by military applications [4, 5]. The majority of the time, optimum control issues are solved numerically because of the complexity of the applications, which makes analytical solutions difficult or impossible. The optimal control issue was first solved numerically in the 1950s thanks to the efforts of [6]. The intricacy of the optimal control approach has grown along with the complexity and variety of applications, to the point that optimal control theory is now a science that is significant to many areas of human endeavour [1].

Almost every field of human endeavour has seen the application of optimal control theory to simulate issues in the social sciences, engineering, medical, biological, and agricultural sciences. Numerical approaches are employed since it is difficult to find analytical answers for some of these complex issues. Numerical techniques are mostly in iterative form; to get a solution, a step size is fixed before the computation is started. The process must be repeated from the beginning stage in order to determine the solution in between stages [7]. When an Artificial Neural Network (ANN) defeats this iteration loop, relief is experienced [8].

This paper considers the numerical solution to optimal control problems of the form

$$J(u)=\int_{t_0}^{T_f} f\big(t,x(t),u(t)\big)dt \tag{1}$$

subject to

$$\dot{x} = G(t, x(t), u(t)) \tag{2}$$

$$x(t_0) = 0,\ x(t_f) \text{ is free}$$

Three approaches can be used to solve Equations (1) and (2): (i) Dynamic programming; (ii) Direct method (nonlinear programming); and (iii) Indirect method (Pontryagin Maximum Principles). It is well known that indirect methods have a very good convergence speed and are more accurate than direct methods [1, 3, 5, 9, 10]. This study's methodology is based on the indirect way of solving control problems. Determining the prerequisite for optimality is the goal of the indirect method, a class of analytical optimization techniques related to the Pontryagin maximum principle.

An artificial neural network attempts to replicate the learning process of the human brain. The word "artificial" refers to the fact that neural networks are used on computer programs that can manage the numerous calculations required for learning [11]. Artificial Neural Networks (ANNs) replicate some of the functions seen in human brains [11, 12]. One of the most well-liked fields of Artificial Intelligence (AI) study is artificial neural networks, which are abstract computational models inspired by the structure of the human brain [13].

Simulink features a graphical user interface that makes it easy to see the system's structure. Applications in real time are made possible by this [14]. Simulink is preferred in this work over other numerical methods like Runge-Kutta method because, in Simulink, you design, simulate, develop, and test a wide range of time-varying systems, with Simulink's configurable block libraries and interactive graphical environment. Instead of requiring code, Simulink provides a graphical user interface for numerical problem solving. Without developing a lot of code, Simulink will be able to perform computations and resolve challenging algorithmic problems.

The Radial Basis Function Neural Network (RBFNN) was initially presented by [15] as a solution to the multi-dimensional interpolation problem, which needed as many centres as data points. Subsequently, [16] eliminated the strict restriction and employed fewer centres than data samples, enabling a large variety of useful RBFNN applications with a large sample size. The existence of a quick, linear learning algorithm in a network that may express intricate non-linear mappings is a crucial characteristic of RBFNNs [17, 18].

The input layer, hidden layer, and output layer are the three layers that make up the RBFNN structure. Every node, or neuron, uses an RBF in the buried layer and functions as a nonlinear activation function. The input layer is made up of an input vector that the hidden layer transforms nonlinearly; in other words, RBF makes up the hidden layer. The vector distance between the weight and the input vector, multiplied by the relevant bias, is the net input for the RBF activation function. The nonlinearity is mapped into a new space by the output layer, a linear combiner. It is possible to simulate the biases of the output layer neurons by adding a new neuron to the hidden layer.

[19] used a single layer feed forward ANN with supervised learning technique to address single variable optimal control problems. A single layer perceptron with several neurons is insufficient to approximate any function, according to [20]. They suggested a two-layer network known as universal approximators, with a linear transfer function in the output layer and a sigmoid transfer function in the hidden layer.

The forward-backward sweep approach using the Pontryagin's principle is used by [1, 3, 5, 9] to solve iterative and block methods for optimal control problems modeled in ordinary differential equations. After studying single-layer neural networks for some time, multilayer perceptrons, or MLPs, were proposed in 1975. Because of their multilayer construction, MLPs were computationally intensive [21]. The functional neural network was created as a result of the lengthy training times and high computational complexity of MLPs. By adding variable functions, this network was able to overcome the complexity of multiple layers [22].

An optimal step size algorithm for accelerating training is introduced by [23], reducing computational complexity of Jacobin and Hessian matrices. The BPOS algorithm, specifically designed for feed-forward neural networks, is effective and suitable for larger training samples and fewer outputs. [24] introduced a sparse training method with sparse forward and backward passes. The method is a continuous minimization problem under global sparsity constraint, separated into weight update and structure parameter update steps, using a variance reduced policy gradient estimator.

Learning algorithm called forward signal propagation learning (sigprop) is introduced by [25], which eliminates structural or computational constraints like feedback connectivity or weight transport. This method enables global supervised learning with only a forward path, making it ideal for parallel training layers or modules. [26] study demonstrates Forward Propagation Through Time (FPTT) learning combined with liquid time-constant spiking neurons, enabling online learning of long sequences and temporal classification tasks, and directly training large-scale dynamic and complex spiking neural network architectures.

Since the multilayer feed forward radial basis neural network has been shown to be efficient [21], MATLAB-Simulink is used in this study to generate the trial values. After that, an RBF (Radial Basis Function) neural network is trained on the trial values to produce accurate and efficient results in the lowest amount of training time. This method uses less computer memory and less computational rigor, making it more efficient.

2. Methodology

2.1 Artificial Neural Network

ANN has been developed as generalization of mathematical models of human cognition or neural biology based on the assumption that:

(i) information processing occurs at many simple connections called neurons,

(ii) signals are passed between neurons over connection links,

(iii) each connection links has an associated weight, which in a typical neural net, multiplies the signal transmitted,

(iv) each neuron applies an activation function to its net input to determine its output signal [27].

***Theorem 1*: Pontryagin's Maximum Principle**

Using [10] as a guide, and given the optimal control problem (1) and (2), iff $u^*(t)$ and $x^*(t)$ are optimal, then there exists a piecewise differentiable adjoint variable $\lambda(t)$ such that

$$H\big(t,x^*(t),u(t),\lambda(t)\big)\leq H\big(t,x^*(t),u^*(t),\lambda(t)\big)$$

$$H = f\big(t,x(t),u(t)\big)+\lambda G\big(t,x(t),u(t)\big) \tag{3}$$

for all controls u at each time t, where H is the Hamilton and

$$\lambda'(t) = -\frac{\partial H(t,x^*(t),u^*(t),\lambda(t))}{\partial x} \tag{4}$$

$$\lambda(t_f)=0 \tag{5}$$

where λ are additional Lagrange multiplier functions called the adjoint and is dependent of t, x and u. The optimality and adjoint conditions in this work will be generated using this result of Pontryagin's Principle.

The necessary and sufficient conditions required are

$$\frac{\partial H}{\partial u}=0, u^* \Rightarrow f_u+\lambda g_x=0 \text{ Optimality condition}$$

$$\lambda' = -\frac{\partial H}{\partial x} = -\left(f_u+\lambda g_x\right) \text{ Adjoint equation}$$

$$\lambda(t_f)=0 \text{ Transversality condition}$$

$$x' = g(t,x,u) \text{ Dynamics of the state equation}$$

$$x(t_0) = x_0.$$

See [1, 9, 10] for details.

2.2 Radial Basis Function Neural Network (RBFNN)

The output of the network for an input pattern x is shown in the following form

$$y_i(x) = \sum_{k=1}^{h} w_{ki}\phi\|x - c_k\| \qquad (6)$$

For $i = 1,\dots,j,$ where $y(x)$ represents the ith output of the RBF, w_{ki} denotes the connection weight from the kth hidden unit to the ith output unit, c_k is prototype of center of the kth hidden unit, and ❑⊑ indicates the Euclidean norm, the RBF $\phi(.)$ is usually considered as the Gaussian function [28].

For a set of N pattern pairs of (x_p, y_p), Equation (6) can be written in matrix form:

$$Y = W^T\phi \qquad (7)$$

where $W = [w_1,\dots,w_{J3}]$ shows a J_2xJ_3 is weight matrix in which $w_i = (w_{1i},\dots.,w_{J2i})^T, \phi = [\phi_1,\dots,\phi_N]$ denotes a J_2xN matrix, $\phi_p = (\phi_{p,1},\dots,\phi_{p,J2})^T$ is the output of the hidden layer for the pth sample, $\phi_{p,k} = \phi(?x_p - c_k ?), Y = [y_1, y_2,\dots,y_N]$ indicates a J_3xN matrix, and $y_p = (y_{p,1},\dots,y_{p,J3})^T$ [28].

2.3 MATLAB Simulink

The simulink of optimal control problems are done by the following four steps:

(i) The highest state derivatives are identified.

(ii) These states are integrated once or more to obtain all states.

(iii) The highest state is computed with a simulation.

(iv) The signals entering the summation are computed by means of gains multiplying known signals and inputs.

Algorithm 2.1: Feed forward algorithm

```
tfinal=1;
[t,x]=sim('problem1',tfinal);
P=t'; T=x';
net=newff([0 1],[10 2],{'tansig','purelin,'),'trainlm');
net.trainParam.show=10;
net.trainParam.lr=0.05;
net.trainParam.goal=1e-10;
net1=train(net,P,T)
```

a=sim(net1,P)

plot(P,a,P,a-T) %error plot

Algorithm 2.2: Radial basis algorithm

goal=1e-10;

spread=0.1;

net=newrb(P,T,goal,spread)

a=sim(net,P)

plot(P,a,P,T-a) %error plot

gensim(net,-1)

plot(P,a) gives the plot for the results

x(:,1), x(:,2)

3. Numerical Examples

The Architecture

The Feed Forward Neural Network Architecture features for the code used to solve Problem 1 to Problem 4 are shown in Figures 1 through 5.

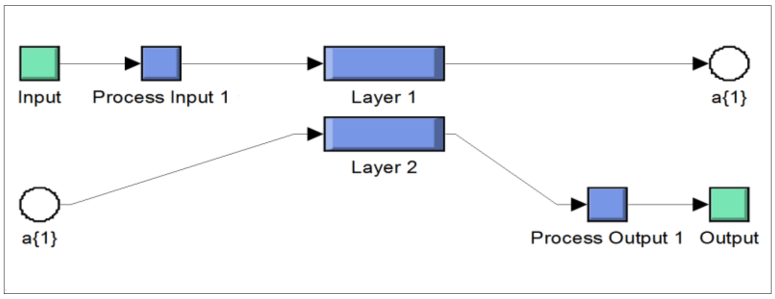

Figure 1: Feed forward architecture for the problems

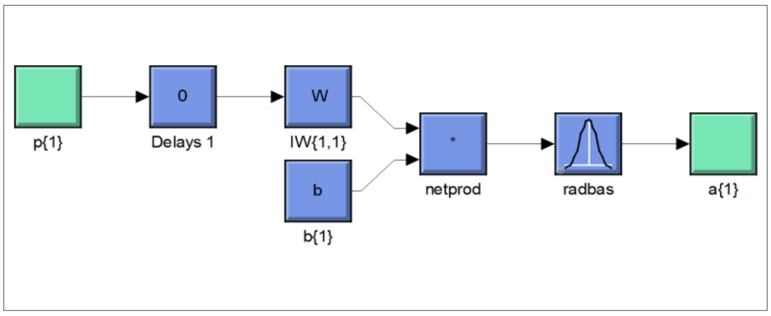

Figure 2: First layer of feed forward architecture for the problems

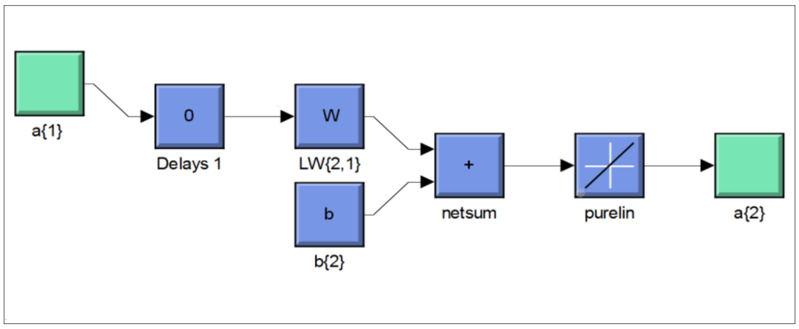

Figure 3: Second layer of feed forward architecture for the problems

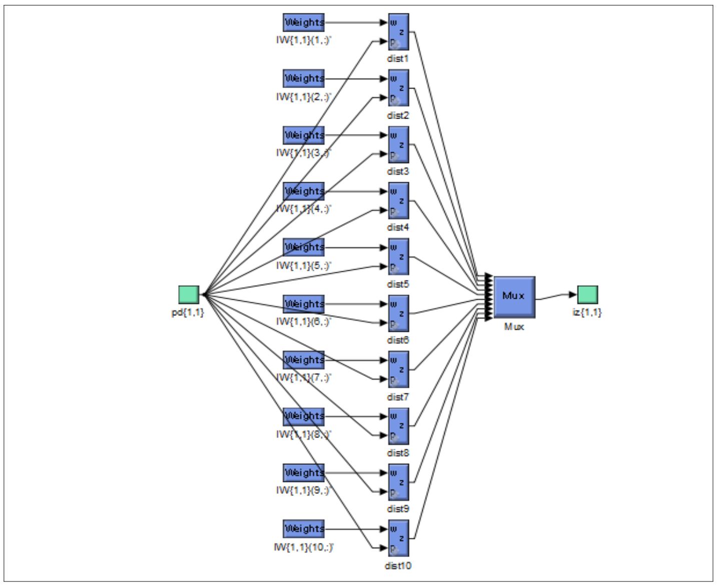

Figure 4: Weight of first layer feed forward architecture for the problems

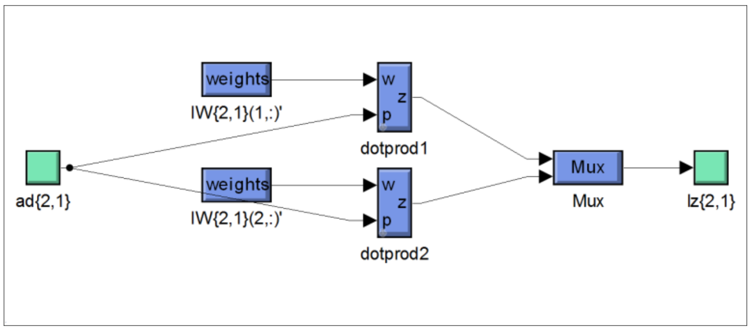

Figure 5: Weight of second layer feed forward architecture for the problems

Problem 1: *[1, 5] considered the optimal control problem*

$$\min_u J = \frac{1}{2}\int_0^1 x^2(t) + u^2(t)dt,$$

(8)

subject to

$$x'(t) = -x(t) + u(t),\ x(0) = 1, \tag{9}$$

with the optimal solution

$$x^*(t) = \frac{\sqrt{2}\cosh\left(\sqrt{2}(t-1)\right) - \sinh\left(\sqrt{2}(t-1)\right)}{\sqrt{2}\cosh\left(\sqrt{2}\right) + \sinh\left(\sqrt{2}\right)},$$

$$u^*(t) = -\frac{\sinh\left(\sqrt{2}(t-1)\right)}{\sqrt{2}\cosh\left(\sqrt{2}\right) + \sinh\left(\sqrt{2}\right)}.$$

Solution: The optimality system is first developed by constructing Hamiltonian

$$H = \frac{1}{2}x^2 + \frac{1}{2}u^2 - \lambda x + \lambda u.$$

The optimality condition

$$\frac{\partial H}{\partial u} = u + \lambda \Rightarrow u = -\lambda. \tag{10}$$

The adjoint equation

$$\lambda' = -\frac{\partial H}{\partial x} = -x + \lambda,\ \lambda(1) = 0. \tag{11}$$

$$x'(t) = -x + u, x(0) = 1 \tag{12}$$

$$-\frac{\partial H}{\partial x} = -x + \lambda$$

The numerical integration of (11)

$$\lambda(1) = \lambda(0) - \int_0^1 x(t)dt + \int_0^1 \lambda(t)dt$$

$$\lambda(0) = 0.38582$$

Using the result $\lambda(0)$ obtained to initialize the Simulink constructed below

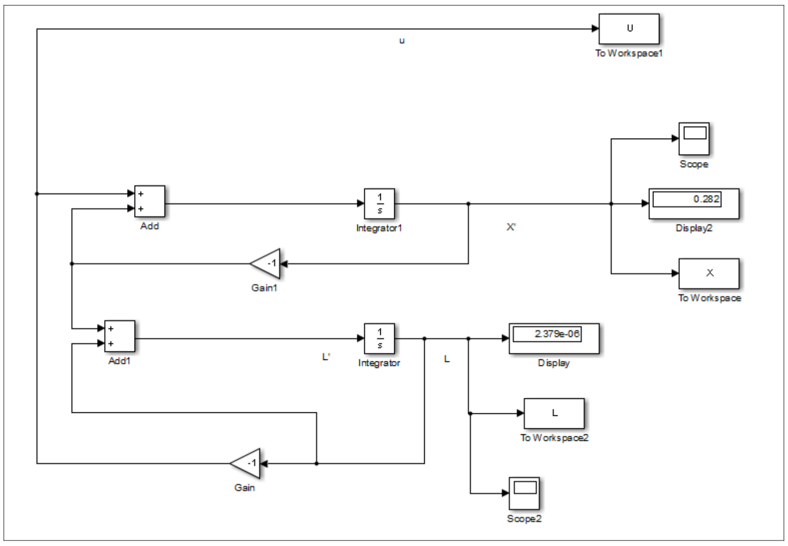

Figure 6: MATLAB simulink for problem 1

Blocks which are relevant to the example are gathered from the simulink blocks library, modified to suit to the desired model and joined with lines to indicate transfer of signals. The complete system is then run.

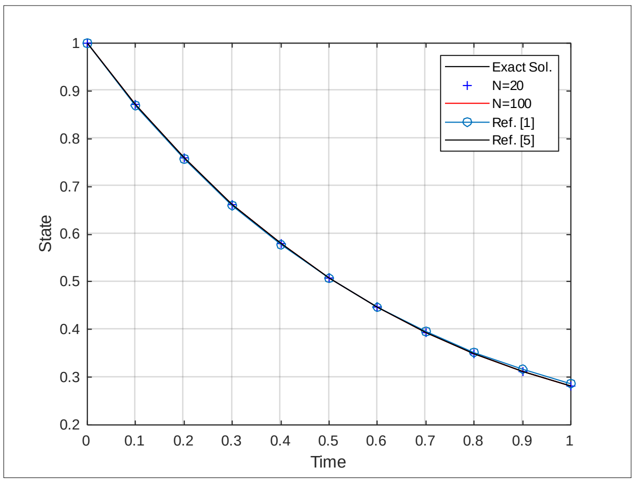

Figure 7: State result for problem 1

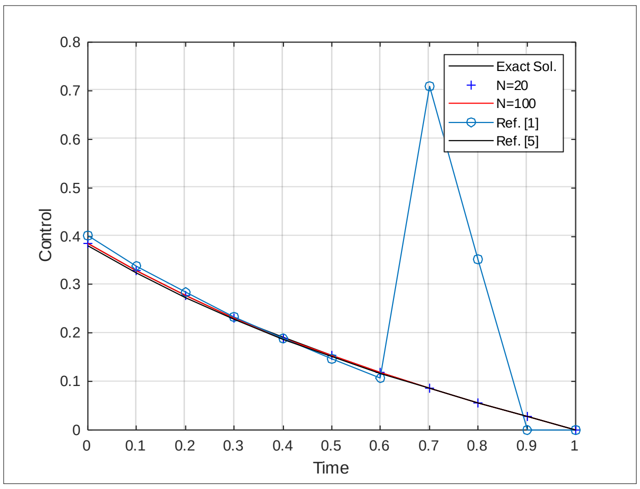

Figure 8: Control result for problem 1

Problem 2: *[1] consider the optimal control problem*

$$\max_u J = \int_0^1 Ax(t) - Bu^2(t)dt$$

$$x'(t) = -\frac{1}{2}x(t)^2 + Cu(t)$$

$$x(0) > -2$$

$$A \geq 0, B \geq 0.$$

Solution: The Hamailtonian

$$H = Ax + Bu^2 - \frac{1}{2}\lambda x^2 + C\lambda u$$

$$\frac{\partial H}{\partial u} = 0 = -2Bu + C\lambda \Rightarrow u = \frac{C\lambda}{2B} \quad (13)$$

$$\lambda' = -\frac{\partial H}{\partial x} = -A + \lambda x, \lambda(1) = 0 \quad (14)$$

$$x'(t) = -\frac{1}{2}x^2(t) + Cu(t), \ x(0) = x_0 \quad (15)$$

From (14)

$$\lambda(1) = \lambda(0) + \int_0^1 \left(-A + \lambda(s)x(s)\right)ds$$

Using numerical integration

$$\lambda(0) = 2.29113$$

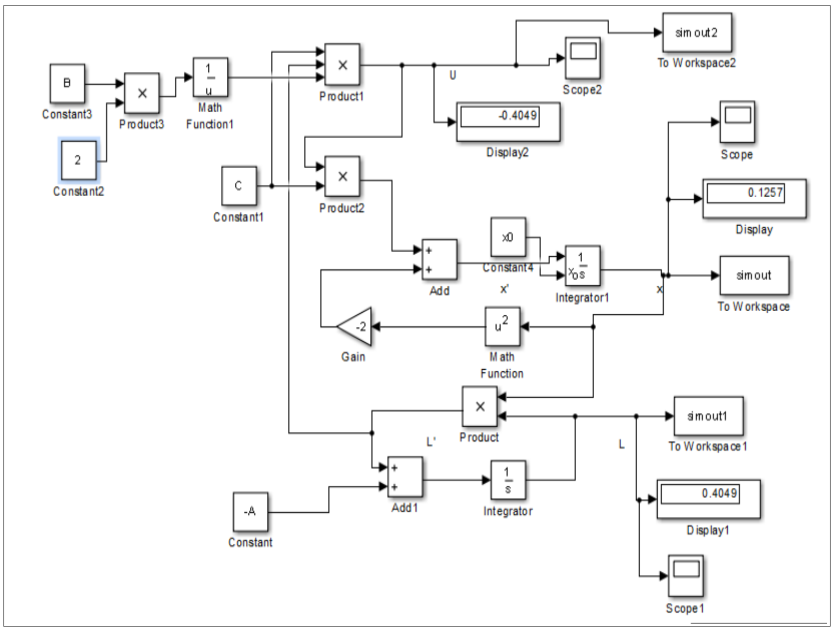

Figure 9: MATLAB simulink for problem 2

Blocks which are relevant to the example are gathered from the simulink blocks library, modified to suit to the desired model and joined with lines to indicate transfer of signals. The complete system is then run.

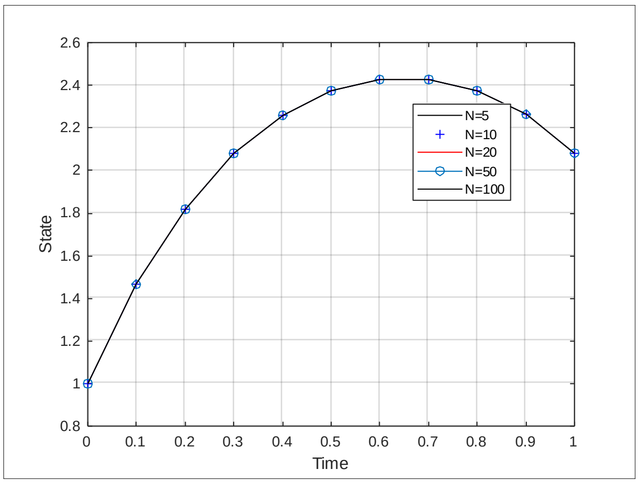

Figure 10: State result for problem 2

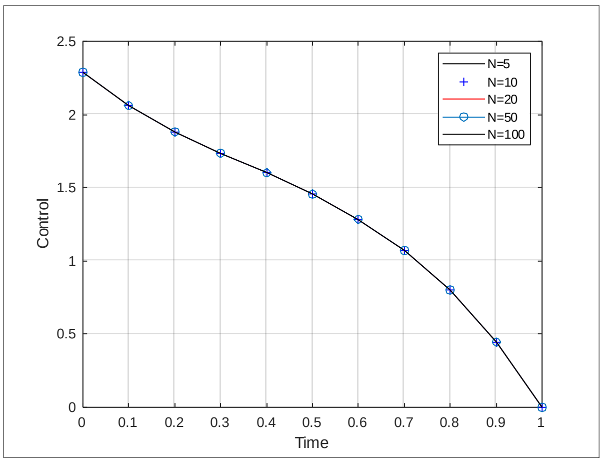

Figure 11: Control result problem 2

Problem 3: *[9] considered the first order optimal control problem*

$$\min_u J = \int_0^1 \left(x(t) + \frac{1}{2}u^2(t) \right) dt$$

subject to

$$x'(t) = x(t) + u(t),\ x(0) = \frac{1}{2}e^2 - 1$$

Solution: Constructing the Hamiltonian

$$H = x + \frac{1}{2}u^2 + \lambda(x+u)$$

using the optimality condition

$$\frac{\partial H}{\partial u} = 0 \Rightarrow u + \lambda = 0 \rightarrow u^* = -\lambda(t) \tag{16}$$

$$\lambda' = -\frac{\partial H}{\partial x} = -(1+\lambda), \lambda(1) = 0 \tag{17}$$

$$x'(t) = x(t) + u(t) \Rightarrow x(t) - \lambda(t) \tag{18}$$

transforming (17) into integral equation gives

$$\lambda(t) = \lambda(0) + \int_0^t -(1+\lambda(s))ds \tag{19}$$

hence

$$\lambda(1)=\lambda(0)+\int_{0}^{1}-(1+\lambda(s))ds \tag{20}$$

Let $P=-\int_{0}^{1}(1-\lambda(s))ds$, using Simpson rule

$$\int_{a}^{b} f(x)dx=\frac{b-a}{6}\left\{f(a)+\frac{(b+a)}{2}+f(b)\right\} \tag{21}$$

then

$$\lambda(0)=1.7983$$

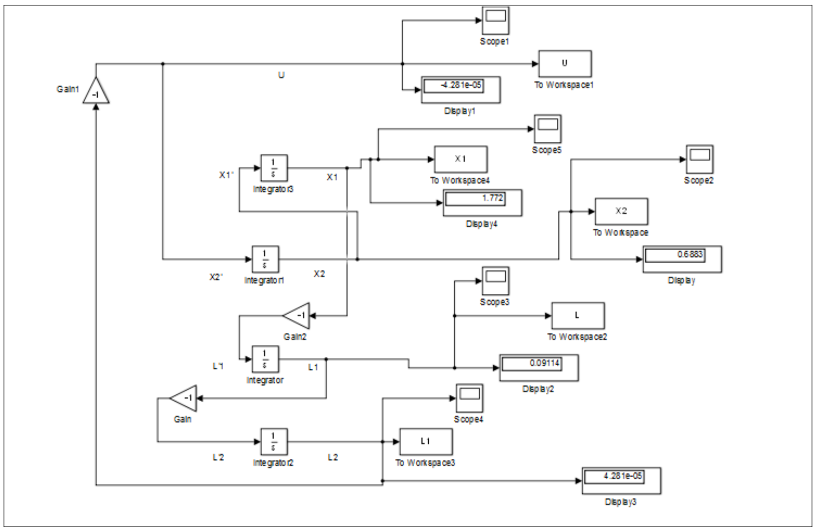

Figure 12: MATLAB simulink for problem 3

Blocks which are relevant to the example are gathered from the simulink blocks library, modified to suit to the desired model and joined with lines to indicate transfer of signals. The complete system is then run.

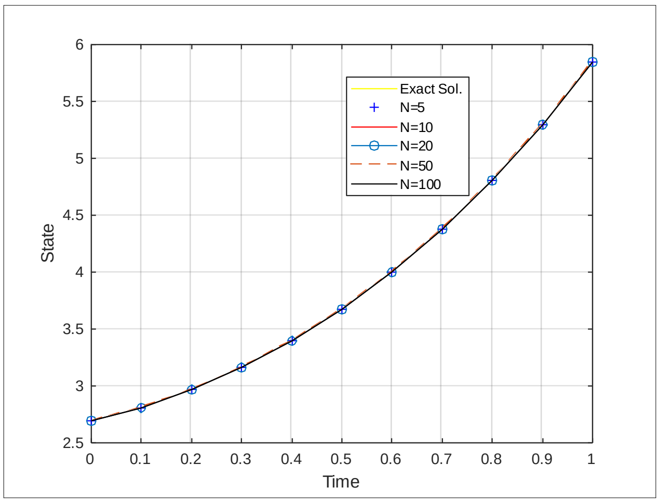

Figure 13: State result for problem 3

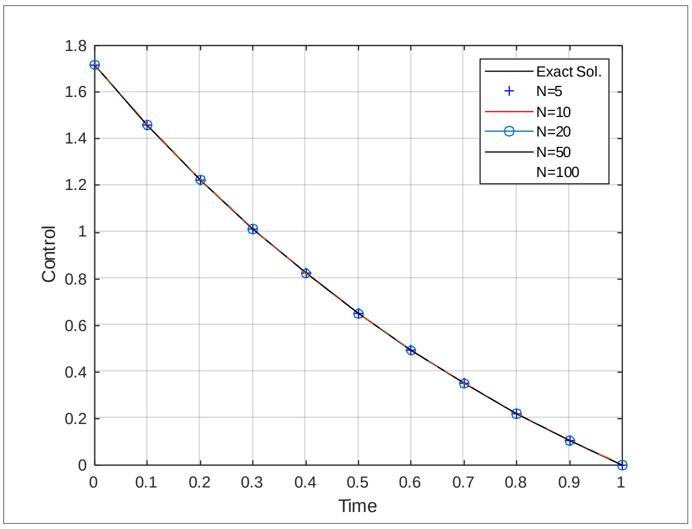

Figure 14: Control result for problem 3

Problem 4: *[9] consider the problem*

$$\min_u J = \frac{1}{2}\int_0^1 x_1^2(t) + u^2(t)dt$$

Subject to the initial condition

$$x_1'(t) = x_2(t), x(0) = 1$$

$$x_2'(t) = u(t), x_2(0) = 1$$

Solution: *The Hamiltonian*

$$H = \frac{1}{2}x_1^2 + \frac{1}{2}u^2 + \lambda_1 x_2 + \lambda_2 u$$

$$\frac{\partial H}{\partial u} = 0 = u + \lambda_2 \Rightarrow u = -\lambda_2 \qquad (22)$$

$$\lambda_1' = -\frac{\partial H}{\partial x_1} = -x_1, \lambda_1(1) = 0$$

$$i.e.\ \lambda_1 = -x_1, \lambda_1(1) = 0 \qquad (23)$$

$$\lambda_2' = -\frac{\partial H}{\partial x_2} = -\lambda_1, \lambda_2' = -\lambda_1, \lambda_2(1) = 0 \qquad (24)$$

$$\lambda_1(1) = \lambda_1(0) + \lambda_1'(0)$$

$$\lambda_1'(0) = -x_1(0)$$

$$0 = \lambda_1(0) - 1 \Rightarrow \lambda_1(0) = 1$$

Using (24)

$$\lambda_2(1) = \lambda_2(0) + \lambda_2'(0)$$

$$0 = \lambda_2(0) - \lambda_1(0)$$

$$\lambda_2(0) = 1$$

hence the equations are

$$u = -\lambda_2 \qquad (25)$$

$$\lambda_1'(t) = -x_1, \lambda_1(0) = 1$$

$$\lambda_2' 9t) = -\lambda_1, \lambda_2(0) = 1$$

$$x_1'(t) = x_2(t), x_1(0) = 1$$

$$x_2'(t) = u(t), x_2(0) = 1$$

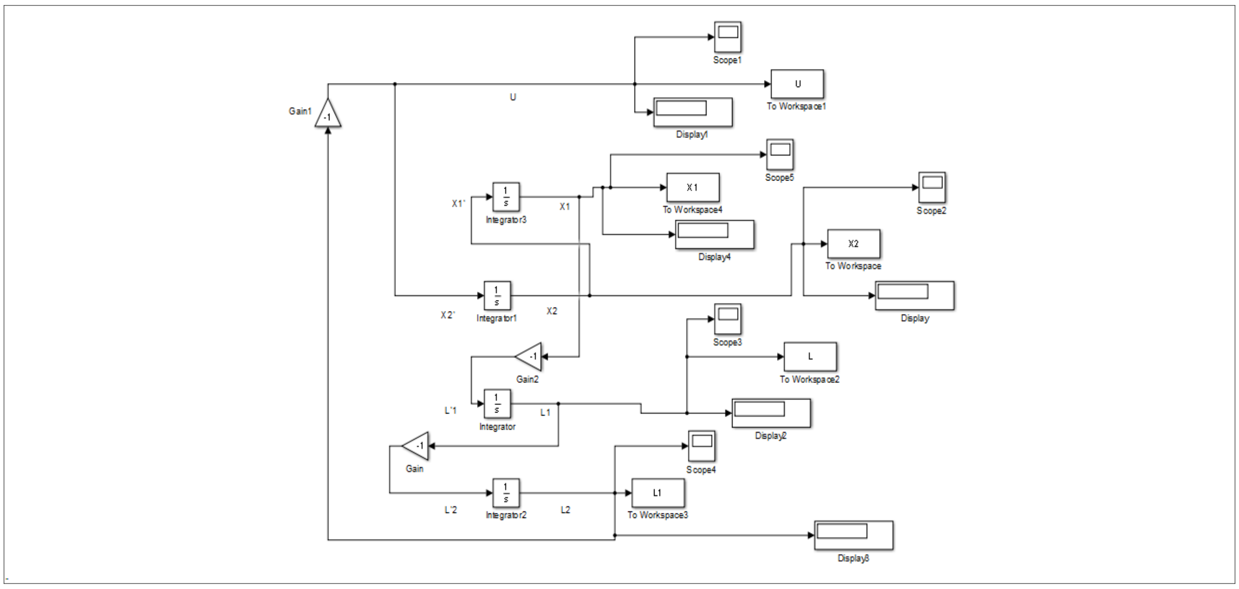

Figure 15: MATLAB simulink for problem 5

Blocks which are relevant to the example are gathered from the simulink blocks library, modified to suit to the desired model and joined with lines to indicate transfer of signals. The complete system is then run.

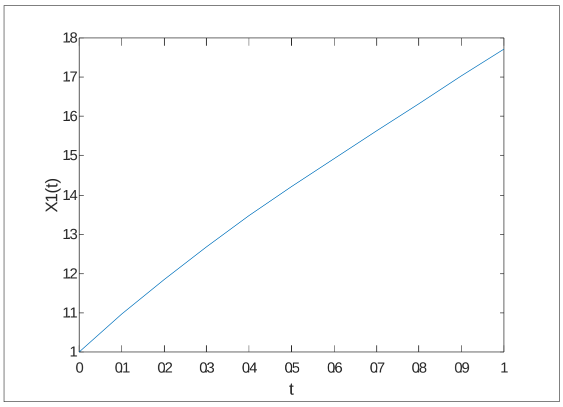

Figure 16: Result of first state for problem 4

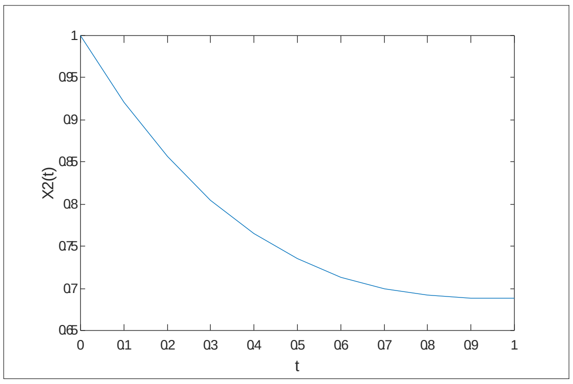

Figure 17: Result of second state for problem 4

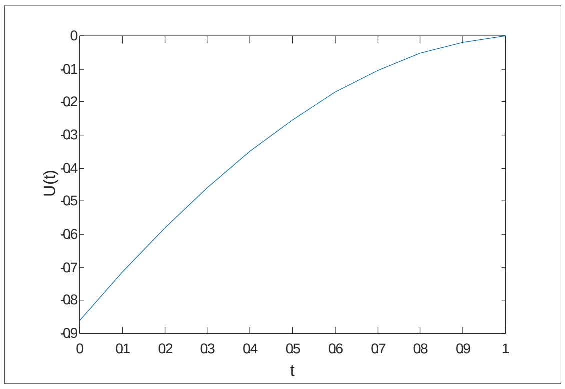

Figure 18: Result of control for problem 4

4. Discussion

Figures 1 to 5 depict the features of the Feed Forward Neural Network Architecture for the code used for the solution of all the Problems. Figures 1 to 3 describe the feed forward for layers 1 and 2 when the code was run. Figures 4 and 5 describe the interconnection among the weights, showing the movement in and out of the weights as shown by the neural networks from the first to the second layer.

Reducing the state is the goal in Problem 1. The MATLAB Simulink is displayed in Figure 6. It is built to generate the trial value, and the Feed Forward Algorithm is used to train and obtain results for Problem 1 by using Feed Forward Neural Networks, as indicated in Figures 7 and 8. When $N = 20$ or 100, the result is consistent and fairly near to the exact solution. The Feed Forward Neural Network Architecture produces a better outcome than [1, 5]. The results indicates that the outcome is superior to that of [1, 5] and likewise comes quite close to the exact solution. One may see that the graph displays zero at $\lambda(1) = 0$. This showed a positive comparison with the findings from [1, 5], suggesting that the outcome is accurate.

The MATLAB Simulink for Problem 2 is shown in Figure 9. Using Feed Forward Neural Networks and the Feed Forward Algorithm, the results of the Radial Basis Algorithm for the State and Trajectory for Problem 2 are displayed in Tables 3 and 4; Figures 10 and 11. For λ, it can be shown that the graph displays zero at λ, meaning that $\lambda(1) = 0$. Also when $N = 5, 10, 20, 50$ or $100,$ the results are consistent and fairly the same to 4 decimal places. These suggest that the outcome is accurate in approximation.

Problem 3, we convert $\lambda(1)$ to $\lambda(0)$ using the Simpson Rule for the integrating to obtain $\lambda(0)=1.7983$. With the Simpson Rule, it is not likely obtain a perfect answer, hence it was then simulated using MATLAB Simulink to generate the trial values. By running the Radial Basis Algorithm, we obtained the results of Radial Basis for the State and Trajectory for Problem 3 as shown in Figure 13 and 14. A cursory look at the Figures shows that, the values obtained when $N=5,10,20,50$ or 100, is consistent and fairly the same to 7 decimal places. This indicates that the result obtained are perfect.

Problem 4; Figure 15 displays the MATLAB Simulink for Problem 4. Figures 16, 17 and 18 shows the graphs for the Trajectory, State and Control respectively. For λ, it can be observed that at $\lambda(1)$, the graph shows zero i.e. $\lambda(1)\rightarrow 0$. This favourably compared with the graph obtained by the source of the problem 5, indicating that the result obtained is correct.

5. Conclusion

The optimality system in this work is constructed using Pontryaginí's principle. Next, the optimality system is used to build the simulink. The trial solution for the multilayer feed forward network known as the Radial Basis Neural Network is provided by a MATLAB 8.5 code that simulates the optimality condition using MATLAB Simulink. Through the use of some numerical examples, the method's effectiveness was determined.

References

[1] R. R. Garret, *Numerical Methods for Solving Optimal Control Problems*,Tennessee Research and Creative Exchange, University of Tennessee, Knoxville (2015). http:/trace.tennessee.edu/utk-gradthes/3401.

[2] A. K. Johan, P. Joos, L. R. D. Bart, Artficial Neural Networks for Modelling and control of Non-Linear systems, *Springer Science+Business MediaDordrecht* (1996) 83-88.

[3] S. Adamu, O. O. Aduroja, K. Bitrus, Numerical Solution to Optimal Control Problems using Collocation Method via Pontrygin's Principle, FUDMA Journal of Sciences 7(5) (2023) 228-233. DOI: https://doi.org/10.33003/fjs-2023-0705-2016.

[4] O. O. Aduroja, S. Adamu, M. Kida & H. L. Buhari (2024). Bi-Basis Function Second Derivative Block Method for the Solution of Optimal Control Problems. International Journal of Development Mathematics 1(1) 16 - 24. https://doi.org/10.62054/ijdm/0101.02.

[5] S. Adamu, Numerical Solution of Optimal Control Problems using Block Method, Electronic Journal of Mathematical Analysis and Applications 11(2) (2023) 1-12. http://ejmaa.journals.ekb.eg/.

[6] R. E. Bellman, The theory of dynamic programming, *Bull. Amer.Math. Soc.* 60(6) (1954) 503--515.

[7] S. Adamu, S., Aduroja, O. O., Onanaye, A. S. & Odekunle, M. R. (2024). Iterative method for the numerical solution of optimal control model for mosquito and insecticide. J. Nig. Soc. Phys. Sci. 6(2024) 1965. DOI: https://doi.org/10.46481/jnsps.2024.1965.

[8] M. T. Hagan, H. B. Demuth, M. H. Beak, *Neural network design*, pws publishing, Boston (1996).

[9] S. Lenhart, J. T. Workman, Optimal Control Applied to Biological Models, Chapman & Hall/CRC, Boca Raton (2007).

[10] S. Adamu, A. M. Alkali & M. R. Odekunle (2024). Runge-Kutta Like Method for the Solution of Optimal Control Model of Real Investment and Fish Management. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 15(2), 155-169.

DOI: https://doi.org/10.61841/turcomat.v15i2.14646..

[11] W. Zhang, Computational ecology. Artificial neural network and their applications, *World Scientific Publishing Co. pte. Ltd* (2014).

[12] O. Awodele, O. Jegede, Neural network and its application in engineering, *Proceedings of informing science & IT education conference* (2009). doi:10.28945/3317.

[13] W. E. Boyce, R. C. Diprima, *Elementary differential equations and boundary value problems*, John Wiley & sons, inc. Newyork (2011).

[14] A. Danney, A Software Safety Certification Plug-in for Automated Code Generators: Feasibility Study and Preliminary Design, NASA Ames Research Center, Muffet Field, CA 94035 (2011).

[15] M. J. D. Powell, Radial Basis Functions for Multivariable Interpolation: A Review (1987).

[16] D. S. Broom, D. Lowe, Radial basis functions, multi-variable functional interpolation and adaptive networks (No. RSRE-MEMO-4148), Royal Signals and Radar Establishment Malvern, United Kingdom (1988).

[17] H. Yu, T. Xie, S. Paszczynski, B. M. Wilamowski, Advantages of radial basis function networks for dynamic system design, IEEE Transactions on Industrial Electronics 58 (12) (2011) 5438 - 5450.

[18] R. L. Gyo, M. Kohler, A. Krzyzak, H. Walk, A Distribution Free Theory of Non-Parametric Regression, Springer Verlag, New York Inc (2002).

[19] S. Effati, M. Pakdaman, Optimal control problem via neural networks, *Neural Computer and Application* (2017). doi:10.1007/s00521-012-1156-2.

[20] M. T. Hagan, H. B. Demuth, M. H. Beak, *Neural network design*, pws publishing, Boston (1996).

[21] N. A. Khan, A. Shaikh, F. Sultan, A. Ara, Numerical simulaion using artificial neural network on fractional differential equations, Numerical simulation-From Brain imaging to Turbulent Flow (2016) 98-112. Doi:10.5772/64151.

[22] Y. H. Pao, M. Philips, D. J. Sobajic, Neural-net computing and the intelligent control of systems, International Journal of control 56(2) (1992) 263-289. Doi.org/10.1080/00207179208934315.

[23] L. Gongi, C. Liu, Y. Li & F. Yuan, Training Feed-forward Neural Networks Using the Gradient Descent Method with the Optimal Stepsize. Journal of Computational Information Systems 8(4) (2012) 1359{1371 http://www.Jofcis.com

[24] X. Zhou, W. Zhang, Z. Chen, S. Diao & T. Zhangy, Efficient Neural Network Training via Forward and Backward Propagation Sparsification. 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

[25] A. Kohan, E. A. Rietman & H. T. Siegelmann, Forward Signal Propagation Learning. IEEE Transactions on Neural Networks and Learning (2022).

[26] B. Yin, F. Corradi & S. M. Bohté, Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time. Nature Machine Intelligence, 5(May) (2023) 518--527. https://doi.org/10.1038/s42256-023-00650-4.

[27] Y. Neha, Y. Anupam, K. Manoj, An Introduction to Neural Network Methods for Differential Equations, New York. *Springer Dordrecht Heidelberg* (2015) 24 -- 30.

[28] K. L. Du, M. N. S. Swamy, Radial Basis Function Networks, Neural Networks in a Softcomputing Framework, Springer, London (2006) 251 - 294.