WWJMRD 2025; 11(06): 32-43
www.wwjmrd.com
International Journal
Peer Reviewed Journal
Refereed Journal
Indexed Journal
Impact Factor SJIF 2017: 5.182 2018: 5.51, (ISI) 2020-2021: 1.361
E-ISSN: 2454-6615

Ezekiel Nnamere Aneke
Department of Electrical & Electronics Engineering, State University of Medical and Applied Sciences Igbo-Enu, Enugu, Nigeria.

Nnamdi Ahuchaogu
Department of Electrical Electronic Engineering, Abia State University, Uturu, Nigeria.

Correspondence:
Ezekiel Nnamere Aneke
Department of Electrical & Electronics Engineering, State University of Medical and Applied Sciences Igbo-Enu, Enugu, Nigeria.

Design and Simulation of an Artificial Neural Network-Based Controller for DC Motor Speed Regulation: A Comparative Analysis with PID Control

Ezekiel Nnamere Aneke, Nnamdi Ahuchaogu

Abstract
This study investigates the application of Artificial Neural Networks (ANNs) for the speed control of a DC motor. It examines key aspects of ANN design, including learning mechanisms, training procedures, and model development. To evaluate performance, both an ANN-based controller and a conventional Proportional-Integral-Derivative (PID) controller were designed and simulated. A comparative analysis of the simulation results revealed that the ANN controller offered superior stability and accuracy. Specifically, the ANN controller achieved a significantly lower speed control error of 0.242%, compared to the PID controller's deviation of 1.435%. These findings highlight the effectiveness of the ANN model in enhancing control precision and demonstrate its advantages over the traditional PID approach. The ANN model was trained using the Levenberg–Marquardt algorithm, which ensured efficient convergence and robust performance. Simulations were carried out in MATLAB/Simulink, enabling a detailed comparison of the controllers' dynamic responses. This work underscores the potential of ANN-based intelligent control strategies in industrial automation and motor drive applications.

Keywords: Artificial Intelligent, Neural Network, Proportional Integral Derivative, Simulation, Network Topologies.

1.0 Introduction
Artificial Neural Networks (ANNs) have emerged as a powerful tool in modern engineering applications, especially in areas requiring intelligent decision-making and control. Neural network research has significantly influenced multiple fields including control systems, speech recognition, medicine, image processing, and robotics (Mehr & Richfield, 1987; Schmidhuber, 2015). As computational models inspired by the structure and function of the human brain, ANNs simulate the way biological neurons process and transmit information through interconnected layers (LeCun, Bengio, & Hinton, 2015).
Artificial intelligence (AI) seeks to replicate human intelligence in machines, enabling them to solve complex problems more adaptively and efficiently (Wajeeha et al., 2022). One of the core subsets of AI, neural networks, focuses on pattern recognition, learning, and generalization from data. These networks function by converting real-world information—such as sound, images, or text—into numerical data that can be analyzed through supervised, unsupervised, or reinforcement learning (Mano, 2014; Goodfellow, Bengio, & Courville, 2016).
In control system engineering, ANNs are gaining traction due to their capacity for non-linear system modeling and adaptive learning. Traditional controllers like the Proportional-Integral-Derivative (PID) controllers, while reliable, often struggle to adapt to the dynamic and non-linear nature of many real-world systems (Åström & Hägglund, 2006). Conversely, ANN-based controllers can learn system behavior through training and improve control performance even when exact system models are unavailable (Lewis et al., 1999; Narendra & Parthasarathy, 1990).
According to Caudill and Butler (1989), neural networks can process information by their dynamic state response to external input, making them well-suited for real-time control tasks.

Their topology, which defines how neurons are connected, plays a critical role in determining network functionality and performance. Common ANN architectures include feedforward, recurrent, and convolutional topologies, each adapted to specific problem types (Haykin, 2009).

Modeling a neuron mathematically involves weighted inputs, bias terms, and a non-linear activation function, typically chosen from step, linear, or sigmoid functions depending on the task (Lewis et al., 1999). While a single neuron is limited in function, networks of interconnected neurons—ANNs—can perform complex computations such as classification, regression, and time-series prediction (Kenji, 2011).

In this context, this study investigates the application of an ANN-based controller for the speed regulation of a DC motor, comparing its performance with that of a conventional PID controller. The research aims to highlight the potential of ANN to enhance system stability and control precision in industrial motor applications.

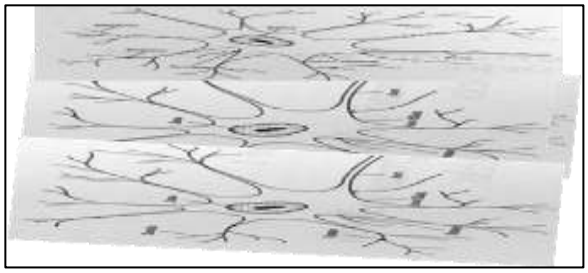

Fig. 1.0: Components of a neuronal cell.

The basic component of brain circuitry is a specialized cell called the neuron, which consist the cell body made up of dendrites and axon the biological neuronal cell system has its components, the components are shown in Figure1(Eduardo, 1992). The components are made up of the dendrites which gets signals from other neurons into the body cell or soma. (Lewis. and. Ildirek 1999). Most often the dendrites multiply each input signal by a transfer weighting coefficient. In the soma, cell capacitance separates the signals which are collected in the axon. When the composite signal exceeds a cell threshold signal, the action is transmitted through the axon. From research it was noticed that cell nonlinearities make the composite action potential a nonlinear function of the combination of arriving signals. Axon which is another component connects through synapses with the dendrites of subsequent neurons and synapses operate through the discharge of neurotransmitter chemicals across intercellular gaps, also it can be either excitatory (tending to fire the next neuron) or inhibitory (tending to prevent firing of the next neuron) (Lewis. and. Ildirek 1999)

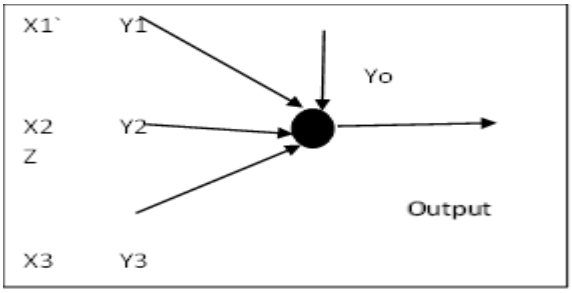

Fig. 2.0: Mathematical Model Representing a Neuron.

A Neuron can also be model mathematically and this can be shown in Figure 2.0. The figure consists of the dendrite which weight is yj , the firing threshold yo which is also called the 'bias', the summing weight incoming signals, and the nonlinear function σ(·). The cell inputs are the n time signals x1(t), x2(t), . . . xn(t) and the output is the scalar y(t), which can be expressed as

$$Z(t) = \left(\sum_{i=1}^{n} yj\, xj\,(t) + Yo\right) \text{ ----------(1)}$$

Equation 1 shows the positive weights *yj* correspond to excitatory synapses and negative weights to inhibitory synapses (Lewis. and. Ildirek 1999).

To gain a comprehensive grasp of the mathematical intricacies achievable through the interconnection of individual artificial neurons, it is advisable to steer clear of random interconnections. Random interconnections can result in an excessively complex system that becomes unmanageable. Early researchers have devised standardized topologies for artificial neural networks. These predefined topologies facilitate more straightforward, quicker, and efficient problem-solving. Various artificial neural network topologies are tailored for solving specific problem types. Once the problem type is identified, the choice of the artificial neural network's topology becomes crucial for the system.

The topology and its parameters need to be modified to solve the problem. Modifying topology of artificial neural network does not mean that we cannot use the artificial neural network, it is only a precondition. Before artificial neural network can be used, we need to teach it or train it on solving that type or particular problem. The artificial neural networks use the inputs they have just as biological neural networks to learn the behavior/responses from their environment or surroundings.

Research has demonstrated that artificial neural networks utilize three primary learning methods: supervised learning, unsupervised learning, and reinforcement learning. The choice of learning method parallels the selection of the neural network's topology and depends on the nature of the problem at hand. Each learning method operates based on distinct principles, yet they share common elements, namely, learning data and learning rules. These two parameters play a critical role in determining the network's functions and associated costs. The fundamental objective of an artificial neural network is to generate appropriate output responses based on given input data. In summary, the process involves the selection of an optimal topology, potential modifications to it, and the subsequent choice of an appropriate learning method as prerequisites to effectively utilizing artificial neural networks for problem-solving Artificial neural networks has been in existence for many years and have been can find working in many areas such as process control, chemistry, gaming, radar systems, automotive industry, space industry, astronomy, genetics, banking, fraud detection, etc. and solving of problems like function approximation, regression analysis, time series prediction, classification, pattern recognition, decision making, data processing, filtering, clustering, etc (Kenji 2011).

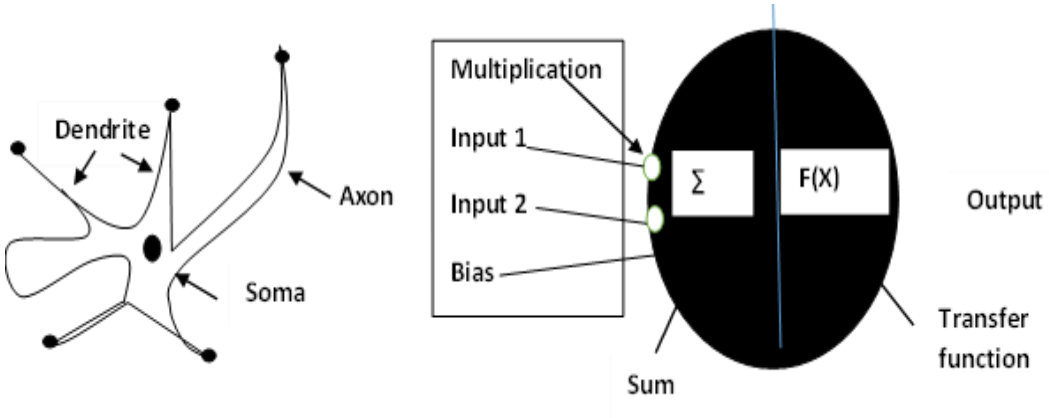

Fig. 3.0: Artificial and Biological Neural.

Figure 3.0. depicts the designs of a biological neuron and an artificial neuron. In the biological neuron, information enters through the dendrite, is processed within the soma, and then transmitted through the axon. However, in the case of an artificial neuron, information enters the neuron's body through weighted inputs (each input is individually multiplied by a weight). The artificial neuron's body is responsible for summing the weighted inputs, incorporating a bias, and processing this sum using a transfer function. Finally, the artificial neuron conveys the processed information through its output(s). The important of artificial neuron model is the simplicity, which can be seen in its mathematical description (F.I. Lewis, S. J and A. Ildirek 1999]

Yi (k) F E wi(k) . xi (k) + b (2)

Where: Xi (k) is input value in discrete time K and t goes from 0 to m; Wi is weight value in discrete time k where t goes from 0 to m, b is bias, F is a transfer function and Yi(k) is output value in discrete time k. the model of an artificial neuron and its equation has unknown variable, the major unknown variable of the model is its transfer function. The transfer function describes the properties of artificial neuron and it can be a mathematical function. We choose it on the basis of problem that artificial neuron (artificial neural network) needs to solve and in most cases, we choose it from the following set of functions: Step function, Linear function and Non-linear (Sigmoid) function. Step function normally should be a binary function that has only two output values, which must be zero and one. That means if input value meets specific threshold the output value results in one value and if specific threshold is not meet that results in different output value. The threshold can be described using equation (3).

$$\begin{cases} 1 \text{ if } f\, wi\, X_i > \text{threshold} \\ 0 \text{ if } wi\, X_t < \text{thresholds} \end{cases} \quad (3)$$

Equation 3 illustrates the artificial neuron perceptron and the type of transfer function employed within artificial neurons. Perceptron are primarily applied in solving classification problems and are typically located in the output layer of artificial neural networks. Regarding the role of the linear transfer function within an artificial neuron, it serves to transform the sum of weighted inputs and bias. Such artificial neurons, in contrast to perceptions, are frequently situated in the input layer of artificial neural networks. The most prevalent nonlinear function used in this context is the sigmoid function.An artificial neural network is produced when two or more artificial neuron are combined. One artificial neuron is not enough or effective in solving real-life problems, but combination of may artificial neuron which forms artificial neural networks have the capacity to solving real life time problems. The artificial neural networks have the ability of solving complicated problems. They solve problems making use of the building blocks which they use in processing information or data in a non-linear, distributed, parallel and local manner.

2.0 Materials and Methods

2.1 Neural Network Learning

The learning process in a neural network involves six key stages, each contributing to a researcher's understanding of how it operates. These stages can be summarized as Initialization, in this stage, all neurons are assigned initial weights. Forward Propagation, here training set inputs are passed through the neural network, and the output is computed. Error Calculation. Since we work with a known training set, the correct output is available. Therefore, an error function can be defined by measuring the difference between the model's output and the actual result. Back Propagation, the goal in this stage is to minimize the error function with respect to the neurons. Weight Update: The optimal weights are adjusted to affect the back propagation and improve the results, often using algorithms like Levenberg Marquardt. Iterate Until Convergence. the artificial neural network (ANN) is trained here, through repeated iterations until it accurately and adequately learns the underlying rule, enabling it to replicate this knowledge when required. Back propagation networks acquire the ability to classify and generalize patterns. (Mano, 2014) The connections between the artificial neurons change as a pattern appears until they provide the right response. It's important to note that minimizing the error function is equivalent to optimizing convergence. These stages help illustrate the general process of training in a neural network.

Table 1.0: Stages and learning process in Neural Network.

S/N	Stages	Description
1	Initialization	Neurons were identified with initial weights at this stage
2	Forward Propagation	Data from the input are processed here which will eventually computed and captured at the output.
3	Error Calculation	The error is calculated at this point by computing the difference between the output and the desired output model
4	Back Propagation	Minimization of error is done here by adjusting the weights
5	Weight Update	Weights update is done to improve convergence. Levenberg-Marquardt can be used to do weight update
6	Iterate Until Convergence	Learning is done here, ANN learns the rule accurately when the process is repeated iteratively

Table 1.0 summarized the learning process. Table 2 demonstrates the process where the input is twice the desired output

Table 2.0: LM propagation applied in the forward direction and initialized.

S/N	Input	Desired Output
1	0	0
2	2	4
3	3	6
4	4	8
5	5	10

Let the weight value be w = 3, then the mode output becomes

Table 3.0: Forward training showing the error and w.

S/N	Input	Desired Output	Model Output (w = 3)
1	0	0	0
2	2	4	6
3	3	6	9
4	4	8	12
5	5	10	15

Table 4.0: Increased error and further increase of the value of w.

S/N	Input	Desired output	Model output w =3	Absolute error	Square error
1	0	0	0	0	0
2	2	4	6	2	4
3	3	6	9	4	8
4	4	8	12	3	6
5	5	10	15	1	1

Table 5.0: A backward training in LM with decrease value of w.

S/N	Input	Desired output	Model output w =3	Absolute error	Square error
1	0	0	0	0	0
2	2	4	6	1	1
3	3	6	9	3	6
4	4	8	12	2	4
5	5	10	15	0	0

Increasing the value of w increases the error, thus we stop increasing the value of w further.
Suppose the value of w is now decreased

Table 6.0: Reduction in the error value.

Input	Desired output	Model output w =3	Absolute error	Square error	Model output (w=2)	Absolute error	Absolute error
0	0	0	0	0	0	0	0
2	2	4	1	1	2	0	0
2	4	6	4	4	4	0	0
3	6	9	6	6	6	0	

Tables (1.0 to 6.0) provided illustrate that both the absolute error and squared error decrease as the weight (w) decreases. The following steps were carried out in the previous illustration:

Weight Initialization: The weight (w) was initialized, and LM (Levenberg-Marquardt) propagation was applied in the

forward direction. Initial Error: Some errors were noticed during the forward training. Weight Adjustment: To address the errors, the weight (w) was increased, and forward training was continued. Increased Error: Unfortunately, this led to an increase in the error.

Backward Training: Backward training using the LM algorithm was applied with a decrease in the value of w. Reduced Error: This resulted in a reduction in the error value. The primary objective was to find the optimal weight value that minimizes the error. Once this state is reached and convergence of the neurons is achieved, the training process is halted. It's worth noting that the Levenberg-Marquardt training algorithm incorporates elements of the Newton algorithm. Levenberg–Marquardt, (LM), optimization training method is normally used to carry out the training in ANN.

There are some other training algorithms such as the steepest descent algorithm, and the Gauss Newton algorithm. Even though the EBP algorithm is still widely in use, but the algorithm has low convergence.

Levenberg–Marquardt Training

The LM algorithm is adjudged to be much faster than other algorithms, this is because the size of the multi– layer perceptron (MLP) is not very large,

$$H \approx J^T J + \mu I \quad 4$$

Is always positive, combination coefficient and I is identity matrix

With this approximation in equation 2.1, the assurance that matrix H is always invertible is guaranteed. Square error (SSE) is usually defined to evaluate the training process. To calculate this quantity, for all training patterns and network outputs, the following relation is used:

$$E(x,w) = \tfrac{1}{2} \sum_{p=1}^{p} \sum_{m=1}^{m} e_{p,}{}^{2} m \quad 5$$

Where

x is the required input

w is the highest vector

$e_{p,}$ m is the training error at output m, when applying pattern p and defined as

$$e_{p,m} = d_{p,m} - O_{p,m} \quad 6$$

Table 7.0: Summary of the updates for various algorithms.

S/N	Algorithm	Rules used	Convergence status	Computation of the algorithm
1	EBP algorithm	$w_{k+1} = w_k - \alpha g_{ki}$	Stable, Slow	Gradient
2	Newton algorithm	$w_{k+1} = w_k - \alpha g_k$	Unstable/fast	Gradient/Hessian
3	Gauss-Newton alg.	$w_{k+1}=w_k-(J_k^T J_k)^{-1} J_k e_k$	Unstable, fast	Jacobian
4	LM algorithm	$w_{k+1} = w_k – (J_k^T J_k)^{-1} J_k e_k$	Stable, fast	Jacobian

Table 7.0 summarizes the performance of different algorithms with respect to their stability

To implement the Levenberg–Marquardt algorithm for training neural network, two problems need to be solved namely: organizing the training process iteratively for weight updating, and calculating the Jacobian matrix. These two problems are addressed shortly. In this study, since parameters for training were extracted from results obtained from the optimization results, the major work will be cantered on how to organize the training process iteratively for weight updating. This is because the calculation of the Jacobian matrix was part of the optimization calculation.

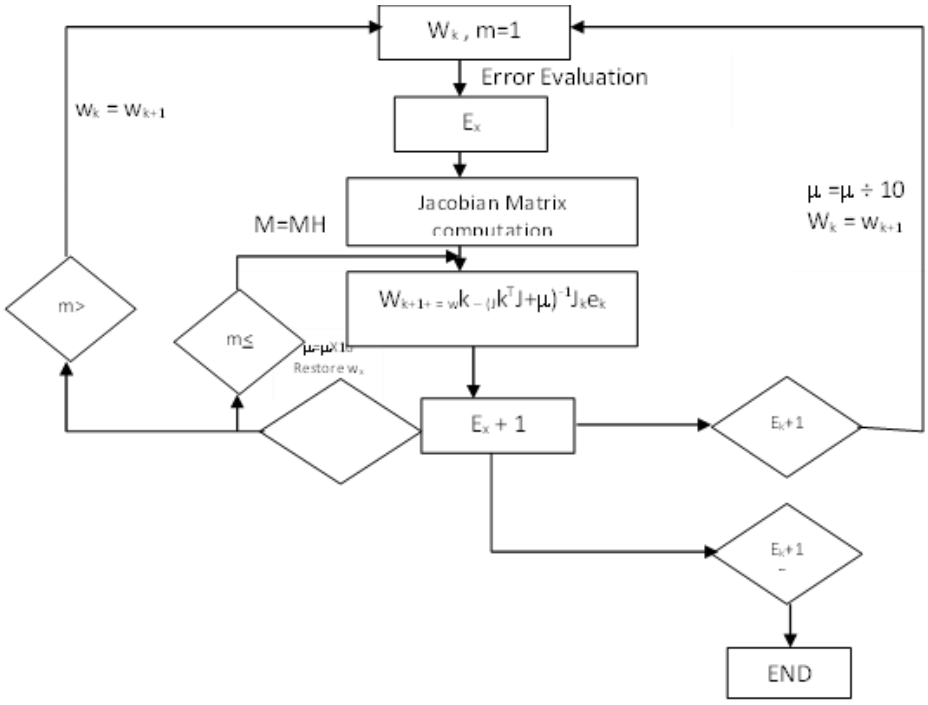

Fig. 4.0 is a flow chart for the training using Levenberg – Marquardt algorithm.

Figure 4.0: Flow Chart for the training process using Levenber – Marquardt algorithm

W_x is the current weight, w_{k+1} is the next weight, E_{k+1} is the current total error, while E_k is the last total error. From the algorithm, using Levenberg-Marquardt algorithm, the training process was designed following the algorithm: Initial weights to be generated randomly and evaluation of the total error (SSE)

Applying the equation $w_{k+1} = w_k (J_k^T J_k + \mu I)^{-1} J_k e_k$ 7

An update was carried out to update and adjust weights. Evaluation of the total error with addition of the new weights. Increasing the total current error due to the update requires thereafter, a repeat of step 2 is made for the update again. The Levenberg-Marquardt training algorithm gives a better result because it solves the problems existing in both the gradient descent (or EBP) method and Gauss – Newton method for neural network training.

3.0 Discussion and Results

3.1 Artificial Neural Network model controller

The focus here is using the application of artificial neural network (ANN) in controlling DC motor.

In the development of the control system, for instance using ANN inverse model controller of dc motor is shown in the block diagram of Figure 5.0

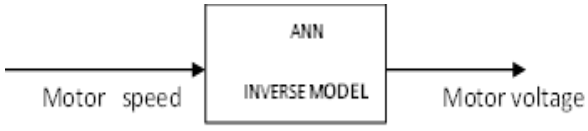

Fig. 5.0: Block diagram of ANN inverse model.

The inverse model when given a particular motor speed generate at the output a corresponding voltage produces a speed. The inverse model as the name implies uses the inverse model of the dc motor. The dc motor takes voltage as input to give speed at the output but the inverse model takes speed as input and generates voltage at its output. In this case, speed at three consecutive instants of time for a particular trajectory are presented to the inputs while the inverse model produces at the output, the voltage that is required at the motor's inputs to generate the particular target reference speed.

The ANN inverse model structure

The ANN inverse model of the dc motor can be described as a three-input single output structure with three speeds taken at three-time instants of n, n+1, and n-1. These three speeds serve as the inputs while the motor terminal voltage serves as the output. The structure of the ANN inverse model of the dc motor is shown in **Figure 6.0**.

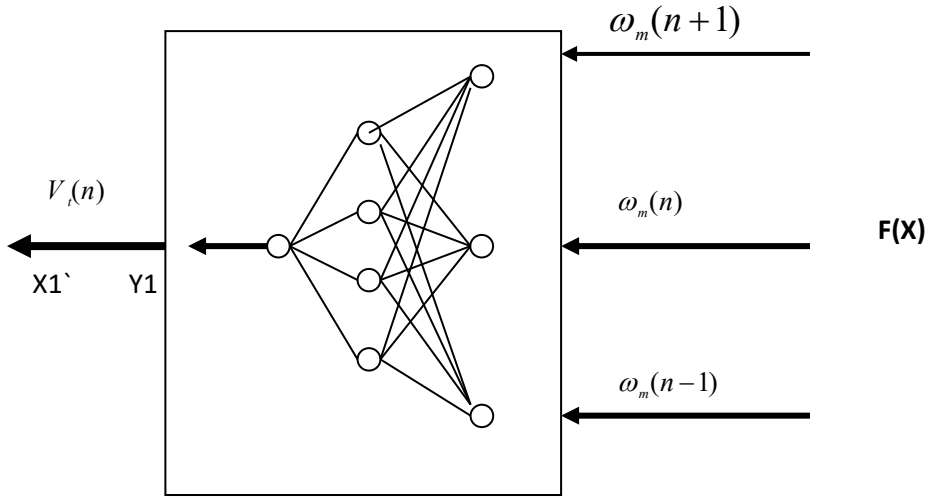

Fig. 6.0: ANN inverse model structure.

$$\omega_m(n+1),\ \omega_m(n)\ \ and\ \ \omega_m(n-1)$$

The values of , are seen and taken as the independent inputs of the ANN inverse model. It is pertinent to state here that the ANN inverse model can be trained to provide the terminal voltage for any DC motor with unknown parameters. The Artificial Neural Network Inverse Model (ANNIM) of the dc motor is achieved through simulation. Simulation is the imitation of the operation of real-world process or system over time. Simulation involves the generation of an artificial history of a system, and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system that is represented.

Simulation is an indispensable problem-solving methodology for the solution of many real-world problems. Simulation is used to describe and analyse the behaviour of a system, ask what if questions about the real system, and aid in the design of real systems. Both existing and conceptual systems can be modelled with simulation. The network data used in the training of this network is obtained from the physical reading taken from the developed dc servomotor system. The data is a two-dimensional data. One is voltage and time while the other is speed and time. The network parameters used in the neural network controller network coding are shown below:

3.1 Building Artificial Neural Network (ANN) Model

The following steps are used in the building of ANN model.

Step 1: At the input layer, the number of inputs=number of input neurons.

Step 2: At the output layer, the number of outputs=number of output neurons.

Step 3: At the hidden layer, the number of neurons and layers are not fixed and may take any number more than zero to map any complex functions.

Step 4: Assign weights

Step 5: Decide activation function. Here logistic function is taken for all neurons.

$$f(x) = \frac{e^x}{1+e^x} \quad 8$$

Step 6: Select appropriate training pattern, that is, input-output pairs.

Inputs Outputs

X_1 X_2 X_3 Y

Step 7: Training of ANN model; Here, the ANN output is calculated and compared with the desired output to determine the error E (desired output-actual output). Finally, minimize this error using some optimization technique. The sum-squared error may be written as

$$E = \sum_{i=1}^{n} (Y_i - V_i)^2 \quad 9$$

The weights of the network have to be updated for the error to be minimized. This process is known as training of the neural network. In the training of the artificial neural network, the error is fed back to the network to update the weights. This will complete one cycle, the complete cycle is performed many times till the predicted error is reduced. The complete stage is called epoch.

Back propagation

Weight adjustment is done by the method of back propagation. The total prediction error E, is also a function of W

.

$$\sum(W) = \sum[Y_i - V_i(W)]^2 \quad 10$$

Training Algorithm

The training algorithm has two process of information flow given, as back propagation and feed forward. Decide the network architecture (Hidden layers, neurons in each hidden layer)

Decide the learning parameter and momentum. Initialize the network with random weights

Do till convergence criterion is met, for i=1 to # training data points. Feed forward the i-th observation through the net, Compute the prediction error on i-th observation, Back propagate the error and adjust weights, Next, I, check for convergence, End Do, When the global minima are reached, the network training has to stop. Practically, if the decrease in total prediction error since the last cycle is small or if the overall changes in the weights (since last cycle) are small. The training data is partitioned into training set and validation set. Training set to build the model, and validation set to test the performance of the model on unseen data.

Simulation Model

The Matlab Simulink was used in modelling and building of the model network that behaves exactly as the system. It was from this model that the input/output data pairs to train the model network is generated. Here, Artificial Neural Network Inverse Model (ANN) of DC motor was used. For the speed control, Artificial Neural Network inverse model was used. Block diagram of **Figure 7.0** represents the system, and the designed model.

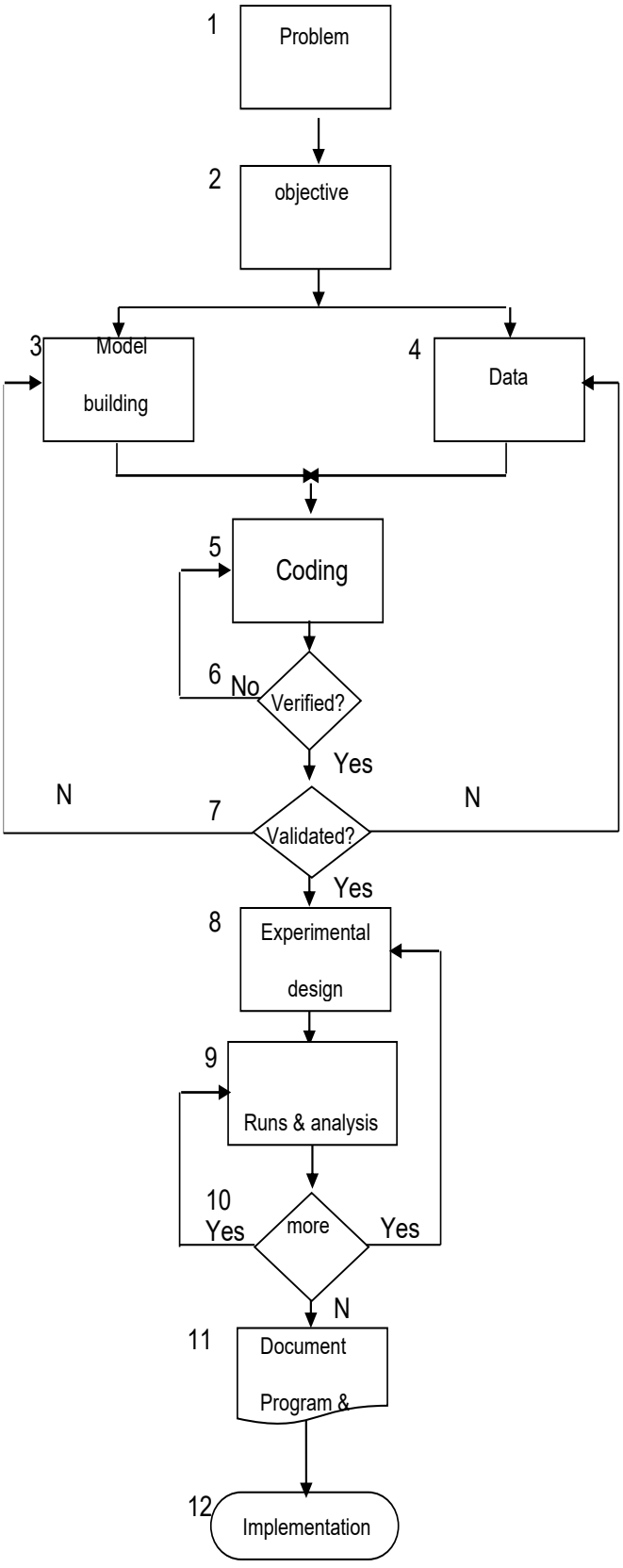

Fig. 6: Steps in simulation of ANNIM.

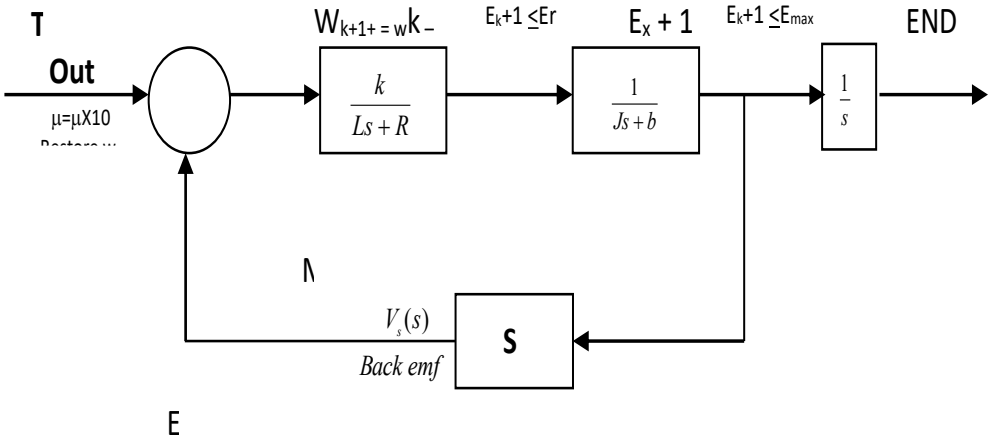

Fig. 7.0: Block diagram for speed of DC motor.

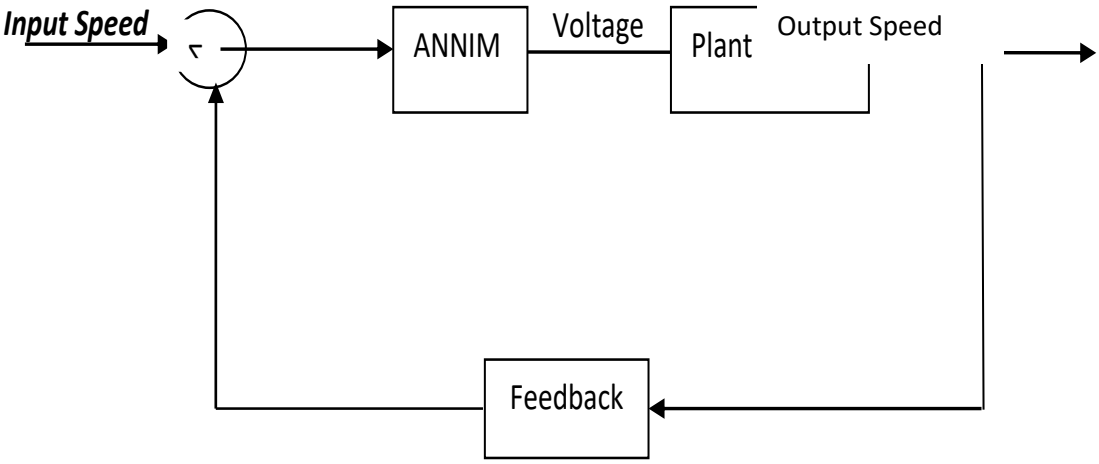

Fig. 8.0 Block diagram for Simulink model.

3.2: Presentation of Results

3.3: Speed control using ANN and PID controllers in Simulink

The speed control using artificial neural network and proportional integral derivative controllers were carried out to see the performance of the two controllers, and know the best controller.

Different models were presented, namely DC motor model, Artificial Neural Network (ANN) model and Proportional Integral Derivative (PID) model. The simulation of the models are shown in Figure 8.0. Models for ANN and PID are shown in figure 9.0 and 10.0 respectively.

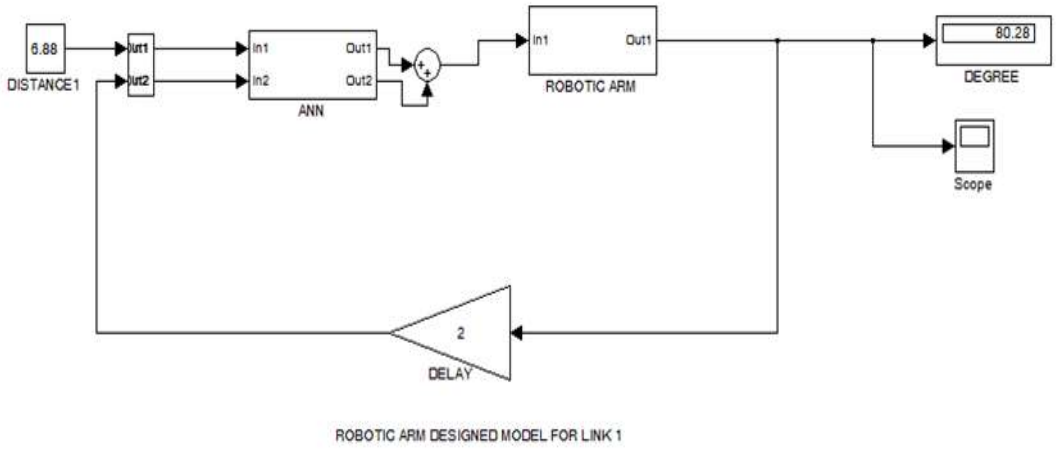

Fig. 9.0: Artificial Neural Network Model for speed control.

Proportional Integral Derivative PID controller was developed. PID controller is a conventional controller used when the system requires improvement under steady state transient condition, (Sukka et al, 2022). The conventional controller was used to compare its performance with the artificial neural network controller performance. The simulation graph for the two controllers was shown in Figures (12.0 and 14.0) respectively.

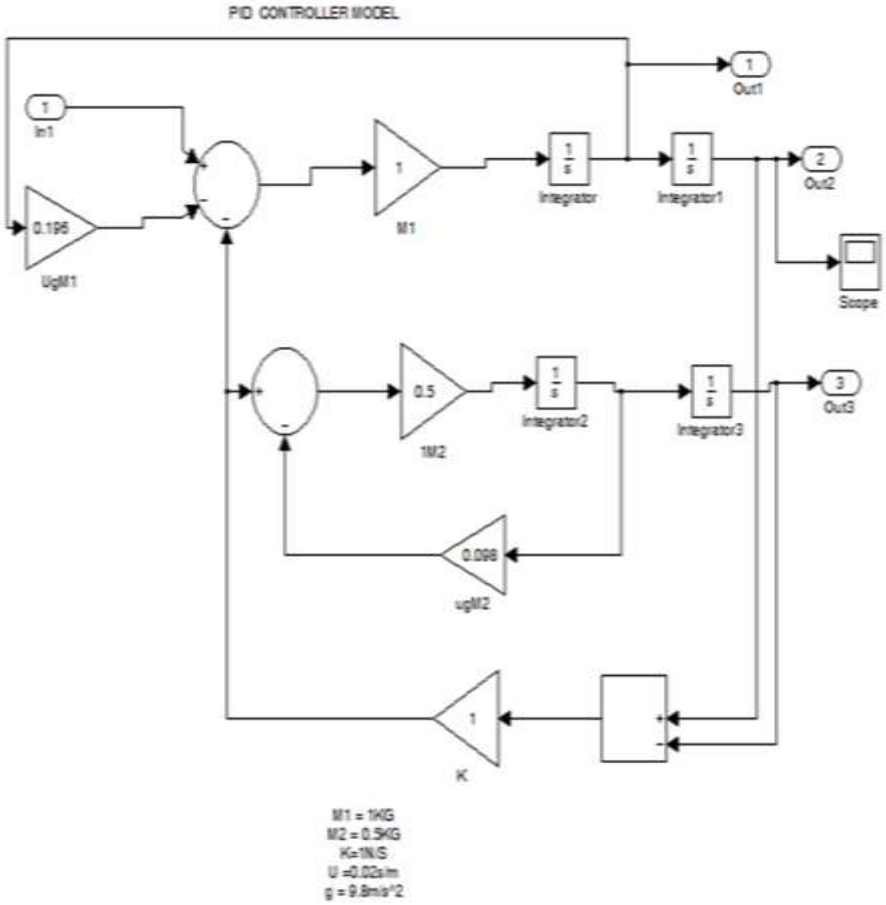

Fig. 10.0: Designed Proportional Integral Derivative (PID) model.

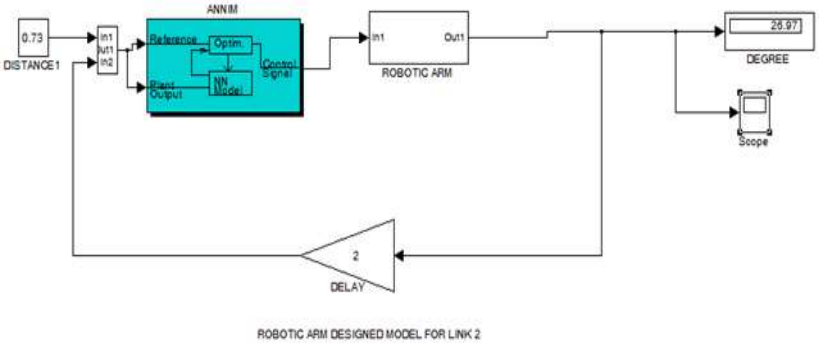

Fig. 11.0: Simulated Artificial Neural Network (ANN) model.

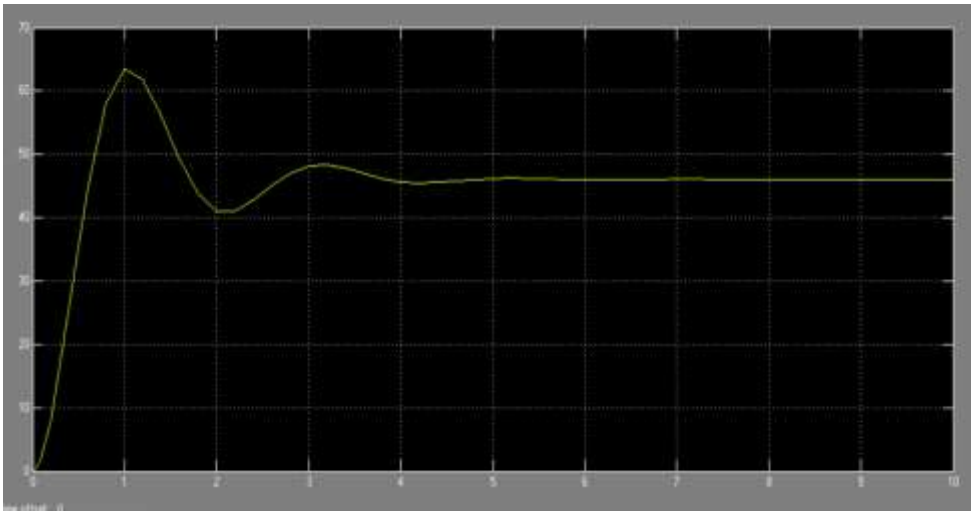

Fig. 12.0: Simulated graph of ANN model.

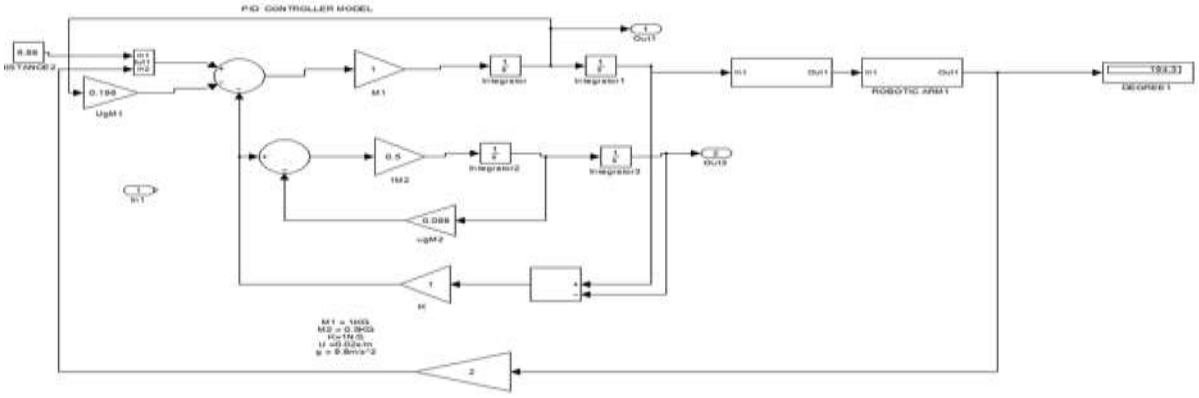

Fig. 13.0: PID Controller Model.

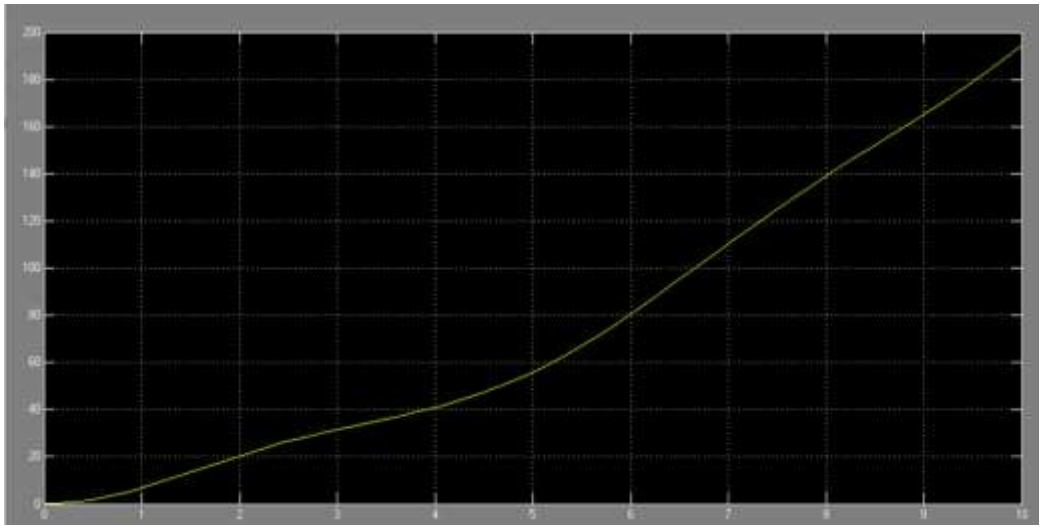

Fig. 14.0: Simulated PID controller model.

From the simulation graphs of the proposed models, it was noticed that the ANN model is more stable than the PID model.

Table 8.0: DC Motor Voltage vs Speed (Designed, ANN, PID).

S/N	Voltage of DC motor (Volt)	Speed (designed model) (m/s)	Speed (ANN Controller) (m/s)	Speed (PID controller) (m/s)
1	0.9	202.0	202.2	207.2
2	1.0	226.8	230.3	234.7
3	1.1	254.8	258.3	263.2
4	1.2	282.8	286.3	291.7
5	1.3	310.9	314.3	320.2
6	1.4	338.9	342.4	348.7
7	1.5	366.9	370.4	377.2
8	1.6	395.0	398.4	405.7
9	1.7	423.0	426.4	434.2
10	1.8	451.0	454.4	462.7

Table 9.0: Speed and Percentage error for ANN and PID Controllers.

S/N	Speed from DC Motor model (m/s)	Speed from ANN model controller output (m/s)	Speed from PID model controller (m/s)	Percentage error in ANN controller (%)	Percentage error in PID controller (%)
1	202.1	202.5	205.9	0.20	1.88
2	273.7	274.5	278.8	0.29	1.87
3	287.2	288.0	290.3	0.28	1.08
4	291.9	292.5	295.9	0.21	1.37
5	298.6	299.3	303.5	0.23	1.64
6	305.3	306.0	309.9	0.23	1.51
7	202.1	202.5	205.9	0.20	1.88
8	273.7	274.5	278.8	0.29	1.87
9	287.2	288.0	290.3	0.28	1.08
10	291.9	292.5	295.9	0.21	1.37

Percentage Error for Artificial Neural Network (ANN) and Proportional Integral Derivative (PID) controllers

The average % error for Artificial Neural Network (ANN) and Proportional Integral Derivative (PID) controllers was calculated from Table 9.0 as follows:

$$(Average\ \%\ Error\ for\ ANN) = \frac{2.42}{10} = 0.242$$

$$(Average\ \%\ Error\ for\ PID) = \frac{14.35}{10} = 1.435$$

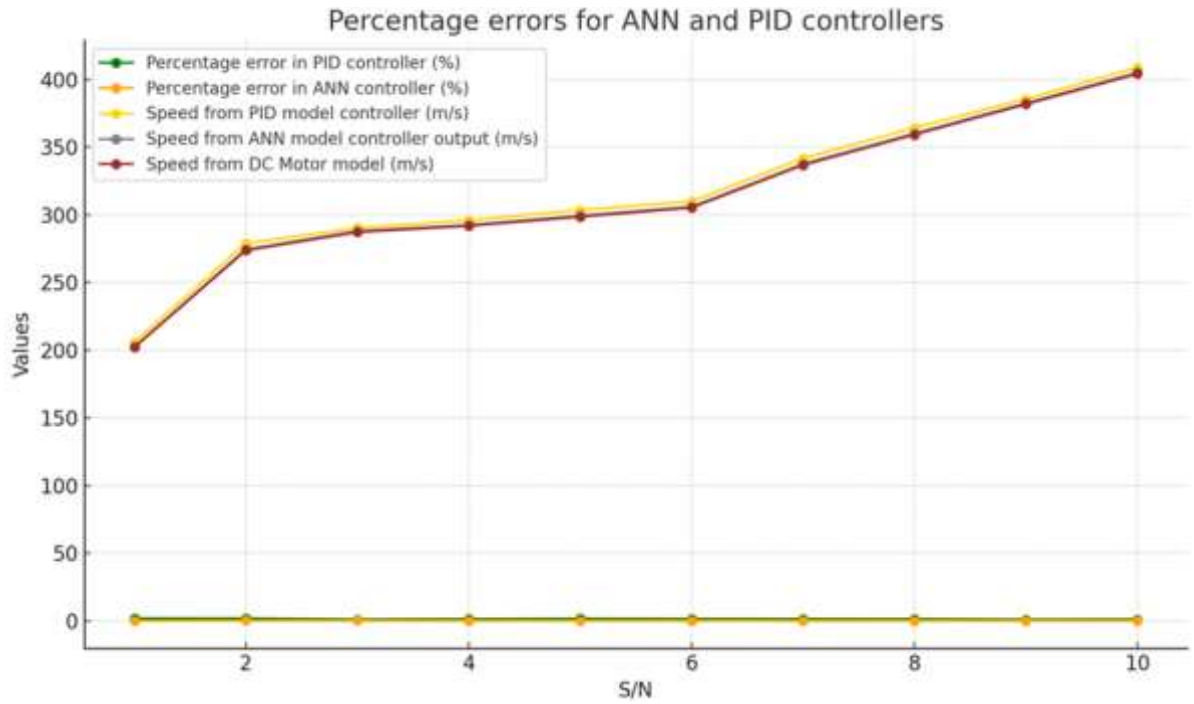

Figure 15.0: Analysis on Percentage error for ANN and PID Controllers

4.0 Summary and Conclusion

4.1: Summary

In this study, our primary objective was to enhance the control system for a DC motor. To accomplish this, two different control approaches was developed, an Artificial Neural Network (ANN) model controller and conventional Proportional-Integral-Derivative (PID) controller. Then simulations were conducted to compare the performance of these controllers, with a focus on stability and precision in regulating the motor's speed. Our simulations showed that the ANN controller significantly outperformed the PID controller in terms of stability. When assessing the precision of speed control for the DC motor, the PID controller exhibited a deviation of 1.435%, while the ANN model achieved a remarkable deviation of just 0.242%. This substantial improvement in performance underscores the effectiveness of the ANN model in achieving precise and stable control.

4.2 Conclusion

In summary, this research has shown that using an Artificial Neural Network (ANN) controller offers significant advantages in the speed regulation of DC motors when compared to the conventional Proportional-Integral-Derivative (PID) controller. The ANN controller demonstrated superior stability and higher accuracy, confirming its effectiveness in motor control applications. These results underscore the promise of intelligent control systems in advancing motor control technologies. With its enhanced precision and robustness, the ANN approach presents a compelling alternative for various real-world applications. As control technologies continue to evolve, the insights gained from this study highlight the growing relevance of ANN-based systems in industries that demand precise and reliable motor control solutions.

Reference

1. K. J. Åström and T. Hägglund, Advanced PID Control, Instrumentation, Systems, and Automation Society (ISA), 2006, ISBN: 978-1-55617-942-6.
2. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, Cambridge, Massachusetts, 2016, ISBN: 978-0-262-03561-3.
3. S. Haykin, Neural Networks and Learning Machines, 3rd ed., Pearson Education, 2009, ISBN: 978-0-13-147139-9.
4. Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.
5. K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," IEEE Trans. Neural Netw., vol. 1, no. 1, pp. 4–27, 1990, doi: 10.1109/72.80202.
6. J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Netw., vol. 61, pp. 85–117, 2015, doi: 10.1016/j.neunet.2014.09.003.
7. A. Hamza and N. B. Yahia, "Artificial neural networks controller of active suspension for ambulance based on ISO standards," Proc. Inst. Mech. Eng. Part D J. Automob. Eng., vol. 237, no. 1, pp. 34–47, Jan. 2023, doi: 10.1177/09544070221075456.
8. B. Sukka, R. Rameshappa, and N. M. P. Shadaksharappa, "Performance analysis of PID and ANN based speed controller for DC motor," Int. J. Electr. Comput. Eng., vol. 12, no. 5, pp. 4700–4711, 2022, ISSN: 2088-8708.

9. M. Caudill and C. Butler, Naturally Intelligent Systems, Bradford Books, 1989.
10. E. E. de Barros Ruano, Application of Neural Network to Control System, PhD thesis, University of Wales, School of Electrical Engineering Science, 1992.
11. K. Suzuki, "Applications of Artificial Neural Networks: Past, Present and Future," Tokyo Institute of Technology, Apr. 2011, doi: 10.5772/644, ISBN: 978-953-307-243-2.
12. F. I. Lewis, S. Jagannathan, and A. Ildirek, "Neural network control of robot manipulators and non-linear systems," Artif. Intell. Eng., vol. 13, pp. 55–68, 1999.
13. C. Mano, "Neural Networks Explained," eHow, 2014. [Online]. Available: http://www.ehow.com/print/about_5585309_neural-networks-explained.html
14. D. Mehr and S. Richfield, "Neural net application to optical character recognition," in Proc. 1st Int. Conf. Neural Netw., vol. 4, pp. 711–777, 1987.
15. W. Ahmed, A. Chaudhary, and G. Naqvi, "Role of Artificial Neural Networks in AI," NeuroQuantology, vol. 20, no. 13, pp. 3365–3370, 2022.