Speed Forecast of DC Motor Using Artificial Neural Network

Adepoju G. A.
Department of Electronic and Electrical Engineering
Ladoke Akintola University of Technology
Ogbomoso, Nigeria

Aborisade, D.O
Department of Electronic and Electrical Engineering
Ladoke Akintola University of Technology
Ogbomoso, Nigeria

Eluwole O. T.
Department of Electronic and Electrical Engineering
Ladoke Akintola University of Technology
Ogbomoso, Nigeria

Abstract

Artificial Neural Network (ANN) has achieved a lot of attention as well as gained enormous popularity over the last two decades owing to its vast applications both in industry and academia. This paper explicitly and effectively examines the application of ANN to speed forecast of dc motors. The angular speed of a dc motor can be determined by prediction with given voltage as input parameter. By using motor parameter values and assigning state variables as armature current, angular speed, and rotor displacement; the dc motor transfer function relating angular speed to voltage was obtained. With the input (voltage) and speed as the target. A simulation process was carried out on the transfer function using one of the most commonly used and versatile technical computing facilities (MATLAB & SIMULINK, version 7.0) to generate various input/target pairs for the ANN training and testing. Levenberg-Marquardt standard back-propagation algorithm with normalized preprocessed data was used to predict and show the pattern of correlation of input (voltage) in relation to the output (speed). The ANN developed comprised 4 layers: an input layer, 2 hidden layers, and an output layer. The input layer has 20 neurons; the first hidden layer has 20 neurons, the second hidden layer has 18 neurons, while the output layer has a single neuron. The ANN was able to develop a good mapping between the input and the target. The results obtained indicated 100% correlation and an absolute mean error (AME) of 0.38% during testing with unfamiliar input/target data pairs. The former shows a high reliability of the network to predict the output while the latter represents on an average, a very high degree of accuracy in the prediction.

Keywords: *Speed forecast, DC motor, Artificial Neural Network (ANN)*

1. Introduction

DC motors are one of the most widely used prime movers in industry today. The advancement made in power electronics has made brushless dc motors quite popular in high-performance control systems. Peculiar properties such as very low inertia, very high-torque-to-volume ratio and low-time constant properties have opened applications for dc motors in computer peripheral equipment such as tape drives, printers, disk drives, word processors, and automation/machine tool industries [1]. The speed of operation of dc motor is the crux of the matter in this research work. Carrying out an experiment in the laboratory to study the dc motor speed performance could prove cost-ineffective and time-consuming especially in a country where we have insufficient funds to maintain existing research facilities, or to even acquire state-of-the-art facilities. A rational, logical and analytical thinking suggests that system modeling and simulation of the actual physical system can offer a better alternative. By this alternative, dc motor speed control can be achieved in a number of ways such as Proportional-Integral-Derivative (PID) control, state-space control, digital PID control and others. However, when we desire a reliable and a very accurate dc motor speed forecast, the best we can think of is using the Artificial Neural Network approach.

According to [2], the ability to effectively and accurately utilize the knowledge garnered from basic sciences to bring about pioneering, scholarly and risk-free inventions to the entire world defines and makes a complete engineer.

No wonder all serious-minded electric machine manufacturers and industrial engineers often find ways of minimizing engineering time, reducing prototyping cost and optimizing product quality so that a sustainable competitive goal can be achieved. The best way to actualize these objectives is to use simulation software to predict the electric machine performance without actually creating prototypes. Without mincing words, the application of ANN in forecasting the speed of a dc motor given the input voltage appears very appealing to design and manufacturing engineers as they can use prediction as part of the specifications for their design [3]. In further sections of this paper, we would clearly examine DC motor modeling in control systems, discuss speed prediction using ANN more elaborately by addressing all the stages involved in the simulation process, look into the neuro-predictive model of the ANN, and finally present results as well as conclusions of the forecast presented in this work.

2. DC Motor Modeling In Control Systems

A dc motor consists of a stationary active part, usually called the field structure, and a moving active part, usually called the armature. Both parts carry direct current; hence, it is so called. For effective analytical purposes, mathematical models are indispensable. The separately excited dc motor is the simplest of all dc motors and it is the one most commonly found in industrial applications. A mathematical model for the separately excited dc motor, whose equivalent circuit is shown in Figure 1, is thus established [1, 2].

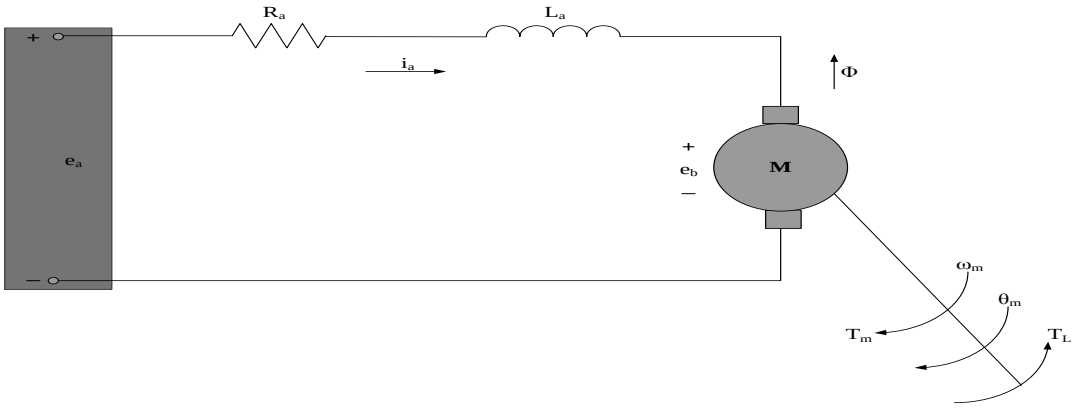


Fig. 1: Model of a separately excited DC motor

Where:

i_a = armature current (A); R_a = armature resistance (Ω); L_a = armature inductance (H)

e_a = applied (input) voltage (volts); e_b = back e.m.f (volts); T_m = motor torque (Nm)

T_L = load torque (Nm); ω_m = rotor angular velocity (rad/s); θ_m = rotor displacement (m)

ϕ = magnetic flux (Weber)

Other motor variables and parameters are defined as:

B_m = viscous friction coefficient (damping ratio of mechanical system) (Kgm^2/s or Nms)

J_m = motor inertia (Kgm^2); K_b = back e.m.f constant (Nm/A); K_a = motor constant (Nm/A)

By applying the basic circuit laws and dc motor principles and assigning state variables as i_a, ω_m and θ_m, the state equations are obtained and given in matrix form below:

$$\begin{bmatrix} di_a/dt \\ d\omega_m/dt \\ d\theta_m/dt \end{bmatrix} = \begin{bmatrix} -R_a/L_a & -K_b/L_a & 0 \\ K_a/J_m & -B_m/J_m & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_a \\ \omega_m \\ \theta_m \end{bmatrix} + \begin{bmatrix} 1/L_a \\ 0 \\ 0 \end{bmatrix} e_a + \begin{bmatrix} 0 \\ -1/J_m \\ 0 \end{bmatrix} T_L \qquad (1)$$

By drawing the state diagram from the state equations and applying Mason's gain formula, the dc motor transfer function relating speed to voltage is obtained as follows:

$$\frac{\omega_m(s)}{E_a(s)} = \frac{K_a}{L_a J_m s^2 + (R_a J_m + B_m L_a)s + (K_a K_b + B_m R_a)} \qquad (2)$$

3. Speed Prediction Using ANN

A neural network comprises simple elements operating in parallel. In fact, it is a massively parallel distributed processor that stores experiential knowledge and makes it available for use when needed. ANN is a model of a biological neural network. It is mathematical model or computational model based on biological neural networks [5]. ANN is an information processing paradigm (highly interconnected network of processing elements) that mimics the functioning and connectivity of the human brain [6]. A neural network, just like human beings learns by example and it is fault-tolerant. The true power and advantage of neural networks lies in their ability to derive meaning from complex or imprecise data [7], [13].

Back-propagation: In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. The back-propagation algorithm is a supervised learning algorithm used to change or adjust the weights of the neural network. It is such that the gradient vector of the surface is calculated. This vector points along the direction of the steepest descent from the current point, so that a movement over a short distance along it decreases its error. A sequence of such moves will eventually find a minimum error point [8]. With back-propagation, the input data is repeatedly presented to the neural network. With each presentation, the output of the neural network is compared to the desired output and an error is computed. This error is then fed back (back-propagated) to the neural network and used to adjust the weights such that the error decreases with each iteration and the neural model gets closer and closer to producing the desired output. A number of steps are involved in back-propagation [5], [9], [10].

1. **Initialization:** this is usually done before training. Weights and biases must be initialized. Initialization means resetting the network weights and biases to their original values. The MATLAB function **'init'** takes a network object as input and returns a network object with all weights and biases initialized.
2. **Training:** training and learning functions are mathematical procedures used to automatically adjust the network weights and biases. While the former dictates a global algorithm that affects the weights and biases of a given network, the latter can be applied to individual weights and biases within a network. ANN training basically consists on determining the network parameters such as weights and others, which allow achieving the desired objective based on the available training sets. Usually, multilayer feed-forward neural networks are trained in a supervised manner according to the back-propagation algorithm. The MATLAB function '**train'** is used to train a network. Once the training data is assembled, the network object is created and then training begins. In this paper, Levenberg Marquardt (LM) back-propagation algorithm initiated by MATLAB function **'trainlm'** is used. It is apparent that out of all back-propagation algorithms for function approximation problems, the LM algorithm is the best. Training stops when the performance has been minimized to the goal, the performance gradient falls below a minimum gradient, the maximum number of epochs is reached, or the maximum amount of time has been executed.
3. **Preprocessing and Post-processing:** neural network training can be made more efficient if certain processing steps are performed on the network inputs and targets (preprocessing). It is often useful to scale inputs and targets before training so that they always fall within a specified range. By this approach, improperly recorded and abnormal data are identified and discarded or adjusted using a statistical method to avoid contamination of the model. The preprocessing function **'premnmx'** is used to scale inputs and targets so that they fall in the range [-1, 1]. To convert the outputs back to the same units that were used for the original target (de-scaling), the post-processing function **'postmnmx'** is used.
4. **Simulation:** the simulation function **'sim'** simulates the network. It takes the network input and the network object, and then returns the network output. Using the trained neural network, the forecast output is simulated using the input parameter.

Motor parameter values (derived by experiment from an actual motor in Carnegie Mellon's Undergraduate Control Laboratory, the University of Michigan United States) were substituted into the equation above and a SIMULINK block diagram was constructed to generate input/target pairs for the ANN. Simulations of a particular block diagram can be used to test a number of 'what-if' questions [4], [12]. Overall, the input/target pairs generated numbered up to 110 values each, 70 of which were used for training the network with the remaining 40 presented as unknown to the trained network so a forecast of the speed can be made accordingly.

4. Neuro-Predictive Model of the ANN

The algorithm used is such that about 60% of the data records were randomly assigned for training, 20% went for testing, and the remaining 20% were relegated to validation.

The network training was carried out using the Levenberg-Marquardt (LM) standard back-propagation algorithm because of its high speed of convergence and accuracy. The stop criteria were based on the mean-square error (MSE) analysis [10]. The best result was obtained for the ANN which comprised twenty neurons in the input layer, twenty neurons in the first hidden layer, eighteen neurons in the second hidden layer, and a single neuron in the output layer. The transfer functions used for the four layers were purelin, tansig, purelin, and purelin respectively. The inputs were preprocessed using the normalizing technique which sought relevant direction for the former so that variance can be maximized. For validation, input/target pairs of untrained data (not known by the ANN) were presented to the network to determine how well it predicts the corresponding outputs.

5. Results

5.1 Graphs and ANN Architecture generated by the LM algorithm

The LM algorithm used yielded four results:

i. The performance graph
ii. The disparity graph
iii. The regression graph
iv. The artificial neural network architecture or SIMULINK

The performance graph shown in Figure 2 indicates the behavior of the network during training, validation and testing. The performance goal was met during:
TRAINLM, Epoch 0/100, MSE 31.4661/1e-005, Gradient 3475.59/1e-010
TRAINLM, Epoch 5/100, MSE 4.00778e-006/1e-005, Gradient 0.51086/1e-010
TRAINLM, Performance goal met.

Epoch shows the presentation of the set of training (input and/or target) vectors to the network and the calculation of new weights and biases. The performance function is the MSE of the network outputs. Level of discrepancy between the targeted output and the simulated (ANN) output is shown by the disparity graph of Figure 3. This graph indicates a perfect mapping between the target and the output. The Regression graph shown in Figure 4 gives the association or relatedness between the outputs and the targets. With R = 1 from the graph, the degree to which the outputs and targets are related and change together is 100%. Thus, line of best fit equation is A = 0.997T + 0.0385. The four-layer fully connected feed-forward artificial neural network generated by the LM algorithm shown in Figure 5 includes an input layer, two hidden layers and an output layer. Signal propagation is allowed only from the input layer to the first hidden layer, from the first hidden layer to the second hidden layer, and from the latter to the output layer. However, the output of the neural network is compared to the target and an error is computed.

5.2. Simulated Results

Various input values were used to test the accuracy the ANN results. A number of instances of such are shown in Table 1. The dc motor responses for various inputs are shown in Figure 6(a-h). During testing with unfamiliar data part of which is shown in the table above, the highest and lowest errors recorded are 0.54% and 0.01% respectively. Overall, the entire error values translate to an absolute mean error (AME) of 0.38% for the network. This represents a high degree of accuracy in the ability of neural networks to predict speed.

Table 1: summary of a part of the test results

INPUT E_a (V)	TARGET ω_m (rad/sec)	OUTPU ω_m (rad/sec)	PERCENTAGE ABSOLUTE ERROR
10.05	1.0000	1.0040	0.40
10.85	1.0800	1.0848	0.48
11.45	1.1400	1.1453	0.53
11.65	1.1600	1.1654	0.54
12.85	1.2800	1.283	0.30
13.65	1.3600	1.3606	0.06
13.85	1.3800	1.3801	0.01
13.95	1.3900	1.3906	0.06

6. Conclusion and Future Research

The results obtained from this work confirm the relevance as well as the efficiency of neural networks in dc motor speed prediction. The ANN approach has proved to be accurate and computationally fast.

The use of normalized preprocessing and post-processing techniques proved undoubtedly essential for an improved overall performance of the ANN. The training algorithm used, usually applicable to function approximation problems, trained the neural network on average, about 55 times faster than the usual back-propagation algorithms. The mean-square error (MSE) decreased much more rapidly with time, and as the error goal was reduced, improvement became more pronounced other than being degraded. However, it must be noted that the algorithm described and used in this paper is not applicable to pattern recognition problems or tasks involving very large number of weights. What happens is that a relatively poor performance could result and the intended goal of prediction may appear erratic. Without any doubt, applications of ANN are multifarious and almost endless. Further studies on this work can take into consideration parameters other than speed. For instance, the dc motor position control can be dealt with using ANN. All known methods of dc motor speed performance/control should also be examined and compared to the ANN technique so that the most efficient technique will be identified.

References

[1] Kuo, B. C. and Golnaraghi, F. "Automatic Control Systems". Eighth Edition, John Wiley & Sons Inc., 2003.

[2] Faculty of Engineering (2011), "What is Engineering?" University of Western Ontario, Canada. Available from: http://www.eng.uwo.ca/comms/about.htm [Accessed 30 May 2011]

[3] Okoro, O. I. "Application of Numerical Softwares in Electrical Machines Modeling", *Lautech Journal of Engineering and Technology.* 2(2)2005:16-22

[4] Richard, C. D. (Editor-in-Chief). "The Electrical Engineering Handbook". Second Edition CRC Press LLC/IEEE Press, 1997.

[5] Eric, D. and Patrick, N. "DARPA Neural Network Study". October, 1987 – February, 1989. MIT Lincoln Lab. Neural Networks.

[6] Bernander, O. and Redmond, W.A. "Neural Network". Microsoft Student 2008 [DVD]. Microsoft Corporation, 2007.

[7] Stergiou C. and Siganos D., "Neural Networks" Available at: http://www.doc.ic.ac.uk [Accessed 30 May 2011]

[8] Rosenblatt, F. "Principles of Neurodynamics", Washington D.C., Spartan Press, 1961.

[9] Adepoju, G.A., Ogunjuyigbe S. O. A., and K.O. Alawode. (2007), "Application of Neural Network to Load Forecasting in Nigerian Electrical Power System", *Pacific Journal of Science and Technology.* 8(1):68-72.

[10] The MathWorks Inc. 1992-2008. "Neural Network Toolbox 6, User's Guide". Available at: http://www.mathworks.com

[11] Masters, T. "Signal and Image Processing with Neural Networks". John Wiley & Sons Inc. 2003, ISBN 0-471-04963-8

[12] Steven, T. K. "Introduction to Simulink with Engineering Applications". Orchard Publications, 2006. Available at: http://www.orchadpublications.com

[13] Stuart, J. R. and Peter, N. "Artificial Intelligence: A Modern Approach". Second Edition, Prentice Hall, New Jersey.

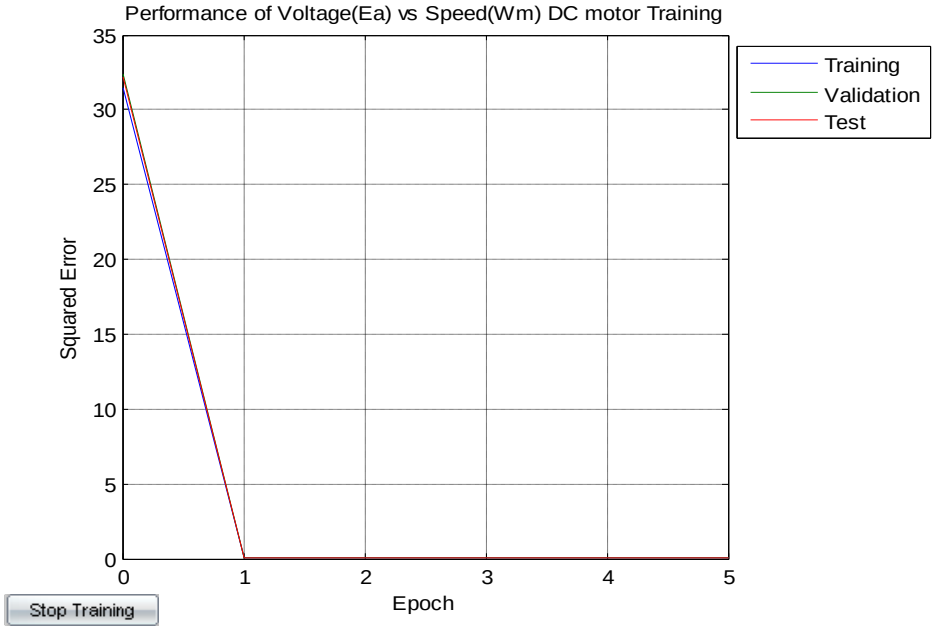


Figure 2: The Performance Graph

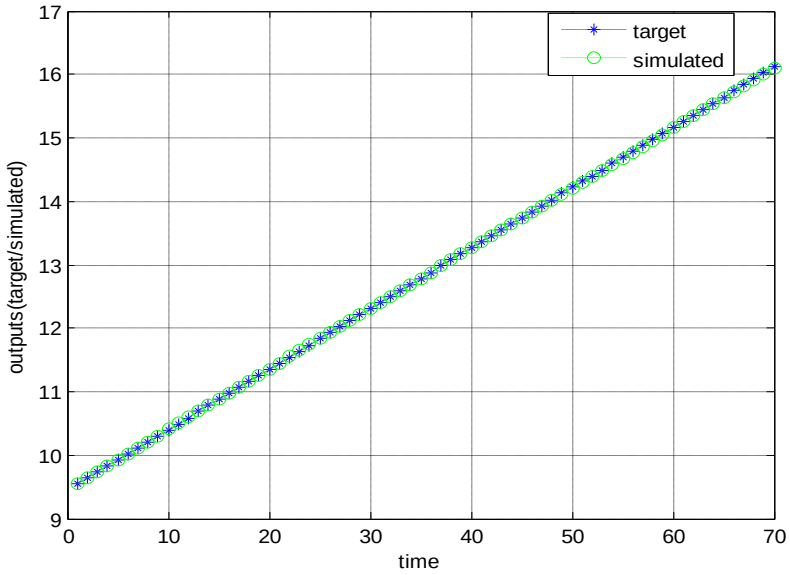


Figure 3: The Disparity Graph

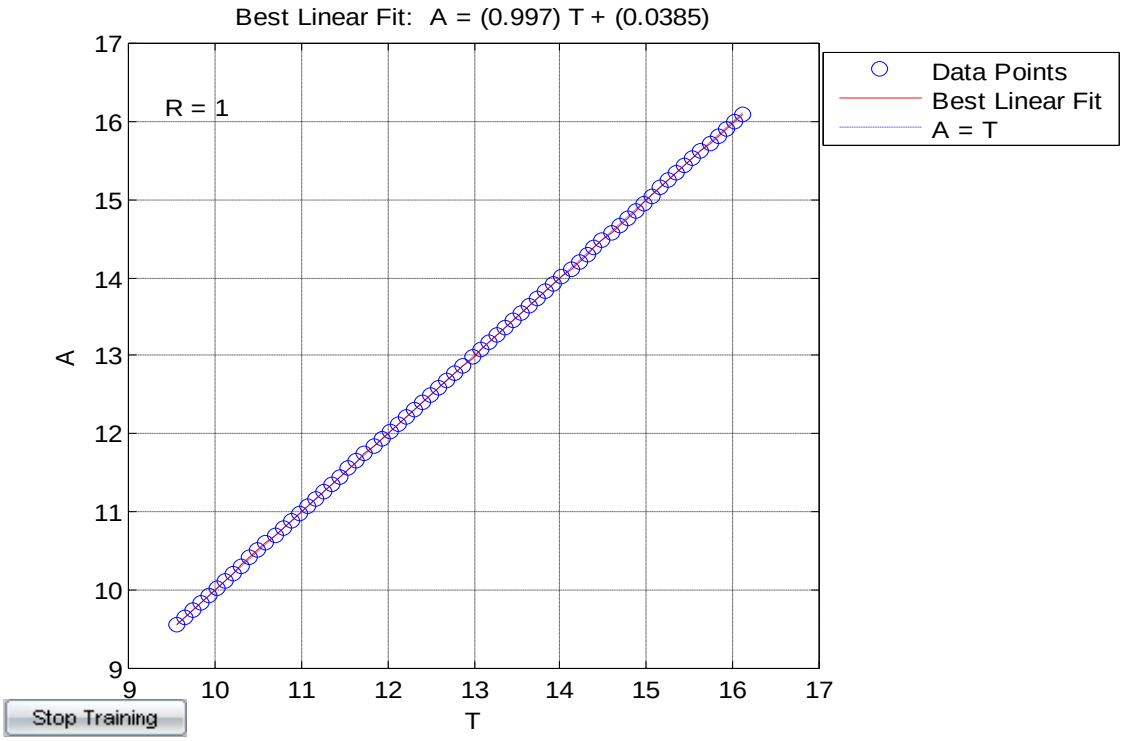


Figure 4: The Regression Graph

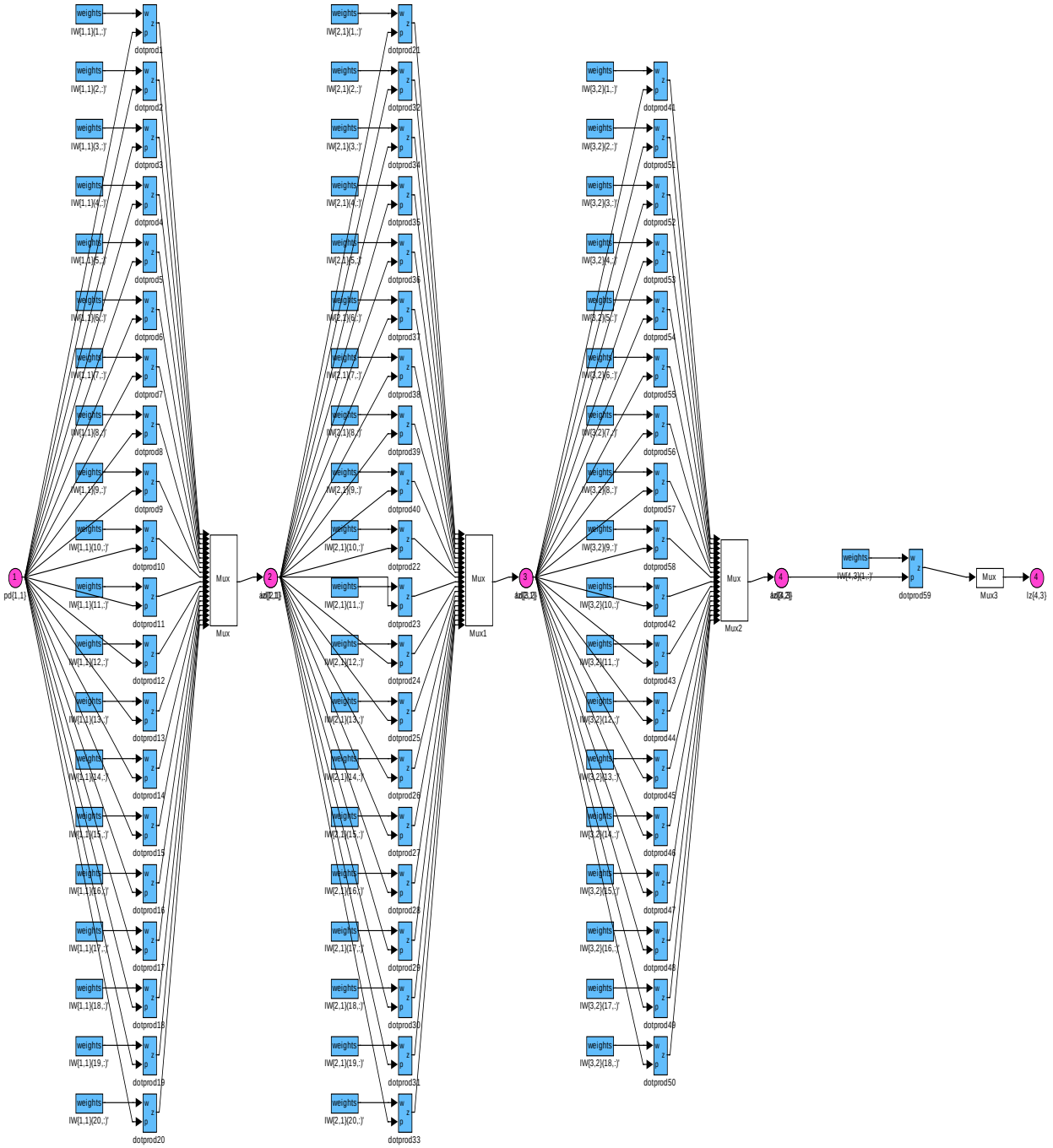
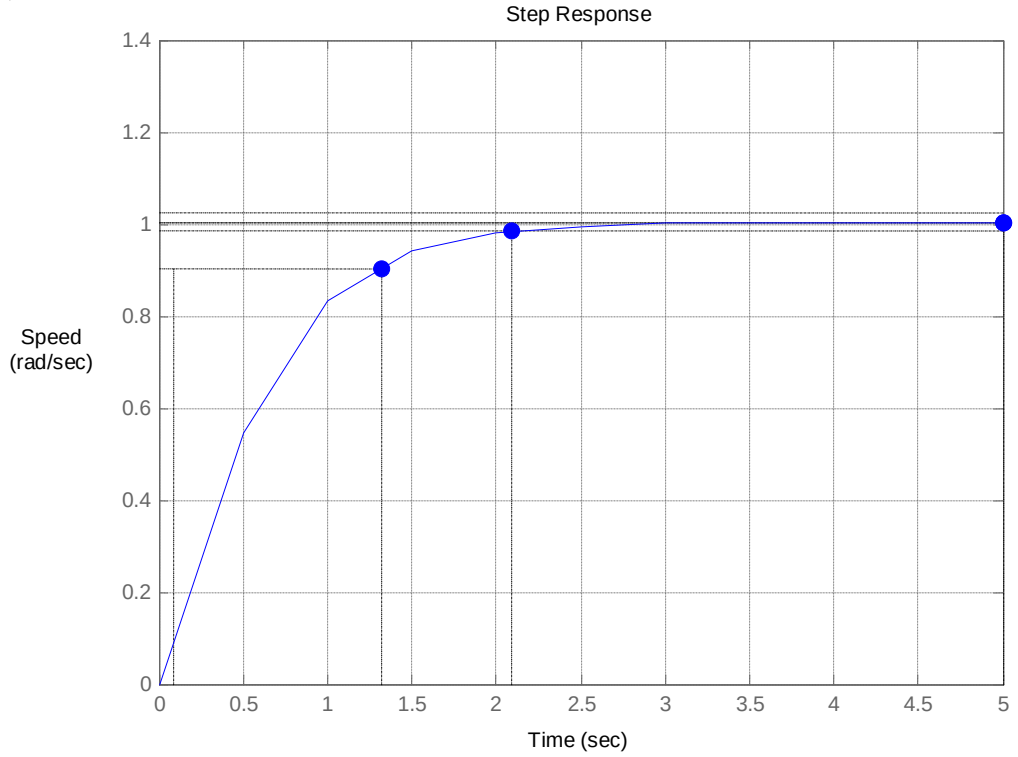


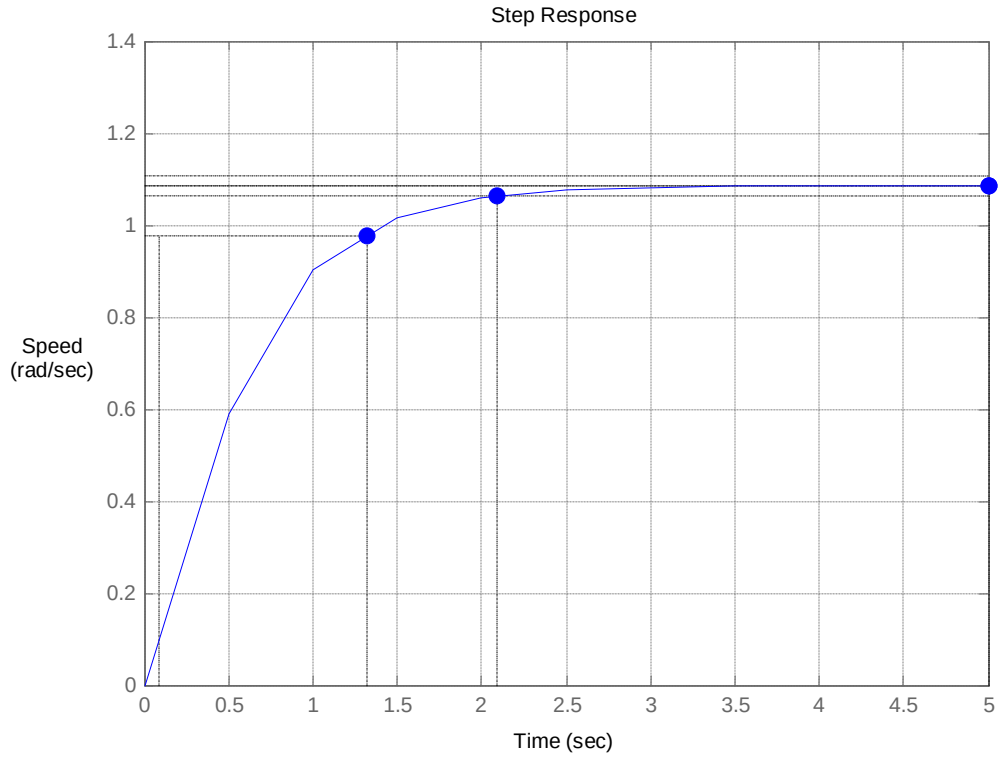
Figure 5: The Artificial Neural Network Architecture

(a) At 10.05 volts



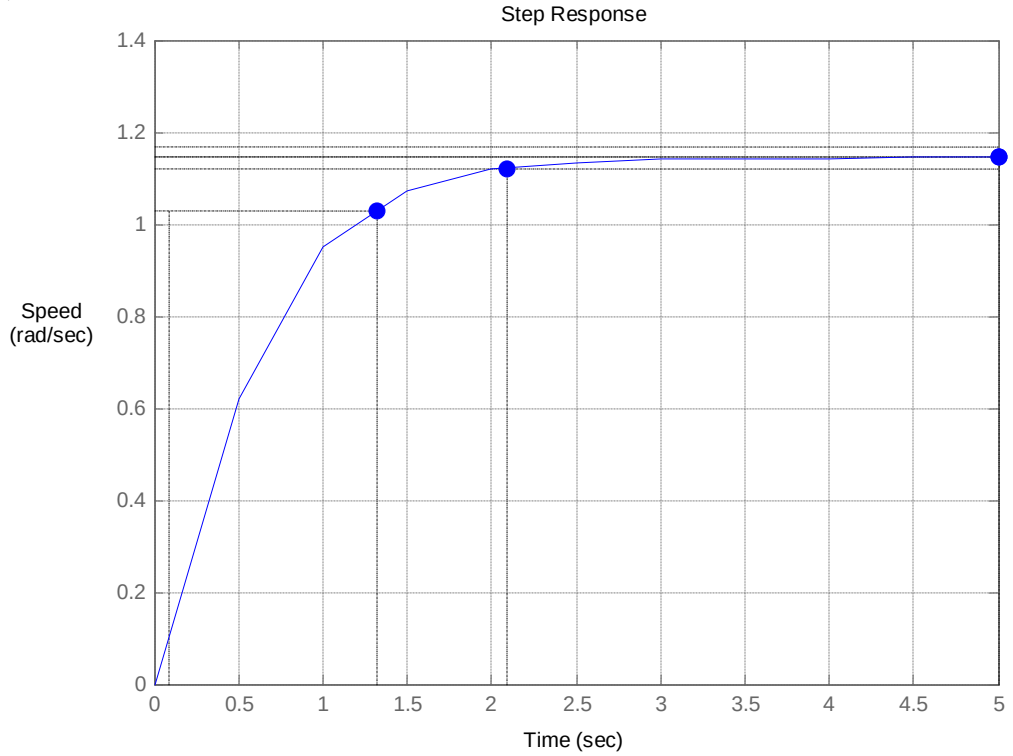
Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.00 rps; ANN result = 1.004 rps

(b) At 10.85 volts



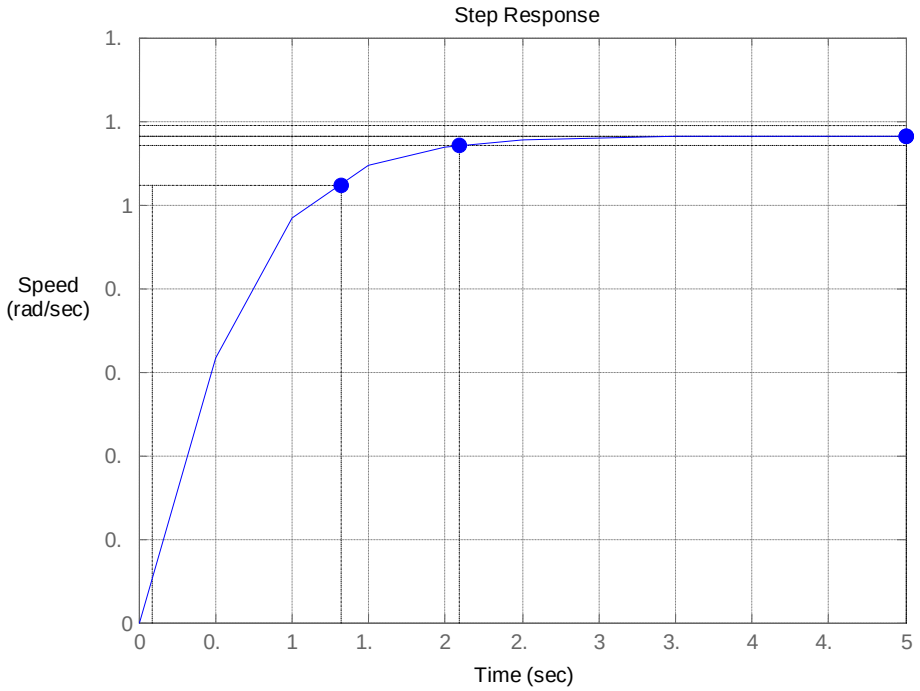
Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.08 rps; ANN result = 1.0848 rps

(c) At 11.45 volts



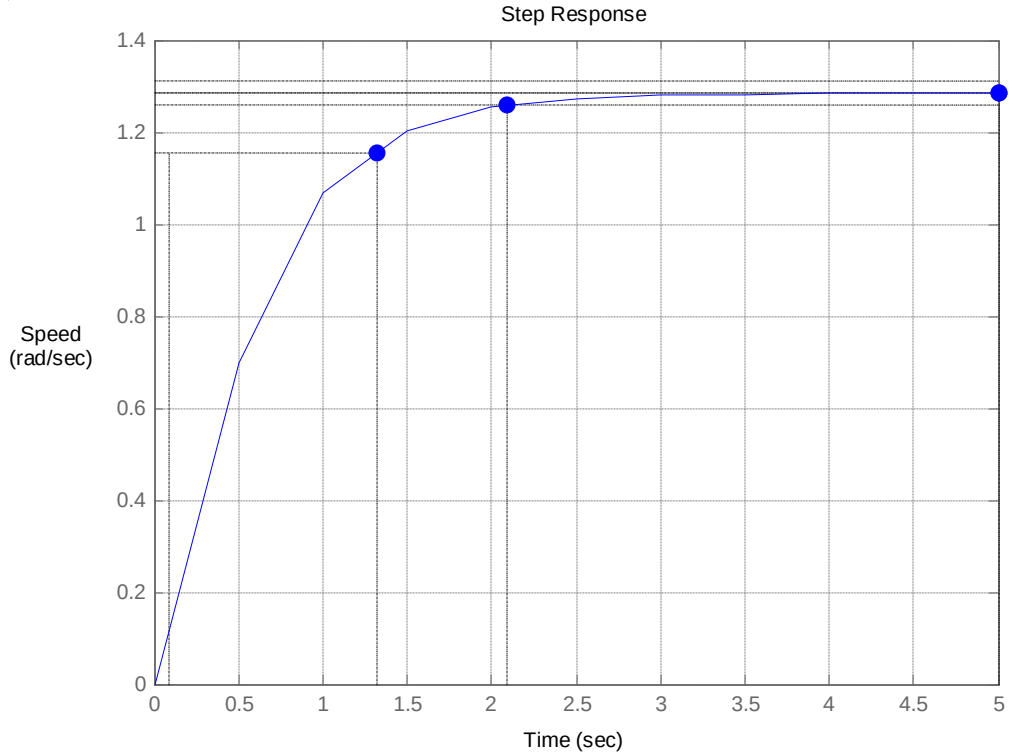
Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.14 rps; ANN result = 1.1454 rps

(d) At 11.65volt



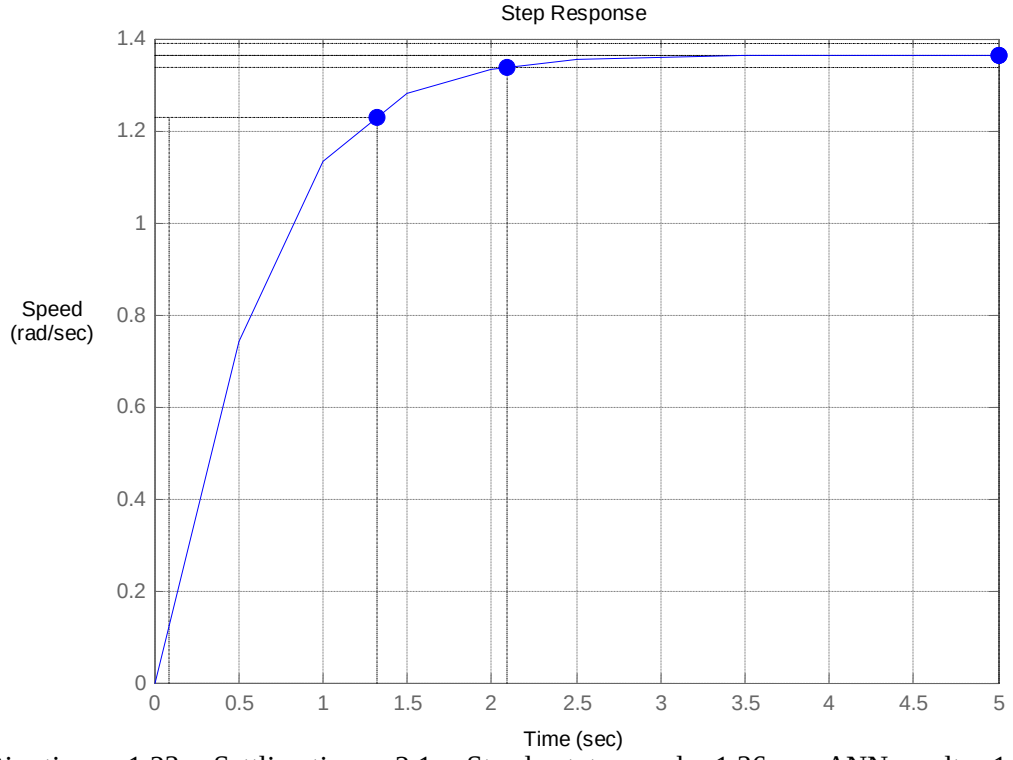
Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.16 rps; ANN result = 1.1654 rps

(e) At 12.85 volts



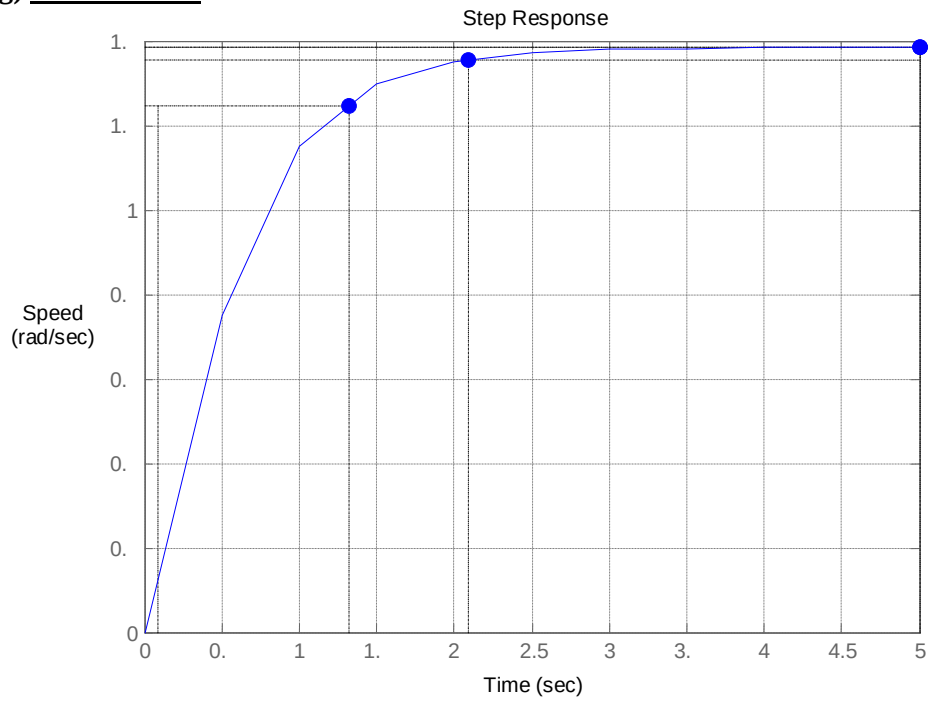
Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.28 rps; ANN result = 1.283 rps

(f) At 13.65 volts



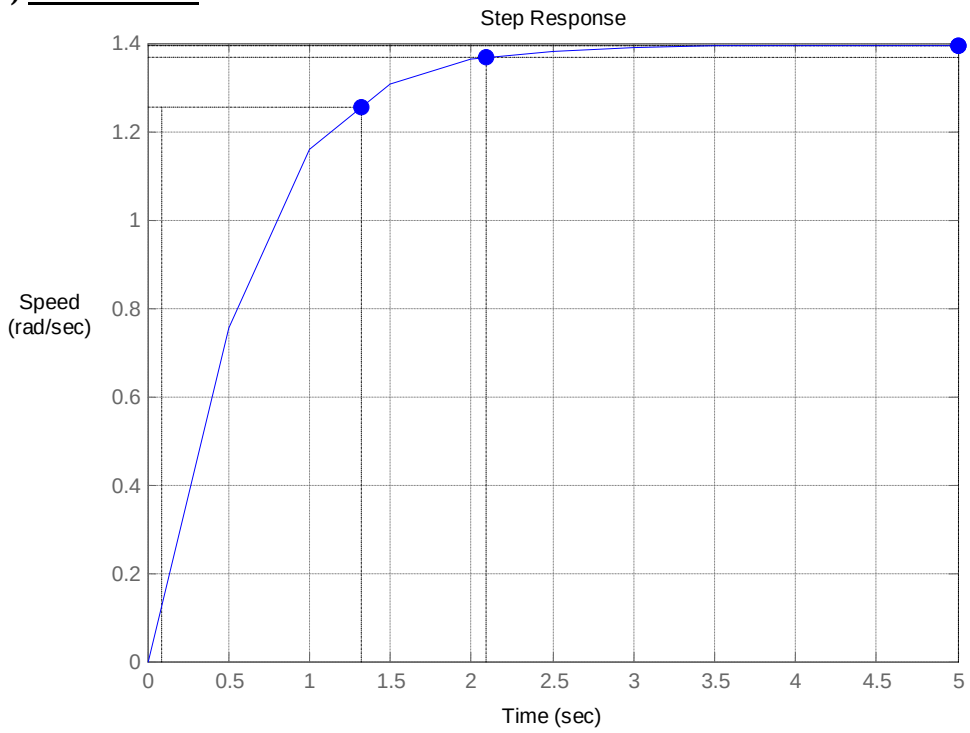
Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.36 rps; ANN result = 1.3606 rps

(g) At 13.85 volts



Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.38 rps; ANN result = 1.3801 rps

(h) At 13.95 volts



Rise time = 1.23 s; Settling time = 2.1 s; Steady-state speed = 1.39 rps; ANN result = 1.3906 rps

Figures 6(a) – (h): Step response of the dc motor transfer function for various inputs