International Journal of Electrical and Electronic Engineering & Telecommunications

ISSN 2319 – 2518 www.ijeetc.com
Vol. 5, No. 3, July 2016

Research Paper

USING LEVENBERG-MARQUARDT STANDARD BACK-PROPAGATION ALGORITHM IN SPEED EXTRAPOLATION FOR DC MOTORS

Opata Udoka C[1]*

*Corresponding Author: Opata Udoka C, ✉ opataudoka@gmail.com

Artificial Neural Network has found so many applications in engineering. This paper focuses on extrapolating the speed of DC motor, given a set of input conditions. Experimental data consisting of input voltage and speed was collected from actual dc motor in the laboratory. This served as input/target pairs for the ANN after modeling and simulation of the DC Motor in Matlab Simulink. 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 results show that the ANN model correctly maps the input to the output. This shows that Artificial Neural Network was able to predict the speed accurately.

Keywords: DC Motors, Artificial Neural Networks, Levenberg-Marquardt standard back-propagation algorithm, and Speed Prediction

INTRODUCTION

Neural networks have been applied by researchers in solving various multifarious problems in engineering and other areas of life. Some of these applications are seen in Sung-Hoe Huh *et al.* (2005), Abdel Karim *et al.* (2006), Adepoju *et al.* (2007) and Arif Al-qassar and Mazin Othman (2008). This paper focuses on its application to DC motors.

Large DC motors are used in machine tools, printing presses, conveyors, fans, pumps, hoists, cranes, paper mills, textile mills and so forth. Small DC machines (in fractional horsepower rating) are used primarily as control devices such as tacho-generators for speed sensing and servomotors for positioning and tracking. DC motors still dominate as traction motors used in transit cars and locomotives as the torque-speed characteristics of DC motor can be varied over a wide range while retaining high efficiency (Sen, 1989). The DC machine definitely plays an important role in the industry.

The objective of this study is to predict or forecast the speed of Dc Motors using Artificial Neural Network (ANN).This will help design/

[1] Faculty of Engineering, Department of Electronic Engineering, University of Nigeria Nsukka, Enugu, Nigeria.

manufacturing engineers to be able to evaluate the speed of DC Motors in the design of production systems, flexible automatic manufacturing plants, robotic systems etc without carrying out specific tests and construction in the laboratory. This will not only save cost but will also save time. Therefore ANN can be used to predict motor speed in control system models and there will be no need to calculate the parameters of the motor when designing the system control. For instance ANN can be used to estimate Dc Motor parameters (Arif Al-qassar and Mazin Othman, 2008).

RESEARCH METHODOLOGY

Modeling of DC Motor

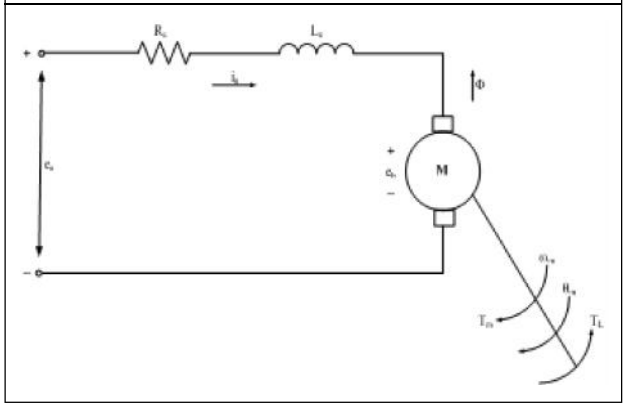
For effective analytical purposes, mathematical models are required (Moleykutty George, 2008). Thus, we would establish a mathematical model for the separately excited dc motor whose equivalent circuit is shown in Figure 1 below.

where

i_a = armature current in A

R_a = armature resistance in Ω

Figure 1: Model of a Separately Excited DC Motor



L_a = armature inductance in H

e_a = applied (input) voltage in V

e_b = back emf in V

T_m = motor torque in Nm

T_L = load torque in Nm

μ_m = rotor angular velocity (speed) in rad/s

θ_m = rotor displacement in m

¤ = magnetic flux in Weber

Other motor variables and parameters are defined as:

B_m = viscous friction coefficient (damping ratio of mechanical system) in Kgm²/s or Nms

J_m = motor inertia in Kgm²

K_b = back emf constant in V/rad/s or Nm/A

K_a = motor constant in Nm/A

Firstly, we write out the cause-and-effect equations for the equivalent circuit shown above. By applying Kirchoff's Voltage Law (KVL) around the loop, we obtain:

$$\frac{di_a}{dt} = \frac{1}{L_a}e_a - \frac{R_a}{L_a}i_a - \frac{1}{L_a}e_b \ldots \quad ...(1)$$

By dc motor principles, $T_m \sim i_a$ so that:

$$T_m = K_a i_a \quad ...(2)$$

Owing to the back emf generated by the motor, we write that:

$$e_b = K_b \mu_m \quad ...(3)$$

Rearranging Equation 2.1, we have

$$e_a = e_b + R_a i_a + L_a \frac{di_a}{dt} \quad ...(4)$$

From Equations 3 and 4 we get

$$e_a = K_b \mu_m + R_a i_a + L_a \frac{di_a}{dt} \quad ...(5)$$

The dynamic equation for the mechanical system is

$$T_m = K_a i_a = J_m \frac{d\mu_m}{dt} + B_m \mu_m + T_L \quad ...(6)$$

The term $B_m \mu_m$ represents the rotational loss torque of the system

But $\mu_m = \frac{d\theta_m}{dt}$ ...(7)

Therefore Equation 6 becomes

$$T_m = J_m \frac{d^2\theta_m}{dt^2} + B_m \frac{d\theta_m}{dt} + T_L \quad ...(8)$$

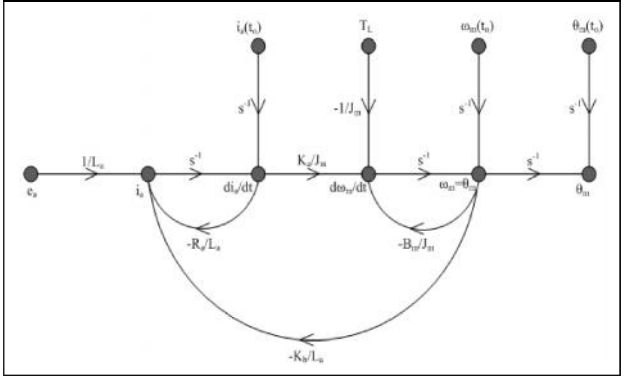
Rearranging Equation 8 yields:

$$\frac{d^2\theta_m}{dt^2} = \frac{1}{J_m} T_m - \frac{1}{J_m} T_L - \frac{B_m}{J_m} \frac{d\theta_m}{dt} \quad ...(9)$$

By assigning the state variables as i_a, μ_m and θ_m, the state equations are obtained and given in matrix form in Equation 10 below:

$$\begin{bmatrix} di_a/dt \\ d\mu_m/dt \\ d\theta_m/dt \end{bmatrix} = \begin{bmatrix} -R_a/L_a & -K_b/L_a & 0 \\ K_a/J_m & -B_m/J_m & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_a \\ \mu_m \\ \theta_m \end{bmatrix} + \begin{bmatrix} 1/L_a \\ 0 \\ 0 \end{bmatrix} e_a + \begin{bmatrix} 0 \\ -1/J_m \\ 0 \end{bmatrix} T_L \quad ...(10)$$

Figure 2: State Diagram



From the state equations above, we can draw the state diagram as shown in Figure 2 below:

Using the state diagram, we can determine the characteristic determinant as:

Δ = Σ individual loop gains – Σ all possible 2 non-touching loop gains + Σ all possible 3 non-touching loop gains +-…

$$\Delta = 1 - \left(\frac{R_a}{L_a s} - \frac{B_m}{J_m s} - \frac{K_a K_b}{L_a J_m s^2} \right) + \left(\frac{B_m R_a}{L_a J_m s^2} \right) \quad ...(11)$$

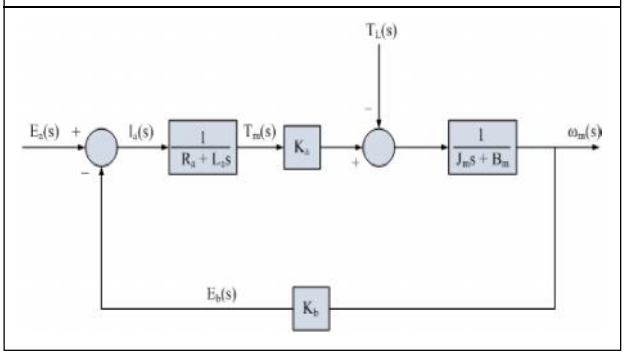
$$\Delta = \frac{L_a J_m s^2 + (R_a J_m + B_m L_a)s + (K_a K_b + B_m R_a)}{L_a J_m s^2} \quad ...(12)$$

Finding the dc motor speed transfer function by applying Mason's gain formula on the state diagram yields:

$$\frac{\mu_m(s)}{E_a(s)} = \frac{K_a}{L_a J_m s^2 + (R_a J_m + B_m L_a)s + (K_a K_b + B_m R_a)} \quad ...(13)$$

From the above result, the block diagram for the speed control of the separately excited dc motor now looks like what we have in Figure 3 below.

Figure 3: Block Diagram of the Separately Excited DC Motor System



Simulation Using Matlab

The program used to complete the modeling and simulation is called SIMULINK, a sub program of MATLAB. In order to complete the modeling and simulation of the DC motors, their parameters were measured and some calculated as shown below.

From the transfer function depicting the ratio of the speed μ_m to input voltage e_a (Equation 13), by substituting for the motor parameters using values given in Table 1, we obtain:

For DC Motor 1

$$\frac{\mu_m(s)}{E_a(s)} = \frac{1.46}{0.00033s^2 + 0.032s + 1.91} \quad ...(14)$$

For DC Motor 2

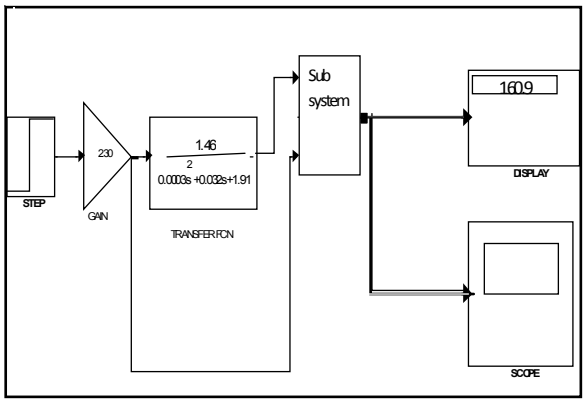
$$\frac{\mu_m(s)}{E_a(s)} = \frac{1.66}{0.00012s^2 + 0.053s + 1.41} \quad ...(15)$$

To generate values for input/target pairs which will be used to train and test the neural network, we will construct a SIMULINK model in the MATLAB environment that depicts the transfer function of Equations 13 and 14 above (Steven, 2006).

Table 1: DC Motor Parameters

Parameter	DC Motor 1 (4 KW)	DC Motor 2 (8 KW)
Armature resistance, R_a	1.35	1.93
Armature inductance, L_a	0.014 H	0.0046 H
Motor inertia, J_m	0.0236 Nm/rad/s^2	0.0271 Nm/rad/s^2
Viscous friction coefficient, B_m	0.0083 Nm/rad/s	0.024 Nm/rad/s
Motor constant, K_a	1.46 Nm	1.66 Nm
Back emf constant, K_b	1.3 Nm/A	0.82 Nm/A
Load Torque, T_L	25.5 Nm	32.2 Nm
Terminal Voltage	230 V	230 V

Figure 4: SIMULINK Model for the Separately Excited DC Motor

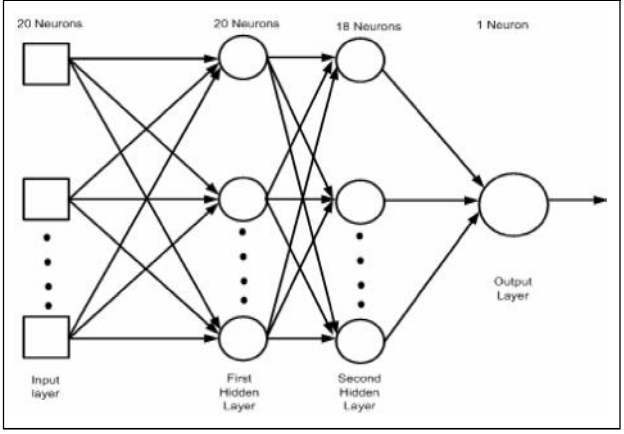


By varying the input voltage over the stipulated range of 230 V-92 V (decrement) of 2 V, singular values of e_a are generated and used as the network input values. In a likewise manner, target values μ_m are generated by simulating the input voltage for every of its initial value.

Artificial Neural Network Architecture

Levenberg-Marquardt standard back-propagation algorithm with normalized

Figure 5: The 4 Layer Feed Forward Neural Network



preprocessed data was used to predict and show the pattern of correlation of input(voltage) in relation to the output (speed). This algorithm was chosen because of its high speed of accuracy and convergence. A total number of 100 data values generated from the simulation in MATLAB was presented to the network. The algorithm used is such that 70% of the data records were randomly assigned for training, 15% for testing, and the remaining 15% were relegated to validation. The ANN developed has 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 transfer functions used for the four layers were purelin, tansig, purelin, and purelin respectively. The 4 layer feed forward neural network used is shown in Figure 5.

Levenberg-Marquardt Algorithm (Trainlm)

Levenberg-Marquardt algorithm is a very popular curve-fitting algorithm used in many software applications for solving generic curve-fitting problems. It is an approximation to Newton's method. If a function $V(x)$ is to be minimized with respect to the parameter vector x, then Newton's method would be:

$$\Delta \underline{x} = -\left[\nabla^2 V(\underline{x})\right]^{-1} \nabla V(\underline{x}) \quad \text{...(16)}$$

where $\nabla^2 V(\underline{x})$ is the Hessian matrix and $\nabla V(\underline{x})$ is the gradient.

The Levenberg-Marquardt modification to the Gauss-Newton method is:

$$\Delta \underline{x} = \left[J^T(\underline{x}) J(\underline{x}) + \mu I\right]^{-1} J^T(\underline{x})\, \underline{e}(\underline{x}) \quad \text{...(17)}$$

where J(x) is the Jacobian matrix

For the neural network-mapping problem, the terms in the Jacobian matrix can be computed by a simple modification to the back-propagation algorithm (ÖzgürKisi, 2004).

The training parameters for trainlm are *epochs, show, goal, time, min_grad, max_fail, mu, mu_dec, mu_inc, mu_max,* and *mem_reduc*. The training status is displayed for every show iterations of the algorithm. The parameter *mu* is the initial value for μ. This value is multiplied by *mu_dec* whenever the performance function is reduced by a step. It is multiplied by *mu_inc* whenever a step would increase the performance function. If mu becomes larger than *mu_max*, the algorithm is stopped. The parameter *mem_reduc* is used to control the amount of memory used by the algorithm. The other parameters determine when the training stops. The training stops if the number of iterations exceeds epochs, if the performance function drops below goal, if the magnitude of the gradient is less than *mingrad*, or if the training time is longer than time seconds. *max_fail*, is associated with the early stopping technique (Demuth *et al.*, 1992-2009).

RESULTS AND DISCUSSION

The Simulated Results

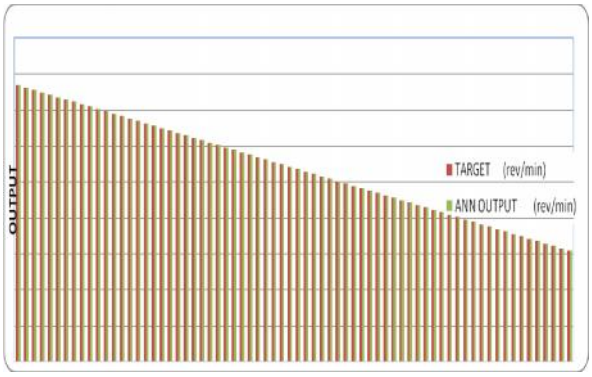
The training of the neural network entailed collection of historical data. Motor parameter values (derived by experiment from Dc motor in Electrical Engineering Machines Laboratory, the University of Lagos, Nigeria) was used in the MATLAB/SIMULINK environment to generate input/target pairs for the ANN.The training data set generated, was presented to the network. The input represents voltage, E_a(V) while the target represents speed, μ_m(rev/

Table 2: Summary of a Part of the Test Results

Input E_a (V)	Target w_m (rev/min)	ANN Output w_m (rev/min)	Absolute Erroe	Percentage Absolute Error (%)
230	1536.772	1536.775	0.003	0.3
228	1523.4	1523.402	0.002	0.2
226	1510.029	1510.032	0.003	0.3
224	1496.657	1496.58	0.001	0.1
222	1483.286	1483.288	0.002	0.2
220	1469.914	1469.918	0.004	0.4
218	1456.543	1456.546	0.003	0.3
216	1444.126	1444.131	0.005	0.5

min). The ANN outputs for the various inputs are shown and the absolute error calculated.

Figure 6: A Bar Chart Displaying the Input Voltage (volts) versus Speed (rev/min) for the DC Motor



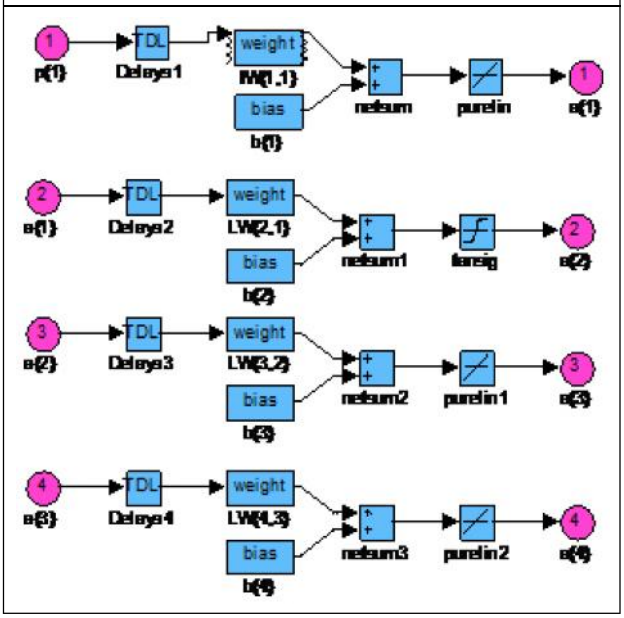
Commentsonthe Results

From the results shown in Table 2 it is clear that the ANN Output ((rev/min)) is almost equal to the Target (μ_m (rev/min)). Note that the ANN Output is different values for speed generated after the ANN training, while the Target is the values for speed generated from the MATLAB Simulink.

- Take for instance when an input voltage of 228 V was applied, from Table 2, the Target was 1523.400 rev/min while the ANN Output gave 1523.402 rev/min. The Absolute Error is 0.2.
- Also the Bar chart in Figures 6 shows little or no difference between the Target and the ANN Output confirming that the ANN model correctly maps the input to the output.This shows that Artificial Neural Network was able to predict the speed accurately since the error is minimal.
- The LM algorithm used yielded the following:
  - The four layers of the ANN architecture
  - The performance graph
  - The regression graph
  - The disparity graph

Figure 7 shows the four-layer fully connected feed-forward artificial neural network generated by the algorithm. It includes an input layer, two hidden layers and an output layer. Signal propagation is allowed only from the

Figure 7: The Four Layers of the ANN Architecture



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. 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.

The performance plot in Figure 8 shows the behaviour of the network during training, validation and testing. The performance goal was met with the best validation performance of 2.1322E-007 at epoch 1000. Epoch (number of iterations) shows the presentation of the input and/or target vectors to the network and calculates new weights and biases. The graph also shows that the mean squared error was decreasing during the learning process.

The regression plot shows the closeness or relatedness between the outputs and the target. An R value of one (R = 1) indicates a perfect relation between the target and the output. From the regression plot in Figures 9, R = 1, and the equation for the line of best fit is Y = (1)T + (0.056). This shows that the target and output values are very close. This means that the network predicted the output satisfactorily according to (Demuth *et al.*, 1992-2009).

The Disparity graph shows the level of discrepancy between the targeted output and

Figure 8: The Performance Graph

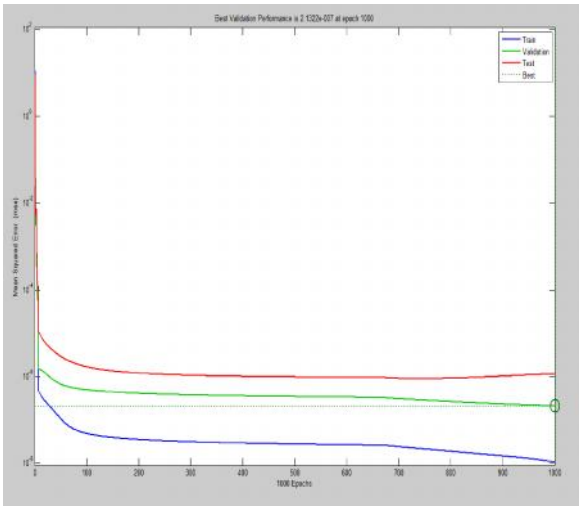


Figure 9: The Regression Graph

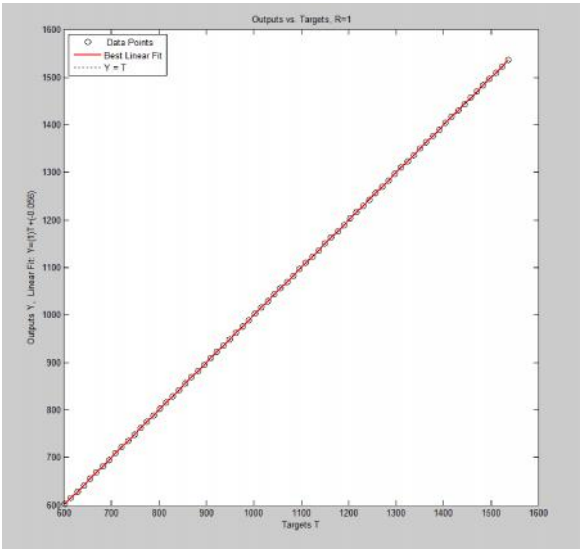
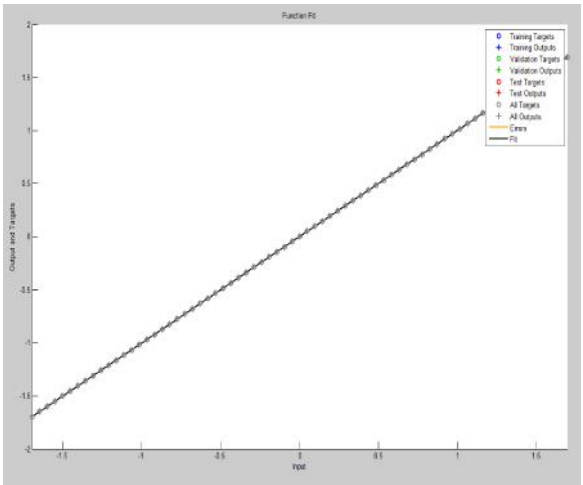


Figure 10: The Disparity Graph



the simulated (ANN) output. The disparity graph shown in Figure 10 below indicates a perfect mapping between the target and the output.

CONCLUSION

Actual experimentation on bulky power components can be expensive and time consuming but simulation offers a fast and inexpensive means to learn more about these components. In this work, the block diagram of a DC motor was developed and by using SIMULINK, a toolbox extension of the MATLAB program, the transfer function was simulated. The simulation and modeling of the DC motor was used to determine the angular speed of the DC Motor with given voltage as input parameter.

The results obtained from carrying out this project confirm the capacity of neural networks in extrapolating speed of dc motor. The Levenberg-Marquardt standard back-propagation algorithm used, proved to be faster than the usual back-propagation algorithms (Demuth *et al.*, 1992-2009). It showed a very high speed of accuracy and convergence. It appears to be the fastest method for training moderate-sized feed forward neural networks (up to several hundred weights). It also has an efficient implementation in MATLAB software, because the solution of the matrix equation is a built in function so its attributes becomes more pronounced in the MATLAB environment.

REFERENCES

1. Abdel Karim M Baareh, Alaa F Sheta and Khaled Al Khnaifes (2006), "Forecasting River Flow in the USA: A Comparison Between Auto-Regression and Neural Network", *Non-Parametric Models Journal of Computer Science*, Vol. 2, No. 10, pp. 775-780.
2. Adepoju G A, Ogunjuyigbe S O A and Alawode K O (2007), "Application of Neural Network to Load Forecasting in Nigerian Electrical Power System", *Pacific Journal of Science and Technology*, Vol. 8, No. 1, pp. 68-72.
3. Arif A Al-qassar and Mazin Z Othman (2008), "Experimental Determination of Electrical and Mechanical Parameters of DC Motor Using Genetic Elman Neural Network", *Journal of Engineering Science and Technology*, Vol. 3, No. 2, pp. 190-196.
4. Chen S and Billings S A (1992), "Neural Networks for Nonlinear Dynamic System Modelling and Identification", *International Journal of Control*, Vol. 56, No. 2, pp. 319-346.
5. Demuth H, Beale M and Hagan M (1992-2009), "Neural Network Toolbox User's Guide", Matlab User's Guide, The Mathworks Inc., Natick MA.
6. Moleykutty George (2008), "Speed Control of Separately Excited DC Motor", *American Journal of Applied Sciences*, Vol. 5, No. 3, pp. 227-233.
7. ÖzgürKisi (2004), "Multi-Layer Perceptrons with Levenberg-Marquardt Training Algorithm for Suspended Sediment Concentration Prediction and Estimation", *Hydrological Sciences-Journal-des Sciences Hydrologiques*, Vol. 49, No. 6.

8. Sen P C (1989), *Principles of Electric Machines and Power Electronics*, 2nd Edition, John Wiley and Sons Inc.
9. Steven T K (2006), "Introduction to Simulink with Engineering Applications", Orchard Publications.
10. Sung-Hoe Huh, Kyo-Beum Lee, Dong-Won Kim, Ick Choy and Gwi-Tae Park (2005), "Sensorless Speed Control System Using a Neural Network", *International Journal of Control, Automation, and Systems*, Vol. 3, No. 4, pp. 612-619.