# Modeling, Analysis and Performance Evaluation of a Universal Motor

B.Kalaiselvi

Associate Professor, Dept. of E&E, Bharath University, Chennai, Tamil Nadu, India

**ABSTRACT**: This project presents the modeling of a universal motor. The behavior of the system is studied by simulation using MATLAB SIMULINK. The speed Vs input voltage characteristics are analyzed. The speed must be monitored & controlled continuously to prevent motor from damage. The speed of the motor is controlled by adjusting input voltage/current to the motor. Two controllers namely (i) Conventional PID controller, (ii) Neural network controller using PERCEPTRON, Back Propagation Neural Network (BPN) and ADALINE Algorithm were designed to control the speed of the universal motor. The Response of the conventional PID controller and Neural Network Controller were compared and summarized.

## I.    INTRODUCTION

In the past, the rotors were similar to those of universal motors, but their brushes were connected only to each other. Transformer action induced current into the rotor. Brush position relative to field poles meant that starting torque was developed by rotor repulsion from the field poles. A centrifugal mechanism, when close to running speed, connected all commutator bars together to create the equivalent of a squirrel-cage rotor. As well, when close to operating speed, better motors lifted the brushes out of contact.The motor model is obtained and simulated in MATLAB SIMULINK. After simulation, PID Controller and Neural Network Controller are used to run the motor in desired rate.

Learning Rule: Learning rule is defined as a procedure for modifying the weights and biases of a network. (This procedure may also be referred to as a training algorithm.) The learning rule is applied to train the network to perform some particular task. Learning rules are classified into two broad categories. They are, Supervised learning and Unsupervised learning[1]

Supervised learning:

In supervised learning, the learning rule is provided with a set of examples (the training set) of proper network behavior,

$$\{p_1,t_1\},\{p_2,t_2\},\ldots\ldots,\{p_q,t_q\}$$

Where,

$p_q$- an input to the network,

$t_q$- the corresponding correct (target) output.

As the inputs are applied to the network, the network outputs are compared to the targets. The learning rule is then used to adjust the weights and biases of the network in order to move the network outputs closer to the targets. The perceptron learning rule falls in this supervised learning category.[2]

Unsupervised learning:

In unsupervised learning, the weights and biases are modified in response to network inputs only. There are no target outputs available. Most of these algorithms perform clustering operations. They categorize the input patterns into a finite number of classes. This is especially useful in such applications as vector quantization. The perceptron learning rule that we will describe shortly is capable of training only a single layer. Thus, here we will consider only one-layer networks. This restriction places limitations on the computation a perceptron can perform. Output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer function. Perceptrons can only classify linearly separable sets of vectors. If a straight line or a plane can be drawn to separate the input vectors into their correct categories, the input vectors are linearly separable. If the vectors are not linearly separable, learning will never reach a point where all vectors are classified properly. Note, however, that it has been proven that if the vectors are linearly separable, perceptrons trained adaptively will always find a solution in finite time.

Feedforward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors. The linear output layer lets the network produce values outside the range -1 to+1.
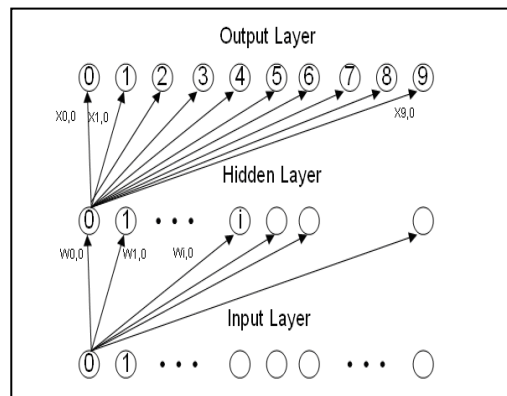


*Figure-1.  Backpropagation network architecture*

If the last layer of a multilayer network has sigmoid neurons, then the outputs of the network are limited to a small range. If linear output neurons are used the network outputs can take on any value. In backpropagation it is important to be able to calculate the derivatives of any transfer functions used. Each of the transfer functions above, tansig, logsig, and purelin, have a corresponding derivative function: dtansig, dlogsig, and dpurelin.[3-5]

Training Process

There are generally four steps in the training process:

1.        Assemble the training data
2.        Create the network object
3.        Train the network
4.        Simulate the network response to new inputs

Algorithm: Back Propagation Algorithm (BPN)

Given are P training pairs $\{z_1,d_1,z_2,d_2\ldots\ldots z_p,d_p\}$

Where $z_i$ is (I x 1), $d_1$ is (K x 1) & i= 1,2…P

Step-1: $\eta>0$, $E_{max}$ is chosen. Weights w&v are initialized at small random
  values

w is (K x J), V is (J x 1)

q $\leftarrow$ 1, P$\leftarrow$ 1, E$\leftarrow$ 0

Step-2: Training step starts here. Input is presented and the layer's outputs
  computed.

Z$\leftarrow$ $z_p$, d$\leftarrow$ $d_p$

$y_j \leftarrow f(v_j^t z)$ , for j= 1,2, …….. J

Where $v_j$ , a column vector is the $j^{th}$ row of v and,

$O_k \leftarrow f(w_k^t y)$, for k=1,2, ……K

Where $w_k$, a column vector, is the $k^{th}$ row of w.

Step-3: Error value is computed.

E$\leftarrow$ $I(d_k-O_k)^2/2 + E$, for k= 1,2,…K

Step-4: Error signal vectors $\delta_0$ and $\delta_y$ of both layers are computed. Vectors

$\delta_0$ is (K x 1),

$\delta_y$ is (J x 1).

The error signal terms of the output layer in this step are

$\delta_{ok}$  $= (d_k - O_k)[1 - O_k^2]/2$, for k= 1, 2, ……,K

The error signal terms of the hidden layer in this step are

$\delta_{yj} = (1 - y_f 2)/2 \sum_{k=1}^{k} \delta_{ok} w_{kj}$, for j= 1,2,…,J

Step-5: Output layer weights are adjusted

$w_k j + \eta\, \delta_{ok}\, y_j$, for k=1,2,…,K and

j=1,2,….J

Step-6: Hidden layer weights are adjusted.

$V_{ji} \leftarrow V_j i + \eta \delta_y j z_i$, for j=1,2,….J and

i=1,2,….I

Step-7: If P<p then p← p+1, q← q+1 and go step 2. Otherwise, go to

step 8.

Step-8: The training cycle is completed.

If E< $E_{max}$ terminate the training session.

Output weights w, V, q, E

If E > $E_{max}$, then E← 0, P← 1 and initiate the new training cycle

by goin to step2

**Adaptive Linear Neural Network (ADALINE)**

Architecture:

An important generalization of the perceptron training algorithm was presented by Widrow and Hoff as the 'least mean square' (LMS) learning procedure, also known as the delta rule.
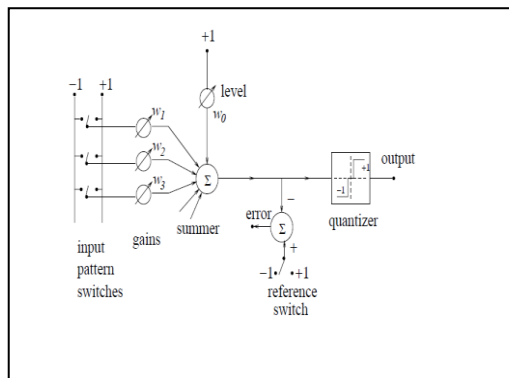


*Fig.2 Adaline-Architecture*

Then the output of the central block is defined to be:

$$y = \sum_{i=1}^{n} w_i x_i + \theta \qquad 10)$$

Where $\theta = w_0$. The purpose of this device is to yield a given value y = dp at its output when the set of values $x_p$ i , i = 1,2..... , n, is applied at the inputs. The problem is to determine the coefficients $w_i$, i = 0, 1......., n, in such a way that the input-output response is correct for a large number of arbitrarily chosen signal sets[6-8]

Networks with linear activation functions

Delta Rule:

$$y = \sum_{i=1}^{n} w_i x_i + \theta,$$

Such a simple network is able to represent a linear relationship between the value of the output unit and the value of the input units. value dp by $(d^p - y^p)$ where Yp is the actual output for this pattern. The delta-rule now uses a cost- or error-function based on these differences to adjust the weights.

$$E = \sum_p E^p = \tfrac{1}{2} \sum_p (d^p - y^p)^2,$$

The error function:
where the index p ranges over the set of input patterns
Ep represents the error on pattern p. [9]
The LMS procedure finds the values of all the weights that minimize the error function by a method called gradient descent.

$$\Delta_p w_j = -\gamma \frac{\partial E^p}{\partial w_j}$$

Where γ is a constant of proportionality. The derivative is

$$\frac{\partial E^p}{\partial w_j} = \frac{\partial E^p}{\partial y^p} \frac{\partial y^p}{\partial w_j}.$$

$$\frac{\partial y^p}{\partial w_j} = x_j$$

Because of the linear units

$$\frac{\partial E^p}{\partial y^p} = -(d^p - y^p)$$

$$\Delta_p w_j = \gamma \delta^p x_j$$

Where $\delta^p = d^p - y^p$ is the difference between the target output and the actual output for pattern p.

## II.    RESULTS AND DISCUSSIONS

**Open loop system response**
The motor model is simulated in MATLAB SIMULINK by giving step as an input and response is obtained. It is observed that the system without controller does not settles, the response obtained is uncontrolled.
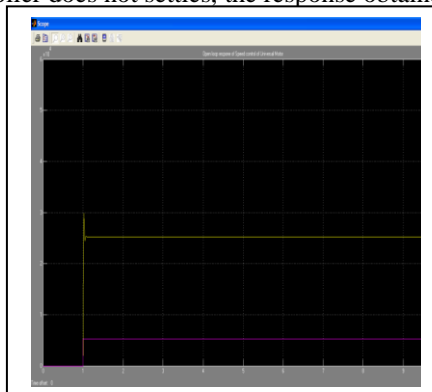


*Figure 3.Open loop system response*

In the above system the set point given to the motor is 5300 rpm.
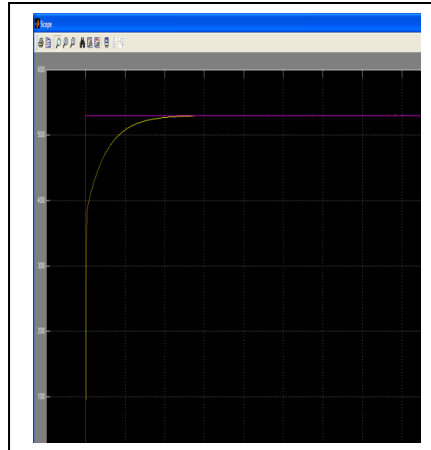So the response obtained is the uncontrolled response.



*Figure.4. Closed loop system response*

**Closed loop system with PID**
After introducing PID controller the system settles at **3.5 seconds**.
**PERCEPTRON Network Response**
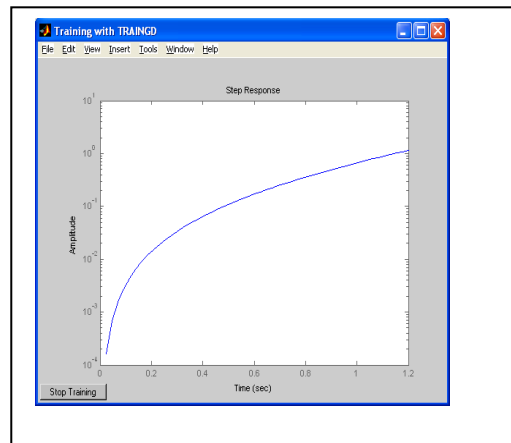**Back Propagation Neural Network (BPN) Response**



*Figure 5.3 Response of the BPN network*

The response settles at **2.5 seconds.** The BPN network is trained to achieve the desired response more quicker than the PID controller and perceptron network. updated and again the network is trained till it achieves the target. **Adaptive**

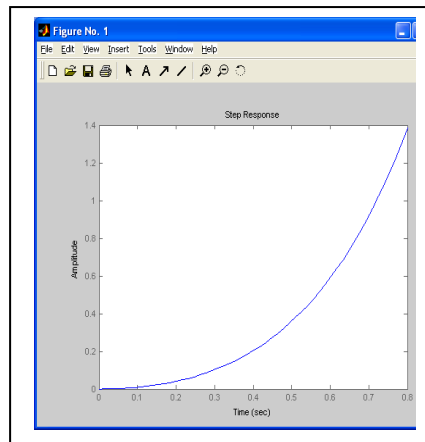**Linear Neural Network (ADALINE) Response**



*Figure 6. Response of ADALINE Network*

In ADALINE the target set is 1. The activation used here is bipolar activation function.

**DISCUSSIONS**

*Table.1 Controller performance evaluation (PID, BPN):*

| Parameters | Open Loop System | Closed Loop System( PID) | Neural Controller (BPN) |
|---|---|---|---|
| Rise time (s) | 95 | 3.325 | 2.37 |
| Peak time(s) | 97.25 | 3.39 | 2.42 |
| Settling time(s) | 100 | 3.5 | 2.5 |

**V. CONCLUSION**

After evaluating the performance of the conventional controller (PID) and Neural Network Controller, it is concluded that the Neural Network Controller gives quicker response than the conventional controller (PID).

**FUTURE PLAN**

**Fuzzy Logic Control**

A fuzzy control system is a control system based on fuzzy logic—a mathematical system that analyzes analog input values in terms of logical variables that take on continuous values between 0 and 1, in contrast to classical or digital logic, which operates on discrete values of either 1 or 0.

In this project, instead of  PID controller and Neural Network Con troller,  the fuzzy logic controller may be implemented to enhance the system performance and to minimize the settling time.

## REFERENCES

[1]        Hwi-Beom, S., and Park, J., 2012, "Anti-Windup PID Controller With Integral StatePredictor for Variable-Speed Motor Drives," IEEE Transactions on Industrial Electronics, vol. 59, no. 3, pp. 1509-1516.

[2]        Lydia Caroline M., Vasudevan S.,"Growth and characterization of l-phenylalanine nitric acid, a new organic nonlinear optical material", Materials Letters,  ISSN : 0167-577X, 63(1) (2009) pp. 41-44.

[3]        Langeswaran K., Gowthamkumar S., Vijayaprakash S., Revathy R., Balasubramanian M.P., "Influence of limonin on Wnt signalling molecule in HepG2 cell lines", Journal of Natural Science, Biology and Medicine, ISSN : 0976-9668, 4(1) (2013) PP. 126-133.

[4]        Grop, H., 2006, "Modelling of a Universal Motor with Speed Control," Ph.D Thesis, Royal Institute of Technology, Stockholm.

[5]        Jayalakshmi T., Krishnamoorthy P., Ramesh Kumar G., Sivamani P., "Optimization of culture conditions for keratinase production in Streptomyces sp. JRS19 for chick feather wastes degradation", Journal of Chemical and Pharmaceutical Research, ISSN : 0975 – 7384, 3(4) (2011) PP.498-503.

[6]         Jebaraj S., Iniyan S., "Renewable energy programmes in India", International Journal of Global Energy Issues, ISSN :  0954-7118, 26(4Mar) (2006) PP.232-257.

[7]        Jose. C., Isel, R., and Cons. Emídio, N.,  "Universal Motor NN Modelling," Proceedings of the 4th WSEAS International Conference on Systems Theory and Scientific Computation

[8]        Gopalakrishnan K., Prem Jeya Kumar M., Sundeep Aanand J., Udayakumar R., "Thermal properties of doped azopolyester and its application", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S6) (2013) PP. 4722-4725.

[9]        Shi, K., Chan, T., Wong, Y., and Ho. S., 1999, "Modelling and Simulation of the three-phase Induction motor using Simulink," Int. J. Elect. Enging. Educ., vol. 36, pp. 163–172.

[10]        D.Kalaivani, Mrs.M.Indirani & Dr.A.Mukunthan, A Theoretical Study of Primary Nucleation Kinetics of L-Histidine Bromide: Semi Organic Optical Single Crystal, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 4192-4197, Vol. 2, Issue 9, September 2013

[11]        D.Prakash and Dr. A.Mukunthan, A Theoretical Study of Internal Pressure And Free Volume for Single Molecule of a Sample Liquid, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 7252-7257, Vol. 2, Issue 12, December 2013

[12]        Dr. A. Mukunthan & Ms.S.Sudha, FTIR Spectroscopic Features of Blood Serum of Diseased and Healthy Subjects (Animals), International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,  pp 2035-2040 Vol. 2, Issue 6, June 2013

[13]        Dr. A.Mukunthan, A Survey of Applications of Laser in Dermatology – Medical Physics, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753 ,pp 33-36, Vol. 1, Issue 1, Nov 2012

[14]        Dr. M. Ganesan, Mergers and Acquisitions, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 9081-9085, Vol. 3, Issue 2, February 2014

[15]        Dr.G.Brindha, A Study on Latest Management Governance Techniques in Indian Companies, International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753,pp 284-292, Vol. 2, Issue 1, January 2013