Application of Computational Intelligence in Motor Modeling

Yousuf Ibrahim Khan, Shahriar Rahman, Debasis Baishnab,

Mohammed Moaz, S.M. Musfequr Rahman

Department of Electrical and Electronic Engineering,
American International University-Bangladesh

ABSTRACT
Modeling is very important in the field of science and engineering. Modeling gives us an abstract and mathematical description of a particular system and describes its behavior. Once we get the model of a system then we can work with that in various applications without using the original system repeatedly. Computational Intelligence method like Artificial Neural Network is very sophisticated tool for modeling and data fitting problems. Modeling of Electrical motors can also be done using ANN. The Neural network that will represent the model of the motor will be a useful tool for future use especially in digital control systems. The parallel structure of a neural network makes it potentially fast for the computation of certain tasks. The same feature makes a neural network well suited for implementation in VLSI technology. Hardware realization of a Neural Network (NN), to a large extent depends on the efficient implementation of a single neuron. In this paper only a motor model is presented along with some neural networks those will mimic the motor behavior acquiring data from the original motor output.

General Terms
Computational Intelligence, Artificial Neural Networks.

Keywords
Modeling, ANN, DC Servo Motor, MATLAB, Simulink.

1. INTRODUCTION
A system comprises multiple views such as planning, requirement (analysis), design, implementation, deployment, structure, behavior, input data, and output data views. A system model is required to describe and represent all these multiple views. Modeling is the process of producing a model; a model is a representation of the construction and working of some system of interest. A model is similar to but simpler than the system it represents. One purpose of a model is to enable the analyst to predict the effect of changes to the system. On the one hand, a model should be a close approximation to the real system and incorporate most of its salient features. On the other hand, it should not be so complex that it is impossible to understand and experiment with it. A good model is a judicious tradeoff between realism and simplicity [1]. How is it possible to forecast electrical load patterns for tomorrow, with access solely to today's load data? How is it possible to predict the dynamic behavior of a spacecraft that has yet to be built? And how is it possible to analyze the motion and path-planning of a robotic rover that will explore on Mars surface in the next year? The answer is computer simulations based on mathematical models and sets of equations that describe the underlying physical properties [2]. Not only do modeling and simulation help provide a better understanding of how real-world systems function, they also enable us to predict system behavior before a system is actually built and analyze systems accurately under varying operating conditions. Modeling is an integral part of Control Systems and Motor drives. If we can find the model of a motor and give it a portability and hardware realization then different experiments can be done with that model without even accessing the motor repeatedly. It will show how that particular motor will act in different applications and control systems. Computational Intelligence techniques like Artificial Neural Networks can be used to build such models and can give us chance to use the motor behavior in various analog-digital mixed signal applications where integrated circuits play vital role (ASIC, FPGA, SoC). Hardware realization of neural networks are possible [3]. FPGA-based reconfigurable computing architectures are suitable for hardware implementation of neural networks. FPGA realization of ANNs with a large number of neurons is still a challenging task but will be possible very soon. Then researchers will be able to work with those models. It will be like an entire original control system or controller on a chip [4]. Here only the modeling part has been shown using one particular and popular motor-DC Servo Motor.

2. COMPUTATIONAL INTELLIGENCE AND ARTIFICIAL NEURAL NETWORK
Computational intelligence (CI) is a set of Nature-inspired computational methodologies and approaches to address complex problems of the real world applications to which traditional (first principles, probabilistic, black-box, etc.) methodologies and approaches are ineffective or infeasible.

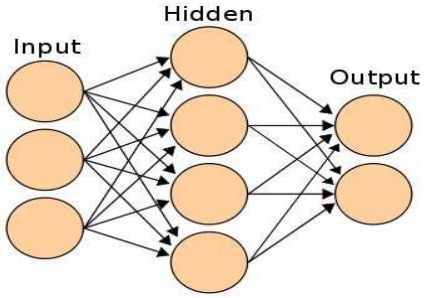

Fig 1: An artificial neural network is an interconnected group of nodes

It primarily includes Fuzzy logic systems, Neural Networks and Evolutionary Computation. Artificial neural nets (ANNs) can be regarded as a functional imitation of biological neural

networks and as such they share some advantages that biological organisms have over standard computational systems. The main feature of an ANN is its ability to learn complex functional relations by generalizing from a limited amount of training data. Neural nets can thus be used as (black-box) models of nonlinear, multivariable static and dynamic systems and can be trained by using input–output data observed on the system [5].

An Artificial Neural Network (ANN) is a mathematical or computational model based on the structure and functional aspects of biological neural networks. It consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. An ANN mostly is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase [6].

3. DC SERVO MOTORS

The DC servo motors are presently used worldwide in applications such as X-Y tables, factory automation, coil winders, labeling equipment, machine tool, insertion machines, robotics [7], pick and place, packaging, converting equipment, assembly equipment and laboratory equipment. Due to their importance, the design of controllers for these systems has been an interesting area for researchers from all over the world. For these purposes the modeling of DC servomotor system is an interesting area that still offers multiple topics for research, especially after the discovery of Artificial Neural Networks (ANN) and their possible usage for intelligent control purposes. From this point of view and the importance of having efficient servomotor systems and experimenting with servomotors, the use of ANN plays a vital role in modeling Servo Motors. A Servomotor system consists of different mechanical and electrical components, the different components are integrated together to perform the function of the servomotor, Figure 2 below shows a typical model of a servomotor system [8].

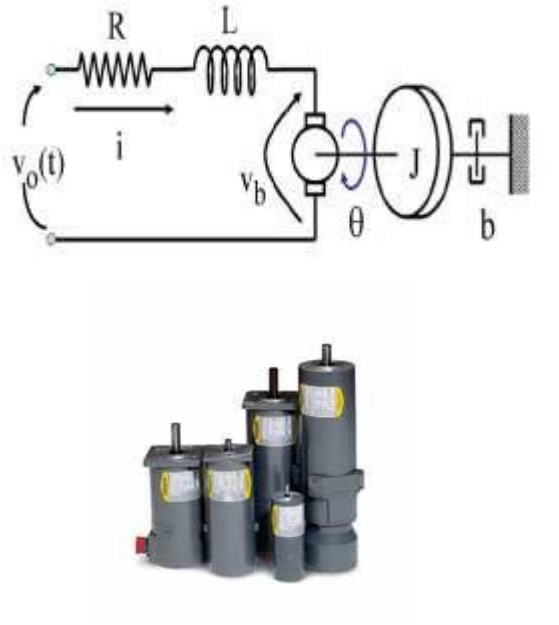

Fig 2: Servo Motor Circuit Diagram and original DC Servo Motors (Courtesy of Baldor Servo Motors)

It's clear that a basic servomotor has two main components, the first is the electrical component; which consists of resistance R, inductance L, input voltage $V_0(t)$ and the back electromotive force V_b. The second component of the servomotor is the mechanical part, from which we can get the useful mechanical rotational movement at the shaft. The mechanical parts are the motor's shaft, inertia of the motor and load inertia J and damping b. θ Refers to the angular position of the output shaft which can be used later to find the angular speed of the shaft ω.

DC Servomotors have good torque and speed characteristics; also they have the ability to be controlled by changing the voltage signal connected to the input. These characteristics made them powerful actuators used everywhere. Before modeling the motor several nonlinearities are also a matter of concern. Modeling these nonlinearities will make the motor look like an original motor. One important non-linear behavior in servomotors is the saturation effect, in which the output of the motor cannot reach the desired value. For example, if we want to reach a 110 rpm angular speed when we supply a 12 volt input voltage, but the motor can only reach 100 rpm at this voltage. The saturation effect is very common in almost all servomotor systems. Other non-linear effect is the dead zone; in which the motor will not start to rotate until the input voltage reaches a specific minimum value, which makes the response of the system slower and requires more controllability. A mathematical type of non-linear effect found in the servomotors is the backlash in the motor gears. Some of the servomotors use internal gears connections in order to improve their torque and speed characteristics, but this improvement comes over the effect in the output speed and position characteristics. The goal in this paper is to create a model which can mimic a practical servo motor and which can be used further in several applications using artificial neural networks.

3.1 Mathematical Description

Recalling the DC servomotor diagram from Figure 2, the transfer function of the DC servomotor can be derived using Kirchhoff's voltage law and Laplace transforms as the following:

$$V_o = V_m = R_m i_m + L_m \frac{di}{dt} + V_b \quad (1)$$

The Back-electromotive force (emf) Vb can be found by using the equation shown below:

$$V_b = K_m \frac{d\theta}{dt} = K_m \omega_m \quad (2)$$

Where V_b is the induced voltage, K_m is the motor torque constant, and ω_m is the angular rotating speed. It can be seen that ω_m can be calculated by the Equation shown below:

$$\omega_m = \dot{\theta} \quad (3)$$

And using Laplace Transform

$$\omega(s) = s\theta(s) \quad (4)$$

The main concern in this stage is to control the angular rotating speed ω_m, by controlling the input voltage V_m.

Where, J = moment of inertia of the rotor; b = dampening ratio of the mechanical system; T = motor torque; I = current; Vm = back emf; θ = shaft position; K = electromotive force constant; ωm = measured angular speed; R = motor armature resistance; L = inductance; V = source voltage.

The transfer function of the output angular speed is derived using Laplace transform using the second order system equation:

$$G(s) = \frac{\omega_n}{s^2+2\xi\omega s+\omega^2} \quad (5)$$

The resulting transfer function:

$$\frac{\omega}{V} = \frac{K}{(Js+b)(Ls+R)+K^2} \quad (6)$$

From Equation 4, the relationship between the angular position and the speed can be found by multiplying the angular position by 1/s. Figure 3 shows the block diagram which represents the servomotor system using MATLAB and SIMULINK.

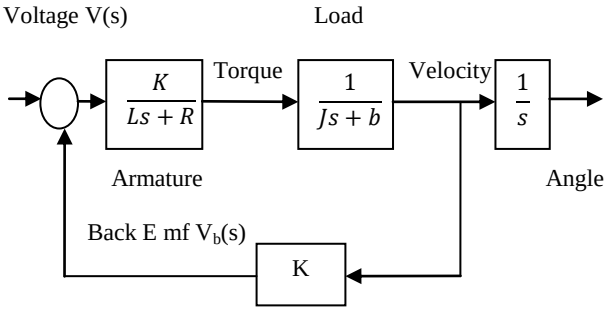

Fig 3: Mathematical block diagram of Servo Motor

The parameter values used in the SIMULINK model were taken from a practical servomotor (Baldor Servo Motors). The parameters are given in Table 1:

Table 1. DC Servo Motor parameter values

Parameter	Value
Moment of Inertia J	0.0062 N.m.s^2/rad
Damping Coefficient b	0.001 N.m.s/rad
Torque Constant Kt	0.06 N.m/A
Electromotive Force Constant Ke	0.06 V.s/rad
Electrical Resistance R	2.2 Ohms
Electrical Inductance L	0.5 Henry

From the SIMULINK model it is clear that we can divide the transfer function into two transfer functions, the first one is the electrical transfer function; which consists of the electrical resistance and the inductance. The second transfer function is the mechanical transfer function; which consists of the moment of inertia and the damping coefficient. By dividing the transfer function into the two parts mentioned above, we can add the non-linear parameters to the system to see their effect. The dead zone value was 1.5 volts, which is very common in this size of servomotors (12 Volts).

4. SIMULINK MODEL

Simulink is a software package for modeling, simulating, and analyzing dynamical systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multi-rate, i.e., have different parts that are sampled or updated at different rates. Models here are hierarchical. This approach provides insight into how a model is organized and how its parts interact. After someone defines a model, he/she can simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in MATLAB's command window.

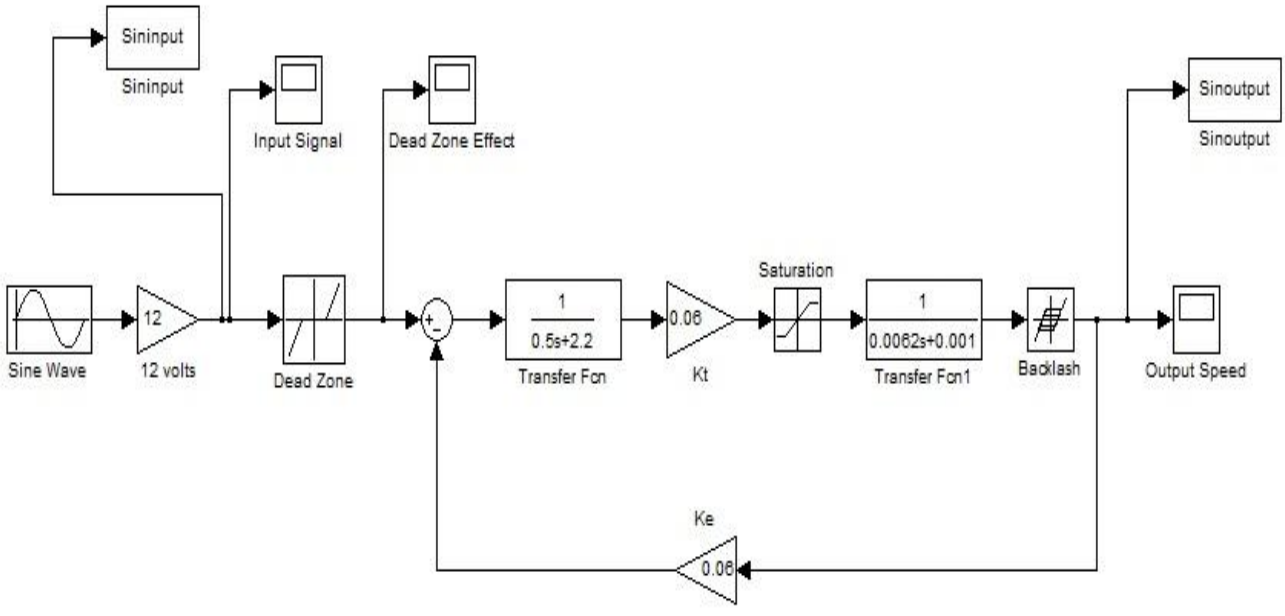

Fig 4: DC Servo Motor Model (Simulink Environment)

In this paper Simulink 7.0 (MATLAB R2007b) was used for modeling purpose. At first the DC Servo Motor model was created according to the mathematical description of section 3. The model (Fig 4) was derived from the Servomotor speed transfer function (Equation 6). Simulink's "To Workspace" block was used to take input and output value of the motor model. The simulation took 10 sec.

Here, fig 5 shows the input and fig 6 shows the dead zone of the motor after simulation. The main idea was to take these input and output points, feed the inputs into neural networks. The neural networks are very good at showing the outputs if sufficient input points are provided. This way the neural network itself will become like the DC Servo motor.

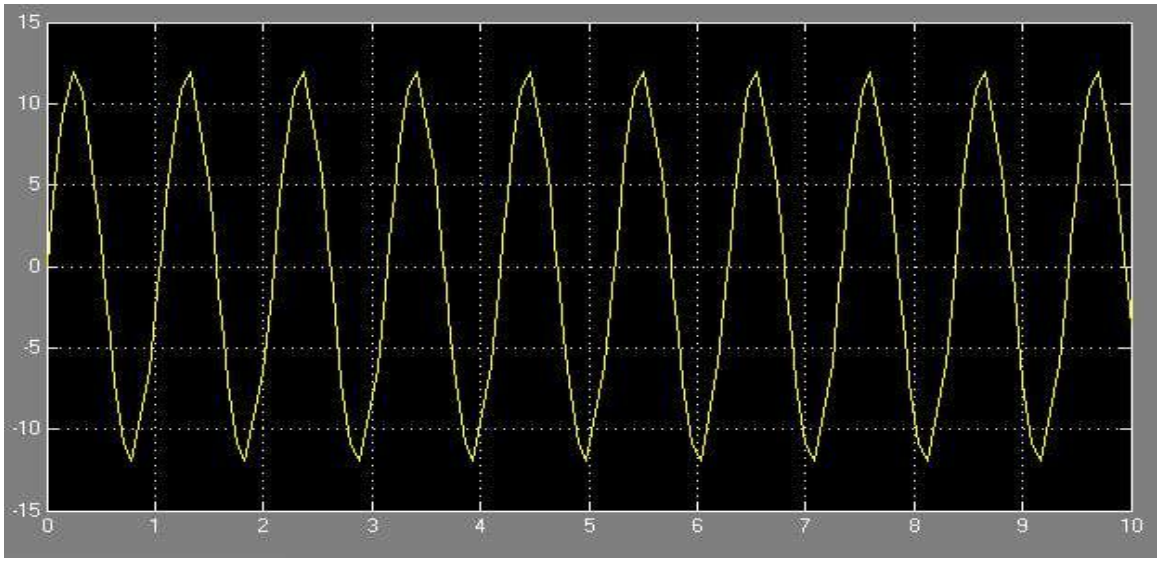

Fig 5: Input Signal (Sine wave of 6 rad/s) of the Motor (Using Scope)

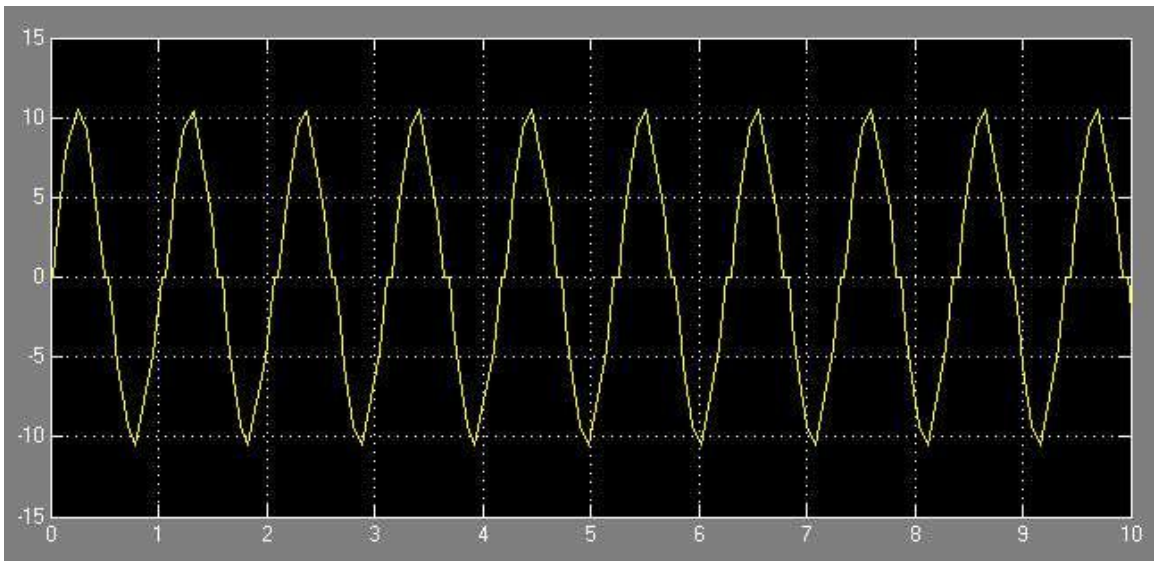

Fig 6 : Dead Zone Effect in Motor which is limiting output to 10.5 V (Using Scope)

5. NN IN MODELING

Using MATLAB command window several neural networks were created and examined. Before creating networks data were normalized between values -1 to 1 to avoid convergence problem writing the following lines in a **.m** file:

```
dataout = datain - min(datain(:));
dataout = (dataout/range(dataout(:)))*(maxval-minval);
dataout = dataout + minval;
```

The inputs were taken into P and the outputs were taken into T matrix. T was the target matrix. Hidden layer had 10 neurons with 'tansig' function and layer had 'purelin' function. The maximum number of epochs used was 1000, which is the maximum number of trials the training method can apply before stopping.

The training goal was to reach a minimum error value of 0.00005; this error value was between the input signal and the generated output. The learning rate was chosen to be 0.05. These parameters were chosen to best fit the performances of the network, more restrict parameters may take the network to have unstable response and it may cause the training method to diverge. Thus different network architecture's were created and examined. The following results were found:

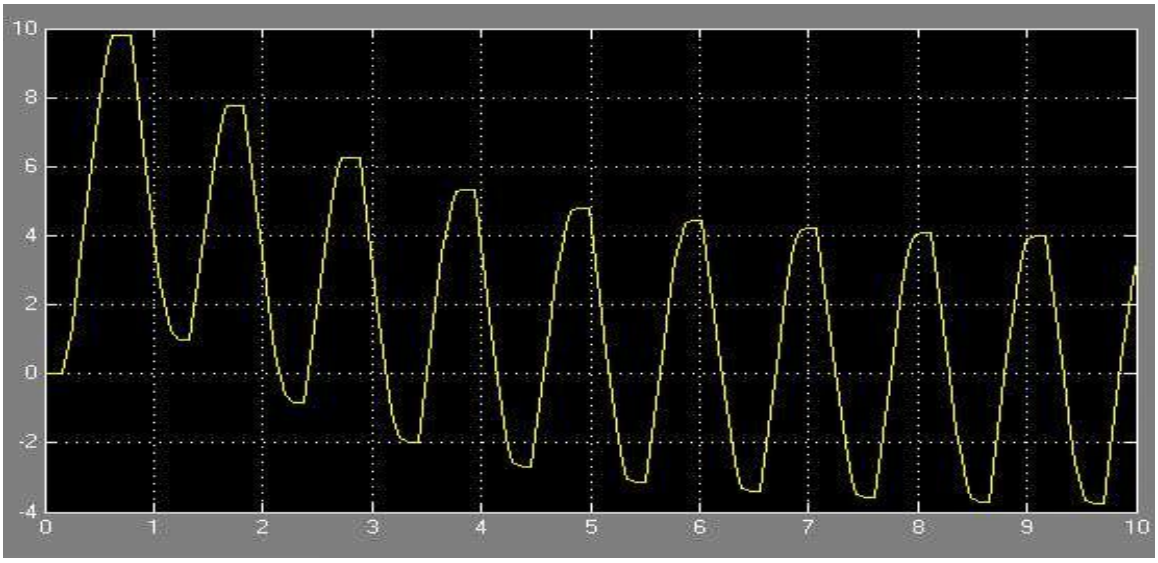

Fig 7 : Output of Motor (Using Scope)

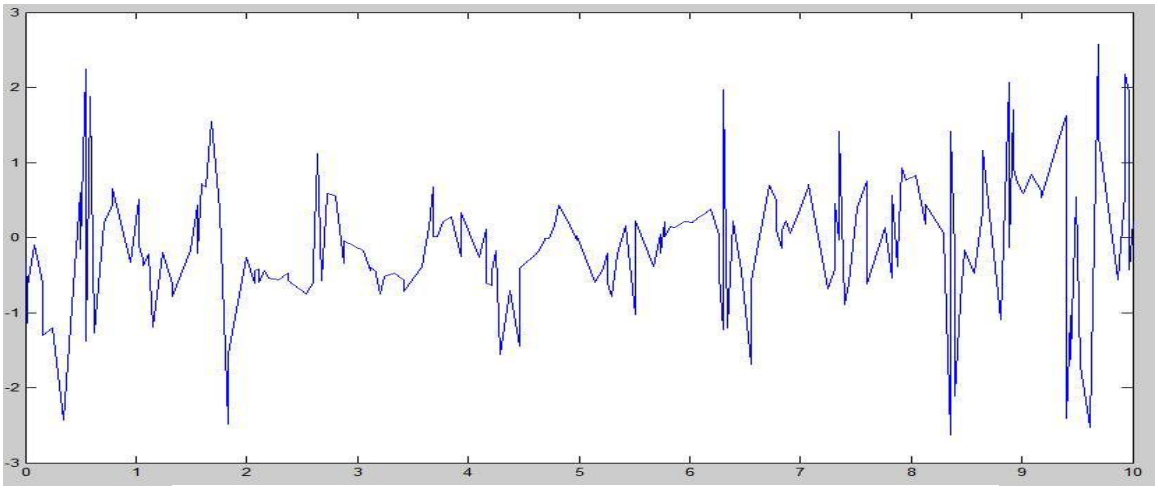

Fig 8: Output of Motor using NN (Before Training)

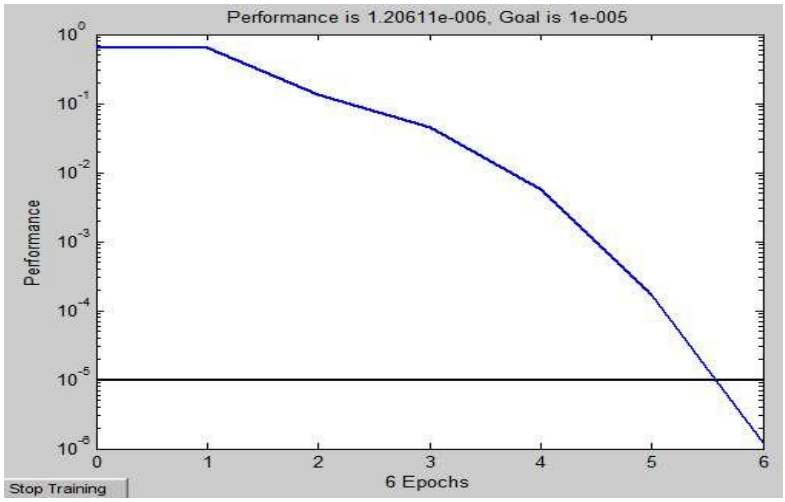

Fig 9: Performance goal met

Comparing figure 7 and 8, it is clear that the neural network couldn't reflect the exact output. So to mimic the behavior correctly training parameters were given in the command window.

So after the successful training the performance goal met fast (6 epochs). From the following figures it can be shown that the network has realized the problem well enough.

Comparing figure 7 and 10, it is clear that the network was able to mimic the original behavior almost successfully. The human eye would not be able to find the difference between two outputs but if we plot both original motor and NN motor model output (overlapped) then some clear difference will come out (Fig 11) which can be further minimized using algorithms like Genetic Algorithm, Particle Swarm Optimization, and Ant Colony Optimization [9].

Several architectures were examined and feed-forward backpropagation with 'trainlm' gave the lowest error (MSE). These previous figures were all taken from feed-forward backpropagation network which was best network for this problem. Fig 12 shows problems with other Neural Networks.

The following table shows the complete result from all networks:

Table 2. Complete Result from different networks

Neural Network	Training Function	MSE
Feed-Forward Backprop	Trainlm	1.20611e-6 (goal met in 6 epochs)
Elman Backprop	Traingdx	0.0474668 (goal didn't met in 1000 epochs)
Elman Backprop	Traingd	0.137859 (goal didn't met in 1000 epochs)
Cascade Forward Backprop	Traingda	0.0116465 (goal didn't met in 1000 epochs)

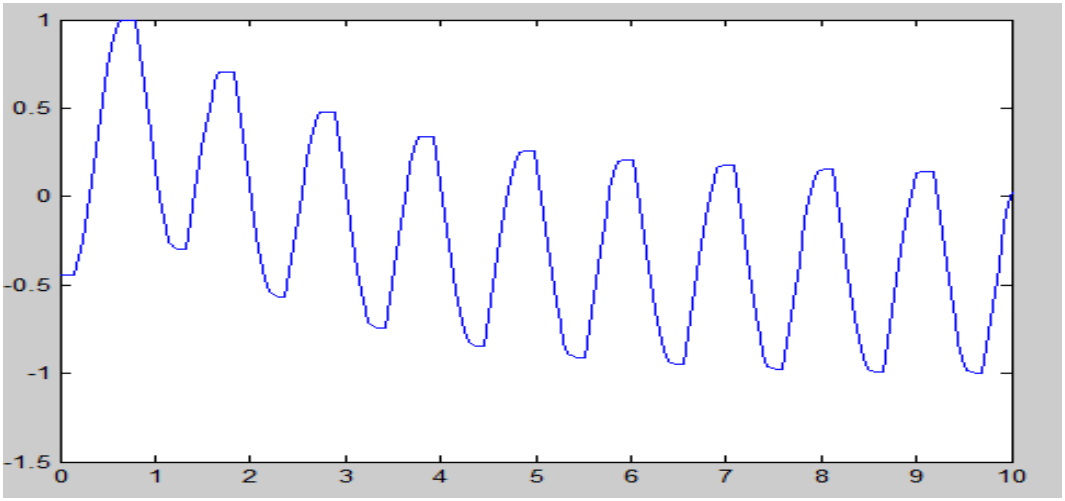

Fig 10: Output of Motor (After Training) (Feed-Forward Backprop with

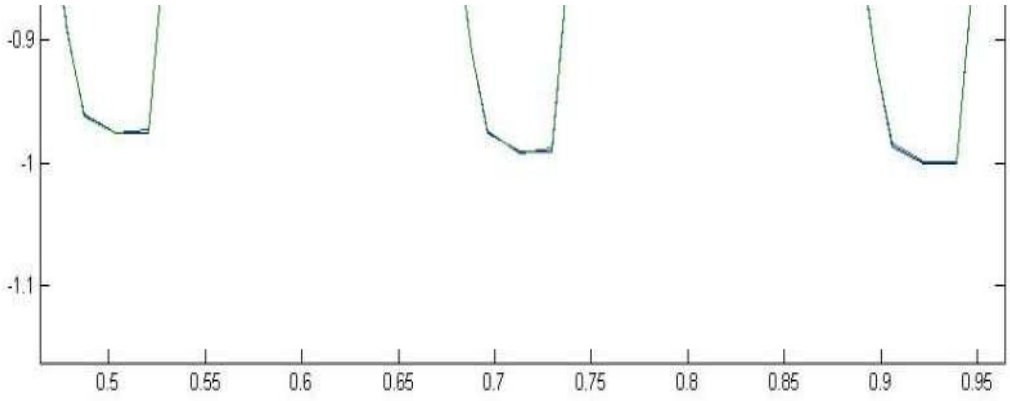

Fig 11: Overlapped Outputs showing minor errors (Original and NN

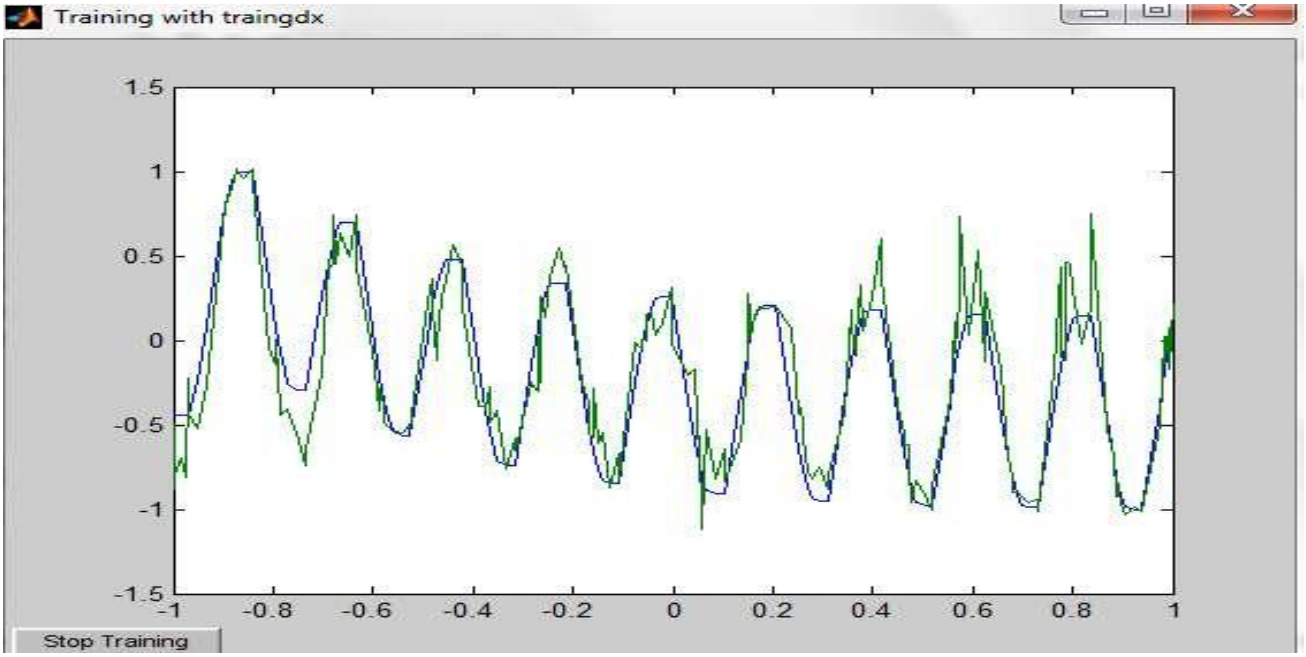

Fig12: Unlike Feed-Forward Backprop some network showed poor result
(Original and NN Elman backprop model with Traingdx; Elman NN in Green)

6. CONCLUSION

This paper presents modeling of a motor (DC Servo Motor) and Artificial Neural Networks in mimicking its exact behavior. The result showed that feed-forward backpropagation neural network worked very well compared to other networks in copying the behavior of the original motor. Meta-heuristic algorithms like PSO, ACO, GA [10][11][12] could further minimize the error.

Real time applications are possible only if low cost high-speed neural computation is made realizable. Towards this goal numerous works on implementation of Neural Networks (NN) have been proposed [13]. Neural networks can be implemented using analog or digital systems.

The digital implementation is more popular as it has the advantage of higher accuracy, better repeatability, lower noise sensitivity, better testability, higher flexibility, and compatibility with other types of preprocessors. The digital NN hardware implementations are further classified as FPGA-based implementations, DSP-based implementation, ASIC-based implementations[14-15].

DSP based implementation is sequential and hence does not preserve the parallel architecture of the neurons in a layer. ASIC implementations do not offer re-configurability by the user. In future this model can be further implemented in VLSI architectures like FPGA using Hardware Description Languages and that particular IC or chip could represent the original motor in various innovative applications. FPGA is a suitable hardware for neural network implementation as it preserves the parallel architecture of the neurons in a layer and offers flexibility in reconfiguration.

Because of superior agility and flexibility, robotic rovers are very popular and widely used by NASA in their various planetary missions (Lunar missions, Mars missions) for surface navigation. DC Servo Motors play a vital role here. Working with original DC Motors time and time again in planetary experiments before final missions might become costly. In these case the above presented technique could become handy. Present day systems are large and complex. Putting their behavior inside Integrated Circuits using this kind of technique could become a useful tool for professional engineers.

7. REFERENCES

[1] Law, A.M., and W.D. Kelton. 1991. Simulation Modeling and Analysis, Second Edition, McGraw-Hill.

[2] Bertil Gustafsson, Fundamentals of Scientific Computing, Texts in Computational science and Engineering, Springer, 1st Edition, 2011.

[3] Amos R. Omondi, Jagath C.Rajapakse, FPGA Implementations of Neural Networks, Springer, 1st Edition, 2006.

[4] David A. Gwaltney, Kenneth D. King and Keary J. Smith, "Implementation of Adaptive Digital Controllers on Programmable Logic Devices", NASA Marshall Space Flight Center, Huntsville, AL.

[5] Ajith Abraham, Recent Advances in Intelligent Paradigms and Applications, Physica-Verlag HD, 1st Edition, 2003.

[6] Yousuf Ibrahim Khan, et al. "Artificial Neural Network based Short Term Load Forecasting of Power System", International Journal of Computer Applications,Vol.30, No. 4, pp.1-7, September 2011. Published by Foundation of Computer Science, New York, USA.

[7] O.A. Jasim, A.Z.Mansoor and M.R. Khalil, International Electrical Engineering Journal (IEEJ), Vol.2, No.3, pp.555-559, ISSN 2078-2365, 2011.

[8] Yahia Makableh, "Efficient Control of DC Servomotor Systems Using Backpropagation Neural Networks", Master's thesis, Georgia Southern University, 2011.

[9] E. Eiben and J. E. Smith, Introduction to Evolutionary Computing. Natural Computing Series. Massachusetts Institute of Technology Press. Springer. Berlin. (2003).

[10] M. Carvalho and T.B. Ludermir, Hybrid Training of Feed-Forward Neural Networks with Particle Swarm Optimization, I. King et al. (Eds.): ICONIP 2006, Part II, LNCS 4233, pp. 1061–1070, 2006.

[11] C. Blum and K. Socha, "Training feed-forward neural networks with ant colony optimization: An application to pattern classification", Fifth International Conference on Hybrid Intelligent Systems (HIS'05), pp. 233-238, 2005.

[12] E. Alba and J.F. Chicano, "Training Neural Networks with GA Hybrid Algorithms", K. Deb(ed.), Proceedings of GECCO'04, Seattle, Washington, LNCS 3102, pp. 852-863, 2004.

[13] Leonardo Maria Reyneri "Implementation Issues of Neuro-Fuzzy Hardware: Going Towards HW/SW Codesign" IEEE Transactions on Neural Networks, vol.14, no.1, pp. 176-194, 2003.

[14] Y.J.Chen, Du Plessis, "Neural Network Implementation on a FPGA ", Proceedings of IEEE Africon, vol.1, pp. 337-342, 2002.

[15] Sund Su Kim, Seul Jung, "Hardware Implementation of Real Time Neural Network Controller with a DSP and an FPGA ", IEEE International Conference on Robotics and Automation, vol. 5, pp. 3161-3165, April 2004.