Neural, Parallel, and Scientific Computations 20 (2012) 101-110

INTELLIGENT CRUISE CONTROL USING AN EFFICIENT PID MODEL NEURAL NETWORK

YAHIA MAKABLEH AND FERNANDO RIOS-GUTIERREZ

Mechanical and Electrical Engineering Department,
Georgia Southern University
Statesboro, GA 30460

ABSTRACT: Servomotor systems are known to have nonlinear parameters and dynamic factors, such as backlash, dead zone and Coulomb friction, making the system hard to control using conventional control methods as PID controllers. Also, the dynamics of the system will change when changing the load, which will add more complexity and nonlinearity to the system. Nontraditional, intelligent control techniques such as Neural Networks (NN), Genetic Algorithms and Fuzzy Logic methods have been used in many applications in order to solve the problem related to these nonlinear systems. This paper presents a NN controller for intelligent cruise control (ICC) of vehicles, in which the model of a servomotor is used as the base for its implementation. In particular, we are using a multilayer neural network to build a model that mimics the function of a DC servomotor system; also we are using a second neural network to control the modeled network. The main objective of the ICC is to achieve automatic vehicle speed control in a safe, reliable and smooth way. The proposed NN controller will be able to deal with the nonlinear parameters and dynamic factors involved in the system and hence the proper control of the servomotor output speed and position. Off-line simulation using MATLAB is used to show final results, and to compare them with a conventional PID controller results for the same model.

AMS (MOS) Subject Classification. 35K60, 35K57

1 INTRODUCTION

Servomechanisms are one important part of automatic control systems, they are considered to be the heart of self-adjusting systems and they are used in multiple engineering applications. Servomotors are considered one type of actuators that uses servomechanisms in their functionality. Servomotors have a wide range of applications, they can be found in cruise control systems, robotic arms, production lines and other applications. Although, different control schemes were implemented for servomotors, they still suffer from low efficiency due to several nonlinear parameters associated with this type of motors [Kandel E, Schawrtz JH, Jessel TM; 2000.Miller, W.T, Sutton, R.S. and Werbos; 1990]. Also different dynamic factors play a vital role to lower the efficiency of the servomotor, such as sudden changes of the motor load or any other changes in the working environment [Hagan T. Martin and Howard B. Demuth; 1996].

Servomotors can be classified as either: single output systems or multi output systems, depending on the application they are used for. The two outputs that are of importance in servomotors are angular speed and angular position. From the control point of view each one of these outputs can be treated separately or related to the other [Mahanijah Md Kamal and NasirahMamat; 2009]. Therefore, two control schemes can be developed for controlling servomotors; while the angular position is an important part of the control scheme we will focus on the angular speed scheme.

Different non-linear parameters are associated with servomotors such as Saturation, Coulomb Friction, Dead Zone and Backlash. These parameters have great impact on the motor's performance. The Saturation effect may lead to overshoot and may limit the output from reaching the desired values; which may cause the system to be unstable in

Received September 15, 2011

some cases [Jun Oh Jang; 2007]. Backlash also can be found on servomotors when the gears used do not mesh completely, which leads to bad effects and unsmooth system functionality [Nizar J. Ahmad, Adel A. Zerai and Muhammad Al- Mumin; 2003]. Dead zone is another non-linear parameter has a great effect on the system, making the servomotor responds to the input voltage after reaching a certain threshold value; which in turn leaves useful voltage levels not used [Norman S. Nise; 2008].

Beside the nonlinearities found within servomotors, the dynamic changes in the working environment also play a vital role in the motor's performance, such as a sudden change in the motor load, which may lead to overshoot, drain more power to meet the change or over loading the system; hence may lead to either a low efficiency or failure in the system. Due to the importance of servomotors in our daily life and their low efficiency, conventional control techniques such as PID controllers may not be the best control schemes to use with servomotors, since they cannot eliminate the nonlinear effect from the system and improve the overall system efficiency. Therefore, non-traditional control techniques, such as intelligent control schemes using neural networks, fuzzy logic and genetic algorithms can be used to serve as the best choice of control method.

In summary, we are proposing a Neural Network (NN) based intelligent system to implement both the servomotor and the controller. This intelligent system is composed of two feed forward neural networks (FFNN):one of them mimics the function of the servomotor and the other one is used to build the controller itself. The proposed NN controller will be able to eliminate the nonlinear effect and improve the overall system efficiency.

2 DC SERVOMOTOR MODEL

The mathematical model of the servomotor can be derived based on the electrical and mechanical circuit diagram of the motor shown in figure 1 [Norman S. Nise; 2008]:

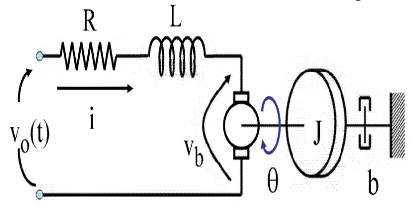

Figure 1.DC Servomotor circuit diagram.

Based on the circuit diagram we can derive the equation of the supplied voltage $V_o(t)$ as:

$$V_{o(t)} = Ri_{(t)} + L\frac{di}{dt} + V_b \tag{1}$$

The induced voltage V_b relates to the angular position through the following equation:

$$V_b = K\frac{d\theta}{dt} \tag{2}$$

Where V_b is the induced voltage, ω_m angular motor speed, K is the motor torque constant, J is the total moment of inertia for the load and the motor, and b is the mechanical damping ratio. For practical purposes we have used the following motor parameters [Robert N. Bateson; 1999]:

Parameter	Value
Moment of Inertia J	0.0062 N.m.s²/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	2.2 ohms
Electrical Inductance	0.5 Henry

Table 1. DC Servomotor Parameters.

The transfer function of the angular speed can be derived from equations 1 and 2, and taking Laplace transform we get:

$$\frac{\omega_m}{V} = \frac{K}{s((Js+b).(Ls+R)+K^2))} \tag{3}$$

The angular speed transfer function contains both the electrical and mechanical parameters of the servomotor. Although, this transfer function represents the complete system, we have divided it into two functions for the simulation purposes, one represents the electrical part and the other one represents the mechanical part, the block diagram of the two systems is shown in figure 2.

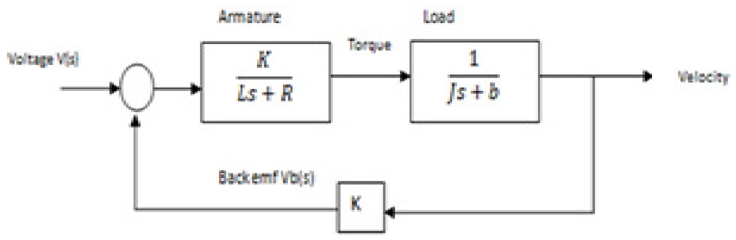

Figure 2.Servomotor electrical and mechanical block diagram.

This model will be used further to add the non-linear components to the system, to show a full practical motor system.

3 NEURAL NETWORK STRUCTURE AND TRAINING METHOD

3.1 Neural Network Structure:

Neural Networks are widely used to solve for complex problems involving nonlinearities. Different NN architectures can be used, Feed Forward Neural Networks (FFNN) can be considered as the most common network to use in function generalization, due to its simplicity and fast learning speeds. FFNN's are NN that propagates the data from the input layer to the output layer, passing through one or more hidden layers, in one forward direction, there are no internal looping in FFNN's [Haykin, Simon; 2001]. A three layer FFNN is shown in Figure 3.

The first stage of a FFNN's is the input layer, in which the input data can be given as a single element, a vector or in matrix form. The input layer accepts the input data and propagates them to next layer. The second stages (are) the hidden layer(s); in which most of the data processing is completed before sending the processed data to the next layer. These hidden layers have transfer functions, weight and bias values. Each single data has to be processed through the hidden layers before its ready to be sent to the next level of the network. After the hidden layers finish their data processing the output layer accepts the data elements and sends them to the output as a ready to use data information [Demuth Howard, Beale Mark; 2004]. The final output of the three layers FFNN shown in figure 3 can be written as:

$$a^3 = f^3(W^3 f^2(W^2 f^1(W^1 \mathrm{p} + b^1) + b^2) + b^3) \tag{4}$$

where a is the network output, f is the transfer function used in each layer, W is the weight value, and b is the bias.

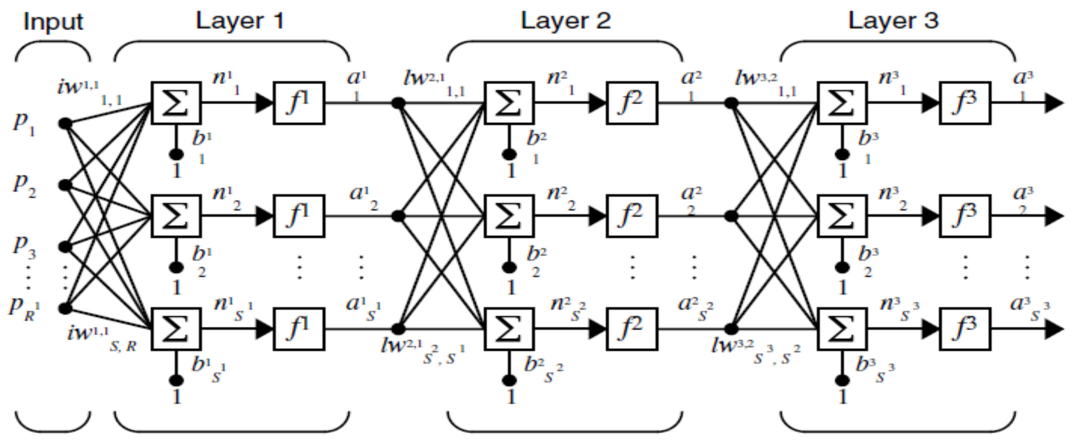

Figure 3. Three layer FFNN.

3.1 Neural Network Training Method:

Each NN has a training method used in order to get a useful data processing from it. Different training methods are used with NN; Back-Propagation (BP) training method is one of the most common training methods used with FFNN. BP is considered as a supervised learning method, in which input/output data sets are used to train the network. When the network receives the data, the input layer neurons accepts the data, and the output layer neurons generate the target data based on the output data values, in between the hidden layer neurons adjusts the weights so that any new input data will have correct output values. The BP method divides the data into three sets, training set, testing set and validating set. The training set is used to train the network and adjust the weights, the testing and validating sets are used to check if the network works correctly by supplying only input data without the corresponding target data.

The training phase of the network runs until a minimal error value is reached, in this phase all the weight values were being adjusted meet the requested error value. The BP method changes the weights and biases and the same direction as the performance function decreases most rapidly [SiriWeereasooriya and M. A. El-Sharkawi; 1991]:

$$E = \frac{1}{2}\sum_i (Ti - Yi(H))^2 \tag{5}$$

Y_i(H) represents the activation of the ith neuron, in the output layerH, $\mathrm{T_i}$is the target value. The weights are updated using the gradient decent technique:

$$\mathrm{w_{ij}^{new(h)}} = \mathrm{w_{ij}^{old(h)}} + \eta \frac{\partial \mathrm{E}}{\partial \mathrm{w_{ij(h)}}} + \upsilon \, \Delta \, \mathrm{w_{ij(h)}} \tag{6}$$

where w is the weight value, η is the learning step, υ is the momentum gain and $w_{ij(h)}$ represents the change of the weight in the previous cycle. This equation is used to calculate the new weights values and to update the weights in the hidden layers; this equation will be used every time the NN runs a new training cycle, after NN reaches the predefined error the training process will stop. The number of hidden layers and neurons in each layer are found using trial and error, since there is no rule to give the best NN structure. The designer can start with two networks structures, one that has a low number of neurons and another that has high number of neurons, test both to get the performance of each one, then decide whether to decrease or increase the number of neurons and layers until the network reaches its maximum performance. The more hidden layers are used in the network, the more efficient network performance is obtained and the network can solve for more complex problems, but it will require more processing and training time and be a more expensive implementation.

As the training method is very important in finding the best network for an application, it is also important to choose the best transfer functions (TF) to produce the outputs of the network. Two TF's were selected to train the network, which were found to be the best for our application: the Tan-Sigmoid (tansig) TF and the linear TF. The Tan-Sigmoid TF accepts any data range between ±∞ and generates an output of the range (-1,1). The other TF used is the linear TF; which takes the output of the Tan-Sigmoid TF and scale it back to normal range. Figure 4 shows both TF's.

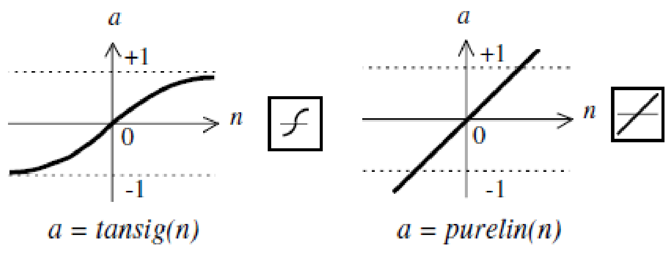

Figure 4.NN Transfer Functions.

The mathematical representation of the Tan-Sigmoid function is given by:

$$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \tag{7}$$

And for the linear transfer function is:

$$a = n \tag{8}$$

Where a represents the TF output and n is the data numbers.

4 RESULTS

4.1 NN Motor Model

A FFNN was used to build a neural network that mimics the function of the servomotor, what makes this NN a model distinguishable step and a new option compared to the other control methods already proposed is that having the NN motor model will enable us to perform efficient off-line simulation, test the model and the controller before implementing the actual system, in this way we will save time and cost for any modifications needed. Besides that having a NN motor model will enable us to use for academic and research purposes. Simulink was used to get the input/output data pairs used to train the NN, Figure 5 shows the complete motor block diagram with all the non-linear parameters added to it, so it performs as a real servomotor.

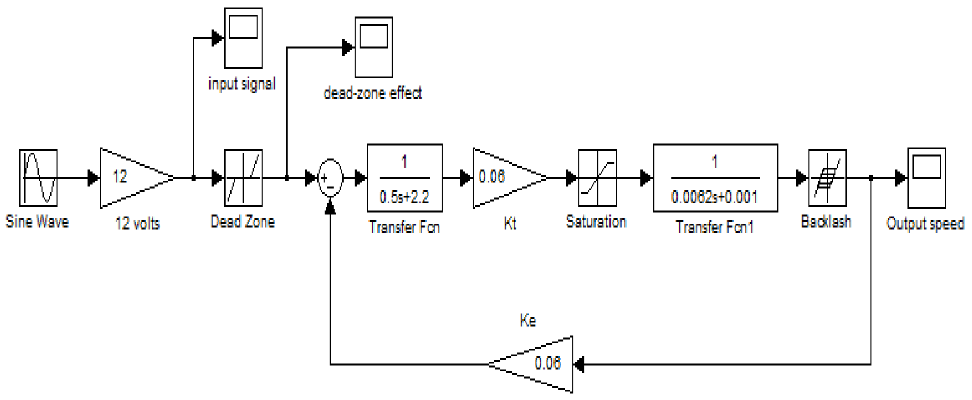

Figure 5. DC Servomotor Block diagram with added nonlinearities.

All the data was obtained using MATLAB Simulink and recorded into MATLAB files until the start of the NN training. The structure of the motor's NN has the same structure layout shown in figure 3 above. 25 neurons were used in this network, the error was defined to be 0.00005 and the maximum number of training trials was set to be 1000. Figure 6 shows the response of the Simulink model and the response of the NN after training. From both plots we can see that the network is able to mimic the function of the motor and has the same response for the same input data.

The network took 23 trials to reach the acceptable error. The performance of the network was very high with most of the data laying at the regression line with R = 0.98969. Figure 7 shows the internal structure of the generated NN, the NN block diagram was generated using the *gensim(net,ts)* MATLAB command. Where *net* represents the trained NN and *ts* is the sampling time. The value of t_s was set to -1 for continue sampling.

4.2 NN Controller Model

In order to build the NN controller model, we needed to have a predefined controller for the system. The predefined model selected was a PID controller; which was tuned to get the best response for the servomotor. The tuned parameters for the PID controller were P = 20, I = 2 and D = 20. Although, the PID controller has a good effect on the system's performance, it is not the best controller to improve the overall motor performance and to eliminate the non-linear effect of the servomotor.

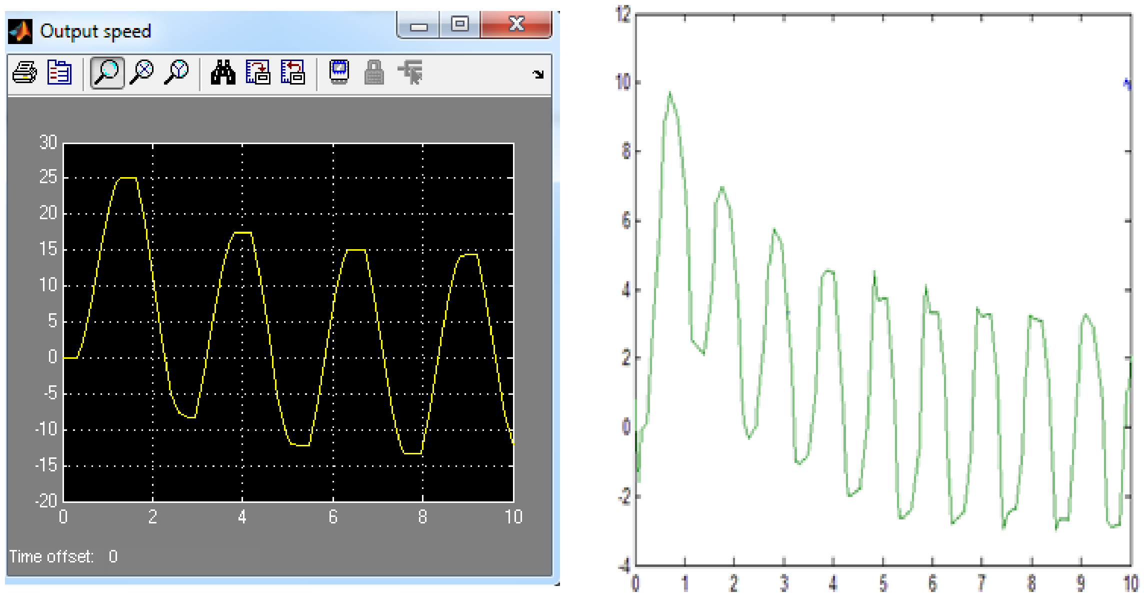

Figure 6. Motor Simulink results and NN model simulation results.

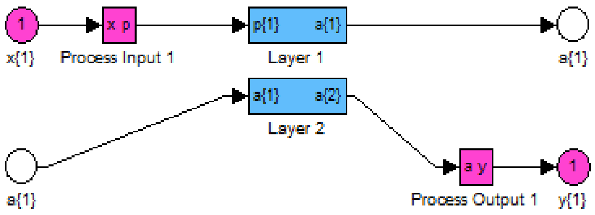

Figure 7. Internal NN block diagram.

To observe the system's reaction with the PID controller connected, we applied a step input to test its response. As can be seen in figure 8, even though the PID controller improved the system's performance, we still can see the high impact of the nonlinearities on the system performance. Figure 8 shows the system's input and output signals.

Also from the simulation results we obtained the input and output PID signals, shown in figure 9, it can be noticed that the two signals are not well-matched with each other and the input signal has a high spike value; which may be dangerous in some applications in continuous running mode, since this spike in the continuous running mode may lead to a system failure.

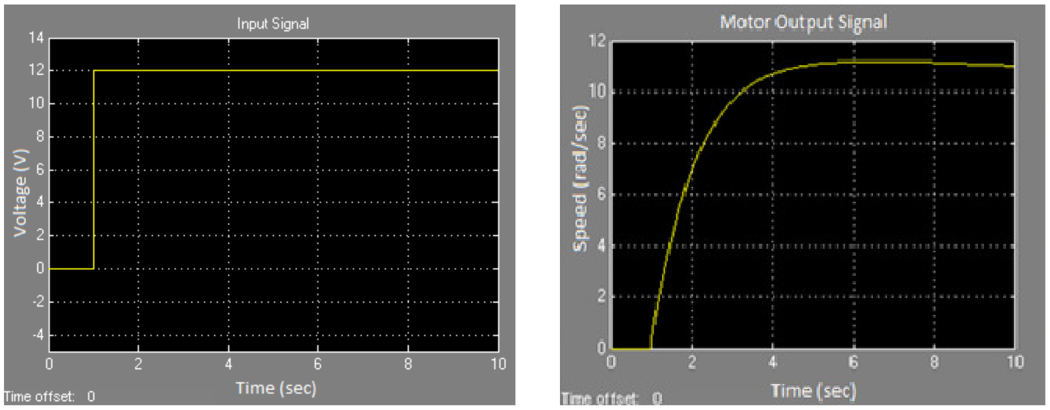

Figure 8. Input and output motor signals.

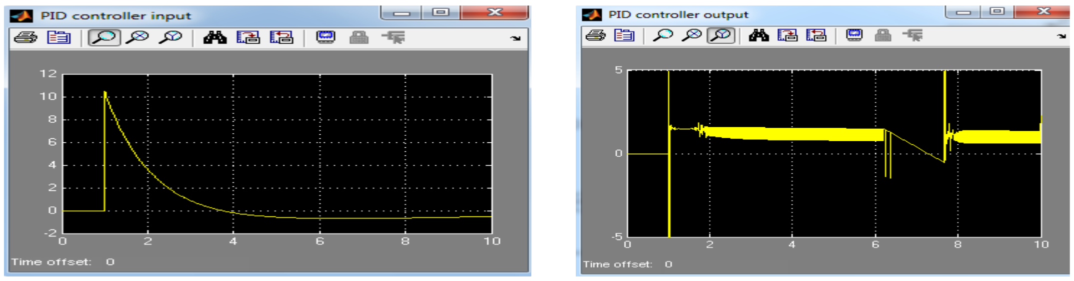

Figure 9. PID controller direct input/output signals.

The NN controller model was built based on the data obtained from the PID controller signals, the structure of the NN controller model has the same structure as the motor model, 25 neurons were used in the hidden layer with *tansig* and *linear* transfer functions. The NN controller was successfully trained using 17 trials to reach the predefined error of 0.00005.

4.3 NN Complete system:

After obtaining the NN motor block and the NN controller block, they were connected to build the complete system. A sinusoidal wave function and step input were used to test the system's performance and to compare the results to the uncontrolled system, and also with the original PID controlled system. Figure 10, shows the connection of the two developed NN used to integrate the complete system.

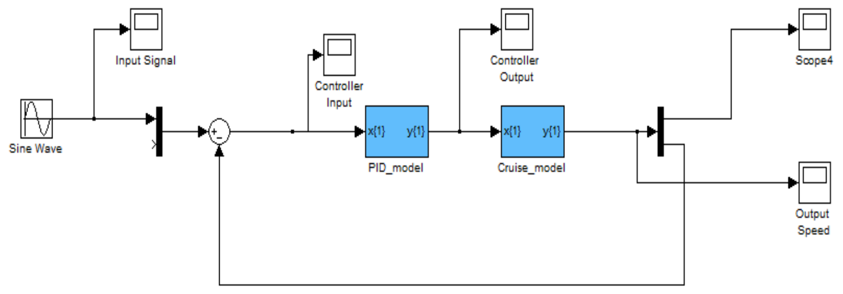

Figure 10. NN complete system.

With negative feedback, the system was tested on a sinusoidal signal, with 2.5 rad/sec and amplitude of 1 volt. Direct input/output controller signals and motor speed signals were plotted using MATLAB Simulink, these results are shown in Figure 11. From this figure we can see that the NN controller input is very smooth with no spikes; which confirms that the NN controller performance is better. Beside this, we can see from the figure that the output speed is compatible with the output of the controller; which confirms that the nonlinear effect found on the system before no longer exist.
From the response of the NN system it is clear that the overall system performance was improved, the system is able to deal with all the nonlinearities found in the system before and that the response of the NN controller better than the PID controller.

5 CONCLUSION

In this paper we have produced two NN network models: 1) A servomotor NN model, 2) APID NN controller model. Both networks were built using FFNN, with 25 neurons in the hidden layer and Tan-Sigmoid and linear transfer functions for both networks. The NN motor model was successful to mimic the function of the real servomotor, with high network performance and very short training time, making this model very useful for different applications. The NN controller model has improved the system's performance, overcame the nonlinearities found on the system and also has a simple neuron structure; which makes it simple and easy to implement the controller and test on a real motor.

In summary, the original aim of this research to develop an efficient controller that was capable to improve the servomotor efficiency was successful. Both implemented networks have high efficiency, low short training time and simple structure. The NN controller can be used in any automated system that requires efficient actuators and control system; which is also very important for time critical applications.

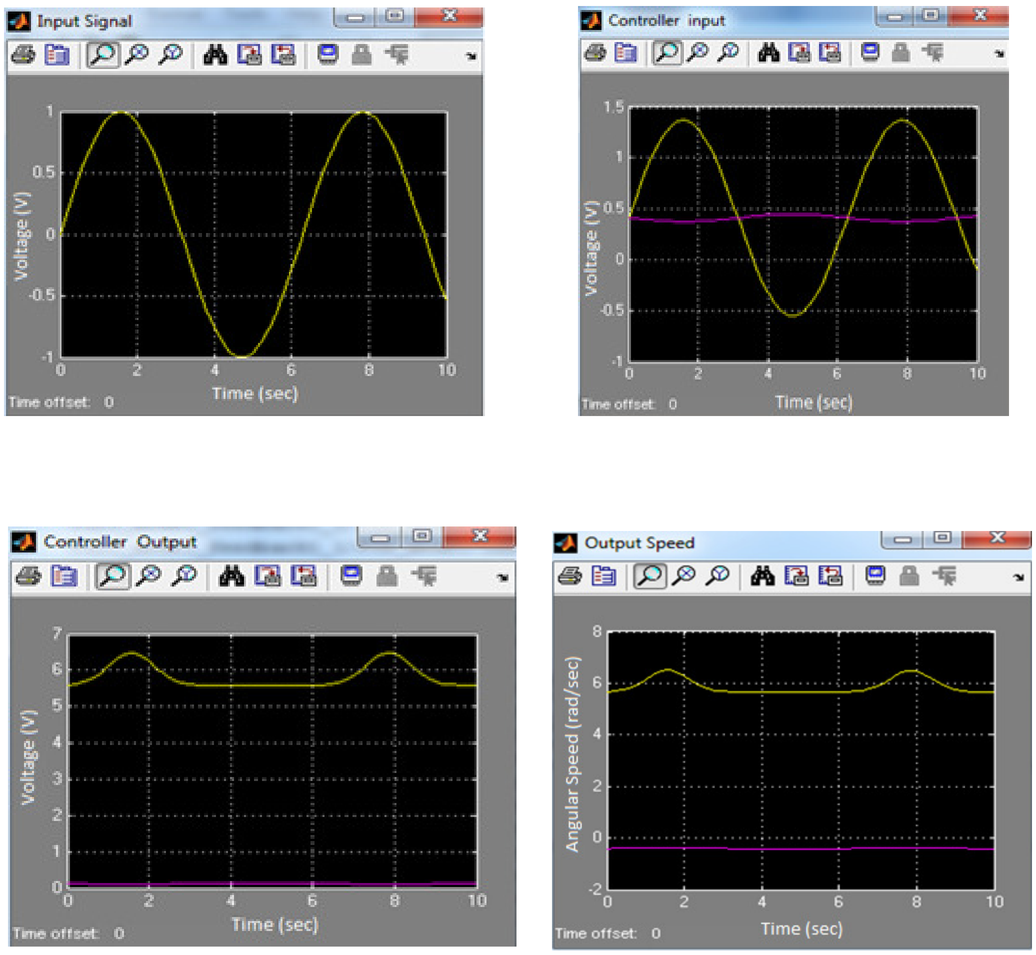

Figure 11. Complete NN system response.

REFERENCES

[1] Kandel E, Schawrtz JH, JesselTM., "Principles of Neural Science". *4th ed.* McGraw-Hill, New York ,2000
[2] Miller, W.T, Sutton, R.S. and Werbos, P.J. Neural Network for Control, MIT Press, Cambridge, MA (1990).
[3] Hagan T. Martin and Howard B. Demuth. "Neural Network Design", PWS Publishing Company, 1996.
[4] Mahanijah M.K., Nasirah M., "Controller Design for Servomotor". IEEE Symposium on Industrial Electronics and Applications. Malaysia 2009.
[5] Jun Oh Jang, "Neural Network Saturation Compensation for DC Motor Systems". IEEE Transaction on Industrial Electronics.Vol 54, June 2007.
[6] AhmadN.J., ZeraiA., Al-MuminM. "A New Continuous Time Adaptive Backlash Inverse Controller and Applications to Output Backlash Compensation". Department of Electrical Engineering Technology Report, College of Technological Studies. Kuwait 2003.
[7] Norman S. Nise. "Control System Engineering".*5th ed*, John Wiley and Sons, 2008
[8] Robert N. Bateson. "Introduction to Control System Technology".Prentice Hall, Inc. Upper Saddle River, New Jersey. 1999
[**9**] Haykin, Simon. "Nonlinear Dynamical Systems: Feedforward Neural Network Perspectives Adaptive and Learning Systems for Signal Processing, Communications and Control", Wiley-Interscience, New York,2001
[10] Demuth Howard, Beale Mark. "MATLAB Neural Network Toolbox Documentation".The MathWorks, 2004
[11]SiriWeereasooriya and M. A. El-Sharkawi."Identification and Control of a DC Motor Using Backpropagation Neural networks",Vol 6.IEEE Transaction on Energy Conversion. 1991.