Speed Control of an Asynchronous Motor Using PID Neural Network

V. Ayhan Maraba[1], A. Emin Kuzucuoglu[2]

[1] Department of Information Technology,
Haydarpasa Vocational High School,
Istanbul, Turkey.
ayhan_maraba@hotmail.com

[2] Department of Electronics and Computer Education,
Faculty of Technical Education, Marmara University,
Istanbul, Turkey.
kuzucuoglu@marmara.edu.tr (Corresponding author)

Abstract: This paper deals with the structure and characteristics of PID Neural Network controller for single input and single output systems. PID Neural Network is a new kind of controller that includes the advantages of artificial neural networks and classic PID controller. Functioning of this controller is based on the update of controller parameters according to the value extracted from system output pursuant to the rules of back propagation algorithm used in artificial neural networks. Parameters obtained from the application of PID Neural Network training algorithm on the speed model of the asynchronous motor exhibiting second order linear behavior was used in the real time speed control of the motor. The real time control results show that reference speed successfully maintained under various load conditions.

Keywords: PID, neural network, PIDNN, control.

1. Introduction

The basic purpose of control systems is to keep system behaviour at desired values. A controller located in a closed loop control system generates the control signal required to keep the output of the system at the desired value. In classical control methods, the selection of the controller to be used in the control of systems and for the detection of The parameters that are determined in this way cannot always provide the desired system stability due to various factors, such as modelling mistakes, changes in the parameters of the controlled system, and disruptive effects. Due to all of these problems in classical control methods, practitioners began to use artificial neural networks (ANNs) in the control field because they have the ability to learn and generalize, and the derivation of a mathematical equation is not required [1,2]. Today, most of the systems used in industry exhibit non-linear, time-delay behaviour. These systems have excessive overshoot and high settling times, and they are not stable. It is very difficult and demanding to design a controller for such systems using classical methods. Controllers can be designed by using various methods if mathematical models or transfer functions are available that represent the behaviour of the system very well. However, it is rather difficult to develop mathematical models of such systems in practice [3].

Classical Proportional Integral and Derivative (PID) controllers are immensely preferred in many areas of industrial control, especially in the chemical industry due to their simple structure and high durability. Although PID controllers are used for controlling many systems, it is difficult to find optimum parameters which are proportional, integral and derivative (K_P, K_I, K_D) for the control of time-delay and non-linear systems [3].

Fuzzy logic has been used to improve the performance of PID-controlled, non-linear systems. The fuzzy logic controller modified the parameters of the PID controller to get better system response [4]. Complex devices, such as hyper-redundant robots, can be controlled with a PID control algorithm. The determination of the combinations of proportional, integral, and derivative parameters, as well as the optimum values of these parameters, is a time-consuming operation [5].

PID-Neural Network (PIDNN) controllers perform adaptive control. Uncertainties in systemic and environmental factors ensure that the controller exhibits acceptable behavior by means of adaptive control. However, in practice, there are many problems that must be solved, and this is considered to be one of the main disadvantages of such controllers. The main problems are the slow learning rate, the slow approximation to the reference value, and

the uncertainties in the parameters of the controller [3].

A PIDNN controller includes the advantages of the PID algorithm and the neural network. Such a controller is not a hybrid structure that consists of artificial neural networks and a PID controller [6]. The PID algorithm exists in the neurons located in the neural structure as an activation function.

There are P-proportional, I-Integral, D-derivative neurons in the PIDNN structure, and the connection weights that arise among these neurons are updated by using a back-propagation training algorithm according to the error value propagated through the system [3].

2. Structure of the PIDNN Controller

2.1. General Structure of PIDNN

The development of a PIDNN control algorithm is illustrated in the flow chart in Figure 1. The structure of the controller consists of four different modules. In the initial values module, various values are entered to the program, including the numbers of inputs and outputs of the system to be controlled, initial values of the connection weights to be used, learning rate, and the number of trainings. The most important task at this stage is to determine the initial values of the connection weights.

These weights can be determined according to classical PID rules. Initial connection weights can be chosen as values for which the system is stable, there are no oscillations around the reference value, and the sum of square errors is above the value determined as the tolerance value. Thus, the controller algorithm will operate more stably, conduct the optimization process more rapidly, and stay far away from insoluble areas compared to randomly selected initial weights.

The forward-direction calculation module is the stage in which the reference value and system output conduct PID functions in the hidden layer. The functions are assigned values as network input and when the control signal given to the system as input is calculated. In the middle layer, P, I, and D neurons are located. In the output layer, the located neurons give control signals. The outputs of these neurons are limited based on the status of the system.

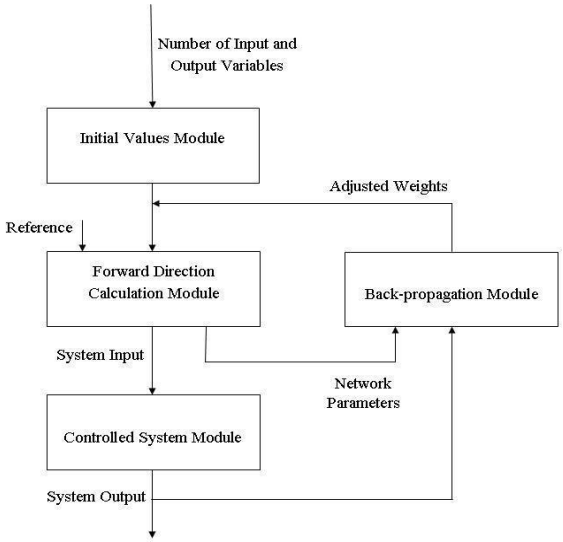

Figure 1. Flow chart of a PIDNN controller [8]

In the backward propagation module, attempts are made to minimize the cost function determined as the average of square errors. The algorithm used for this purpose is the error back-propagation method, which is the most commonly used in artificial neural networks and gradient descent algorithm. Connection weights of the PIDNN controller are updated according to the gradient descent algorithm during on-line training [7]. The process of updating with this method continues until the value specified as the cost function falls below the quantity permitted by the user. In the back-propagation module, the number of changes or updates that will be applied on the parameters of the controller is calculated.

If a system module is controlled, the difference equation of the system to be used during simulation studies is the module in which models obtained for single-input-single-output, multi-variable, linear or non-linear systems are used. Training and updating process are conducted as if the data had been received from a real system [8].

The PIDNN controller is a new type of controller in which artificial neural networks are used with a PID algorithm. It is generally known that, in classical PID controllers, a control signed is determined by multiplying the error value that occurs in the system by coefficients, such as K_P, K_I, and K_D. However, in the controller that we analyzed, the PID structure was determined based on the neurons of the network.

The controller consists of three layers, i.e., the input layer, the middle or hidden layer, and the

output layer. In the input layer, there are neurons that give the reference value and system output values as input into the system. In the middle or hidden layer, there are neurons that perform the PID functions. The P neuron performs the proportional function, the I neuron performs the integral function, and the D neuron performs the derivative function. In the output layer, there is a neuron in which the control signal that the controller gives to the system is calculated. The controller consists of two parts, i.e., the forward direction calculation and the training stage. In the former, the control signal is calculated, and in the latter, the connection weights are adjusted by using the error value propagated through the system and back propagation algorithm. The PIDNN structure for single-input-single-output systems is shown in Figure 2.

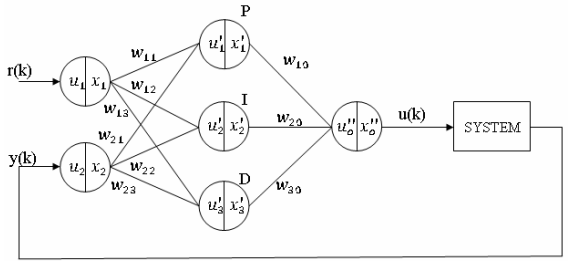

Figure 2. Structure of PIDNN controller [3]

2.2. Feed Forward Calculation

2.2.1. Input Layer

There are two neurons in the input layer, and the input-output functions of these neurons are demonstrated in Equation (1).

$$x_i(k)=\begin{cases}1 & u_i(k)>1\\ u_i(k) & -1\le u_i(k)\le 1\\ -1 & u_i(k)<-1\end{cases} \quad (1)$$

where $i=1,2$ and k = number of samples

2.2.2. Hidden Layer

This layer contains the neurons that perform the basic functions of the controller. The values calculated by multiplying the output values of the neurons in the input layer and the connection weights between the input and hidden layers come to the neurons in this layer, and they are processed in the neurons that conduct the PID algorithm. Inputs formed in the neurons of the hidden layer are calculated as shown in Equation (2).

$$u'_j(k)=\sum_{i=1}^{2} w_{ij}.x_i(k) \quad (2)$$

where $j=1,2,3$ w_{ij} is the connection weights formed between the input and the output layer.

After the values formed in the inputs of the hidden layer neurons are calculated, the functions of neurons in the forward direction calculation stage can be demonstrated, as shown in Equations (3-5):

P-Neuron;

Expression of the output value of P Neuron located in the hidden layer is demonstrated in Equation (3).

$$x'_1(k)=\begin{cases}1 & u'_1(k)>1\\ u'_1(k) & -1\le u'_1(k)\le 1\\ -1 & u'_1(k)<-1\end{cases} \quad (3)$$

I-Neuron;

Expression of the output value of I neuron located in the hidden layer is demonstrated in Equation (4).

$$x'_2(k)=\begin{cases}1 & x'_2(k)>1\\ x'_2(k-1)+u'_2(k) & -1\le x'_2(k)\le 1\\ -1 & x'_2(k)<-1\end{cases} \quad (4)$$

Accordingly, the neuron output is found by adding the value coming to the input of the neuron and the previous value of output.

D-Neuron;

Expression of the output value of D neuron located in the hidden layer is demonstrated in Equation (5).

$$x'_3(k)=\begin{cases}1 & x'_3(k)>1\\ u'_3(k)-u'_3(k-1) & -1\le x'_3(k)\le 1\\ -1 & x'_3(k)<-1\end{cases} \quad (5)$$

Accordingly, the neuron output is found by adding the value coming to the input of the neuron and the previous value of output.

2.2.3. Output Layer

Values obtained at the outputs of hidden layer neurons come to the input of the neuron in the output layer by being multiplied by the connection weights between the hidden and the output layers.

$$u''_0(k) = \sum_{j=1}^{3} w_{j0}.x'_j(k) \quad (6)$$

The net total calculated with Equation (6) passes through the output layer and forms the control signal. The input-output function of the output layer is demonstrated in Equation (7). Accordingly, the output neuron is equally propagated to the output, on the condition that the value at its input is between 1 and -1.

$$x''_0(k) = \begin{cases} 1 & u''_0(k) > 1 \\ u''_0(k) & -1 \le u''_0(k) \le 1 \\ -1 & u''_0(k) < -1 \end{cases} \quad (7)$$

When we look at the network structure of a PIDNN controller, we can see that the multi-layer ANN acts as a classical PID controller by using appropriate connection weights. If we choose the connection weights between the input layer and the hidden layer in the PIDNN structure as in Equation (8),

$$W_{1j} = +1, \; W_{2j} = -1, \; j = 1,2,3 \quad (8)$$

the connection weights between the output layer and the hidden layer are the coefficients found in the classical PID structure. These network parameters are demonstrated in Equation (9).

$$W_{10} = K_P, \; W_{20} = K_I, \; W_{30} = K_D \quad (9)$$

According to these values, values coming to the inputs of neurons in the hidden layer are like the values given in Equation (10).

$$\begin{aligned} U'_1 &= W_{11}X_1 + W_{21}X2 = r - y = e, \\ U'_2 &= W_{12}X_1 + W_{22}X_2 = r - y = e, \\ U'_3 &= W_{13}X_1 + W_{23}X_2 = r - y = e, \end{aligned} \quad (10)$$

According to the formulation in Equation (10), the error value that is the difference between system reference and system output is applied to the inputs of the neurons located in the hidden layer due to the values of connection weights. That is to say, error values come to the inputs of the neurons located in the hidden layer. The value found after the neurons pass through the activation functions, which are P, I, and D neurons, are given in Equation (11).

$$\begin{aligned} X'_1 &= U'_1 = e, \\ X'_2 &= \int_0^t U'_2 dt = \int_0^t e dt, \\ X'_3 &= \frac{dU'_3}{dt} = \frac{de}{dt}, \end{aligned} \quad (11)$$

Accordingly, network output is found by multiplying error values by the connection weights between the hidden layer and the output layer. This expression is given by Equation (12).

$$X''_0 = U''_0 = \sum_{j=1}^{3} W_{j0} X'_j = W_{10}X'_1 + W_{20}X'_2 + W_{30}X'_3 \quad (12)$$

In conclusion, choosing the connection weights between the input layer and the hidden layer according to the values demonstrated in Equation (8) causes the PIDNN network structure to become a classical PID structure. When the activation functions of the neurons located in the latent layer are considered according to Equation (11), the output of the controller network is as shown in Equation (13).

$$X''_0 = K_P e + K_I \int_0^t e dt + K_D \frac{de}{dt} \quad (13)$$

As a result, the designed network structure exhibits the behaviour of a classical PID, depending on the values of the connection weights.

This feature of a PIDNN controller makes it easy to use in practice. ANNs are not used for system control in practice due to the uncertainties in the initial weights and the failure to ensure stable operation of the system. We can find appropriate initial weights by operating the PIDNN like a classical PID control when it is first operated. After these weights are chosen, system response and control performance can be improved by operating the back-propagation training algorithm [9].

3. General Structure of the Controlled System

The asynchronous motor system to be controlled is composed of three main parts, as shown Figure 3a. The mechanical part of the system is designed to act as fixed brake by charging the AC motor operating at fixed speed with a fixed load. In this part, there is a 200 W DC motor, four spot lamps with 12 V, 50 W

power, and the relay card. On the left side, there is a 3-phase, 250 W AC motor, a Micromaster-420 driver, and a tacho generator that generates a voltage according to the speed of the AC motor. A National Instruments PCI-6024E data acquisition card was used as the control unit.

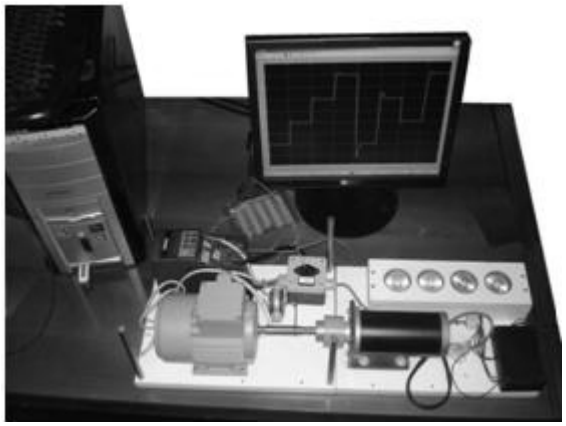

Figure 3a. Experimental setup of the asynchronous motor system

As shown in Figure 3b, the controller algorithm that works on MATLAB Simulink in a PC environment communicates with the system through the PCI-6024E data collection card. The spot lamps connected to the DC motor, which charges the asynchronous motor, are loaded or unloaded analogously. The tacho generator that conveys instantaneous information on revolution frequency in the block diagram is connected directly to the data collection card, and the control signal generated by software written in MATLAB is sent to the driver through the input-output card.

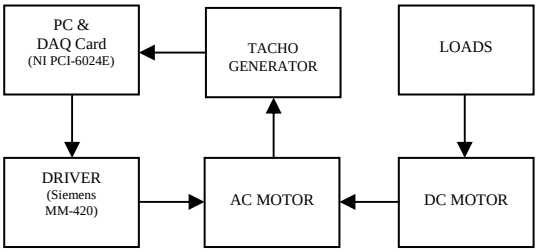

Figure 3b. Block diagram of the system

4. Speed Control Studies of AC Motor

4.1. System Model

Generally, system modelling means measuring the inputs and outputs and finding the physical system variables with a model approach. The input-output dataset that forms a system model contains values sampled in sufficient numbers [10]. Formation of the input-output dataset forming was completed in two stages. In the first stage, the control signal was varied between 0-10 V in step function form. In the second stage, the control signal was varied randomly between 0-10 V, and the speed of the AC motor was recorded for 500 seconds in both stages. The sampling time used was 0.1 second. The input-output dataset was formed by using these data. This dataset was used in MATLAB.

The System Identification Toolbox was used to get the speed model of the AC motor. The Auto Regressive eXpgenous (ARX) model was chosen and is shown in Equation (14).

$$A(q)\, y(t) = B(q)\, u(t) + e(t) \tag{14}$$

where y(t) is the output, u(t) is the input, A(q) corresponds to poles that are common to the dynamic model and the noise model, and B(q) represents the contributions of inputs to predicting all output values [11].

The order of the ARX model was chosen as two. After applying the dataset to the program,

$A(q) = 1\text{-}1.283q^{-1} + 0.3969q^{-2}$

$B(q) = 0.01103q^{-1} + 0.03599q^{-2}$

A and B polynomials were determined. The difference equation of the model is shown in Equation (15).

$$y(k+1) = 0.01103 * u(k) + 0.03599 * u(k-1) + 1.283 * y(k) - 0.3969 * y(k-1) \tag{15}$$

The speed model of the AC motor is shown as a transfer function in discrete time in Equation (16).

$$\frac{y(z)}{u(z)} = \frac{0.01103z + 0.03599}{z^2 - 1.283z + 0.3969} \tag{16}$$

In Figure 4, the output of the ARX model and the speed of the AC motor, taken from the tacho generator, were compared. Clearly the model is good. It catches relevant features of motor during the rising time and the settling time. In Figure 5, the outputs of the real-time system and the model are compared at various speeds. Some minor differences can be accepted in transitions of steps. Figure 6 gives the error between the outputs of the model and the real system.

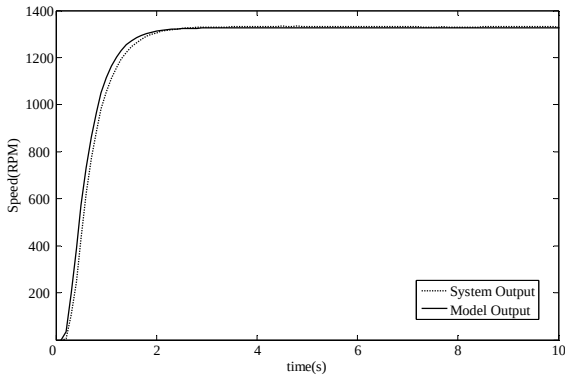

Figure 4. Open loop comparison of system and model outputs at 1300 RPM

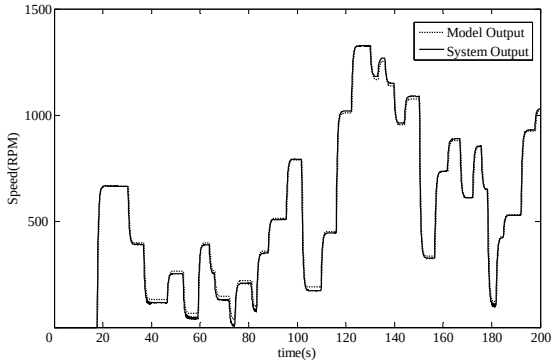

Figure 5. Various speed results of system and model outputs

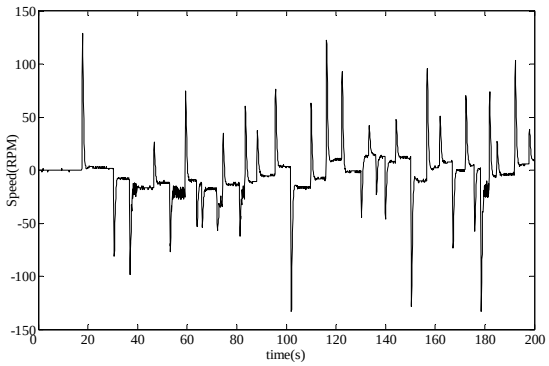

Figure 6. Errors between system and model outputs

The open loop response of the system to a step input in discrete time is shown in Figure 7. The settling time of the system for the step input was approximately 1.7 seconds, and the rise time was approximately 1 s. After the difference equation for the asynchronous motor in discrete time was obtained, the process of improving the response of the system using this model was performed at the training stage of the controller. The importance of ANNs at this point was the use of the back propagation algorithm for the purpose of decreasing the mean square of errors.

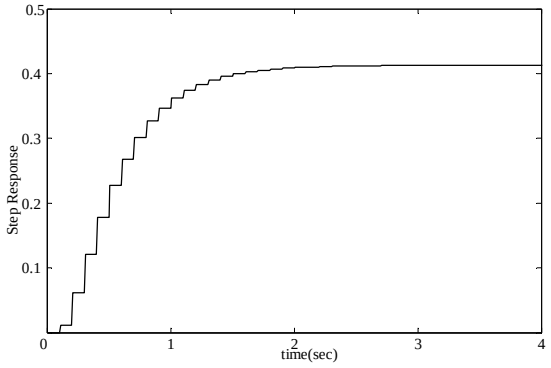

Figure 7. Open loop response of the asynchronous motor in discrete time

The structure of the controller is not complex, and classical PID functions were conducted in the network. Some parameters must be determined before starting the training process, i.e., learning rate, number of trainings, and the initial value of the weights. Optimization of the process is closely related to choosing the initial values of such parameters that are appropriate for the system.

A 10 V control signal was applied to the model to get the nominal speed of the AC motor, which was 1350 RPM. The rise time of the system was around 1s, and the settling time of the system was approximately 2 s as shown in Figure 8.

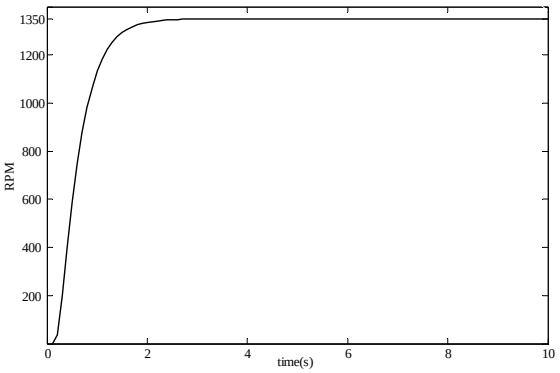

Figure 8. 10 V step response of the difference equation

Controller training was started in consideration of various criteria, such as the learning rate, number of trainings, and initial weights. The system response for the related reference or training set is shown in Figure 9.

Rise time of the system response obtained as a result of training is short, which means the system output has rapidly approached the related reference value. Overshoot amount is low, and steady state error is close to zero.

Trained system response and open loop system response are compared in Figure 10.

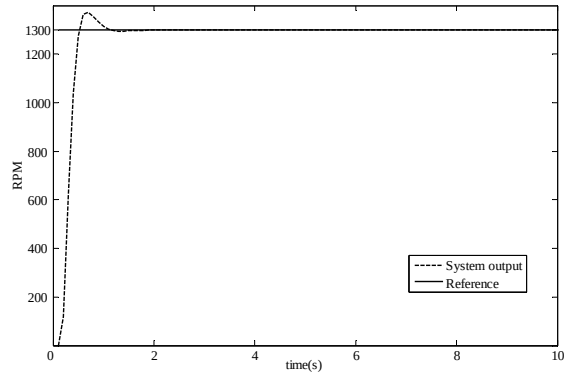

Figure 9. System response after controller training

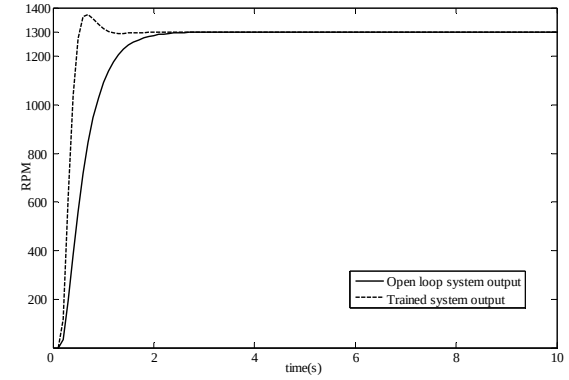

Figure 10. System responses according to open loop and optimized parameters

The settling time was 2 s in the open loop system, which decreased to around 1 s using the controller.

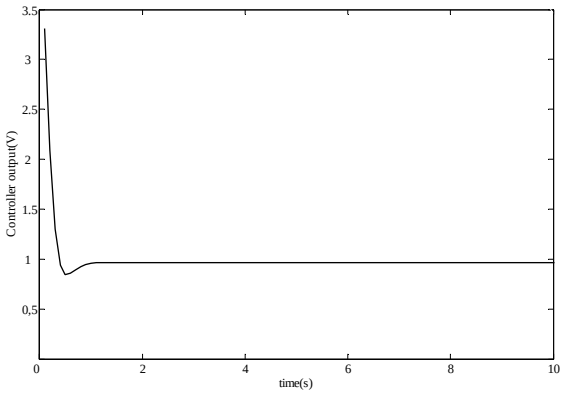

Figure 11. Optimized controller network output

Figure 11 shows the values of the controller signal obtained from the output of the controller In the real-time study, the neurons at the PIDNN output and the controller network output were limited between 0 and 1 or -1 and 1 as was mentioned in the output layer section. For the output layer, however, no restrictive criteria are put at the network output since the controller is operated on the system model. The output of the controller network without any restrictions is shown in Figure 11. The output of the controller network in the real-time study should be limited to 0-10 V; otherwise the related system could be given a control signal above 10 V.

The most important part of PIDNN controller algorithm is the minimization of the cost function. As shown in Figure 12, cost function specified as the average of square errors decreased after each iteration of training. The error value was high in the first training step, but the parameters applied after each update caused the error in the system response to decrease compared to the previous error.

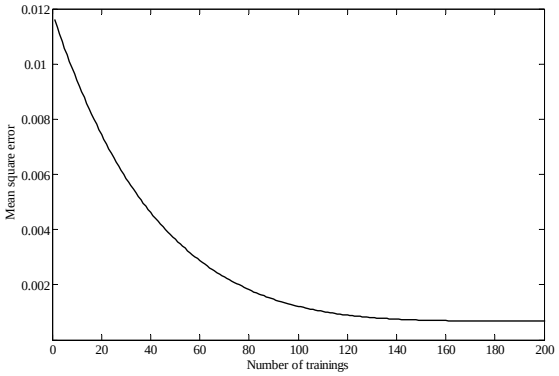

Figure 12. Cost Function

Whether the training is quick or slow does not mean that the appropriate parameters will be found more accurately. Learning rate is a factor that determines the training rate. As was stated earlier, there is no predefined rule for the initial value of the learning rate; it can change depending on the implementation. If the training is continued after the 200th iteration, the system value becomes unstable and oscillates. Therefore, the maximum number of trainings can change depending on the implementation. Termination of the training is decided based on when the criterion determined as the cost function falls below a value determined by the user, or, at the end of each iteration, the user can examine the system response and decide to stop the training. Connection weights obtained at the end of the training were used in the real-time operation of the system.

4.2. System Response to Step Inputs

Connection weights obtained at this stage of the study were used in the real-time operation of the system. One of the real-system responses

and one of the model responses were chosen, as shown in Figure 13. The real-system output and the model output are shown for reference values of 750 RPM in Figure 13a. For this reference value, the control signal is shown in Figure 13b. As is shown at control signal output, when the system response approaches the related value in the reference speed graphic, the control signal ceased the maximum drive and continued to generate the signal required for the related reference. There is overshoot between the model and the output of the system. These differences arise from modelling errors.

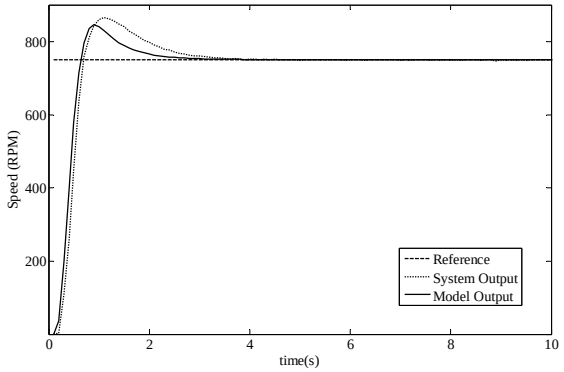

Figure 13a. Real system and model outputs for 750 RPM

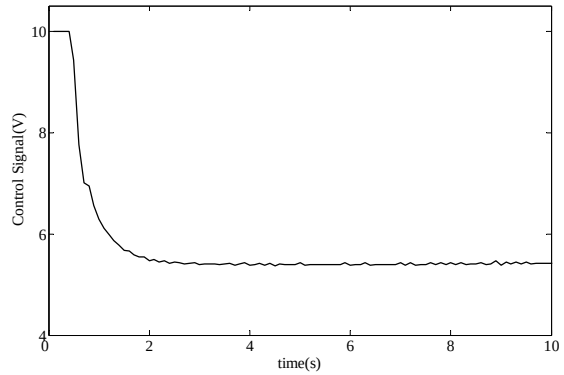

Figure13b. Control signal

The response to the stepped reference input of the system is shown in Figure 14a. If there is a large difference in the reference transition, some overshoot will occur. For example, in the 40th second, the reference dropped approximately 1000 RPM. The error is high for a short time. In this transition, the control signal was 0 V. The control signal was begun again producing after the error was reduced. The control signal values are shown in Figure 14b.

4.3. System Response under Load

A DC motor was connected in order to apply load to the shaft of the asynchronous motor. The DC motor provided electricity for four 50 W lamps, each of which acted like a generator. The lamps were switched on one by one and, the control signals and the speed of the AC motor were observed. Two resulting load conditions are shown in Figure 15a. A rapid decrease of 20 RPM was observed on the shaft of the AC motor operating at 1300 RPM when two 50 W lamps were turned on.

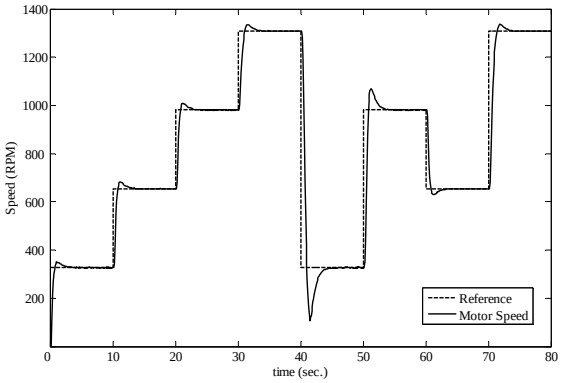

Figure 14a. The system response to a stepped reference input

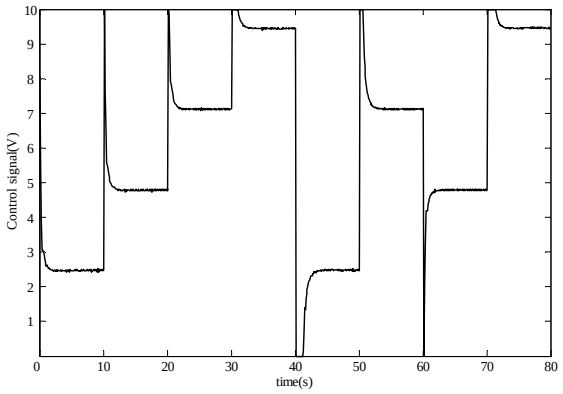

Figure 14b. Control signal

This RPM decrease was recovered quickly by simultaneously increasing the control signal and holding the speed of the motor at the reference speed. When the motor was unloaded, an increase of the speed at the same rate was observed, but the controller turned back to its reference speed value.

In Figure 15a, four 50 W lamps were turned on simultaneously, and a 200 W load was generated in 8th seconds. When the motor was loaded, the revolution rate decreased by approximately 30 RPM, and the controller offset the load by increasing the control signal. When the motor was unloaded in 14th seconds,

the revolution rate increased at the same rate, and the controller decreased the control signal in order to get the related reference RPM at the output of the system (Figure 15b).

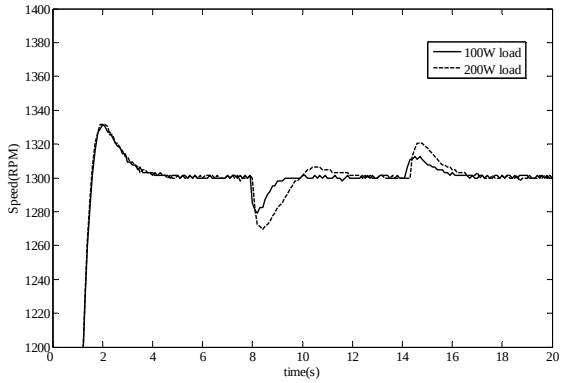

Figure 15a. System response under 100 W and 200 W loads

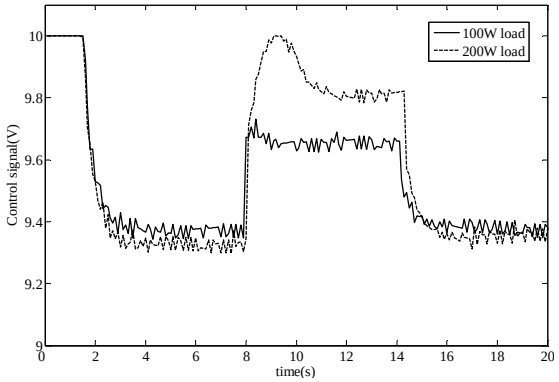

Figure 15b. Control signal under 100 W and 200 W load

5. Results and Discussion

The cost function was decreased by the parameters that were found until it was less than the error tolerance value, thereby completing the optimisation process. With these coefficients, the rise time of the system in closed loop operation was decreased from 2 s to around 1 s. Data obtained from one of the real-time studies at the reference speed of 750 RPM and the data obtained from the simulation studies conducted on the model similar results according to the control criteria, such as rise time, settling time, and overshoot amounts. The response of the AC motor to staircase driving was quick and efficient. When load was applied to the system, the controller increased its drive effect in response to the load.

Conventional PID controller results and PIDNN controller results are shown in Figure 16. The overshoot amount for the PID controller was approximately 1.7% greater than the overshoot amount for the PIDNN controller. The response of the PID controller under load was approximately the same as the response of the PIDNN controller.

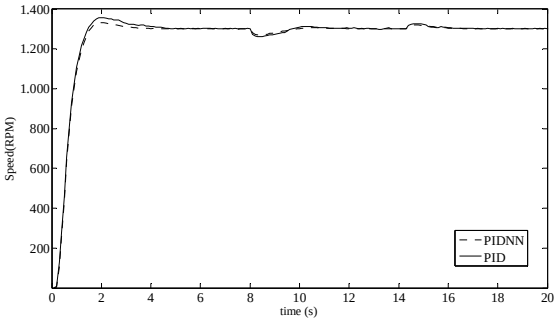

Figure 16. PID and PIDNN responses to the system under 100 W load

6. Conclusion

In this study, initial values of controller parameters, such as the number of trainings, learning rate, and initial weights in the PIDNN structure used were given, and the controller parameters to be used in a real-time application were determined quickly as a result of the operation. In this way, controller parameters of systems with very different structures can be determined using a PIDNN training algorithm, and real-time control processes can be conducted efficiently.

REFERENCES

1. DANDIL B., **Plant Control By Aid Of Artificial Neural Networks**, Master Thesis, Firat University Institute of Applied Sciences, 1997, (in Turkish).
2. PATIC, P. C., R. ZEMOURI, L. DUTA, **Recurrent Neural Networks in Linear Systems Controlling**, Studies in Informatics and Control, vol. 19, no. 2, 2010, pp. 153-158.
3. SHU, H., Y. PI, **PID Neural Networks for Time-delay Systems**, Computers and Chemical Engineering, Volume 24, Issues 2-7, 15 July 2000, pp. 859-862.
4. ABDEL-HADY, F., S. ABUELENIN, **Design and Simulation of a Fuzzy-Supervised PID Controller for a Magnetic Levitation System**, Studies in Informatics and Control, vol. 17, no. 3, 2008, pp. 315-328.

5. IVĂNESCU, M., M. C. FLORESCU, N. POPESCU, D. POPESCU, **The Control of The Hyper-redundant Manipulators by Frequency Criteria**, Studies in Informatics and Control, vol. 18, no. 3, 2009, pp. 279-288.

6. SHU, H., X. GUO, H. SHU, **PID Neural Networks in Multivariable Control Systems**, International Symposium on Intelligent Control, Vancouver, Canada, 2002.

7. SHU, H.,X. GUO, **Decoupling Control of Multivariable Time-Varying Systems Based on PID Neural Network**, 5th Asian Control Conference, 2004, Melbourne, Australia.

8. SHU, H., H. SHU, **Simulation of PID Neural Network Control System with Virtual Instrument**, 7th International Conference on System Simulation and Scientific Computing, Beijing, China, 2008.

9. SHU, H., Y. PI, Decoupled Temperature Control System Based on PID Neural Network, ACSE Conference, Cairo, Egypt, 2005.

10. RONCO E., P. J. GAWTHROP, **Neural Networks for Modelling and Control**, Centre for System and Control Department of Mechanical Engineering University of Glasgow, Technical Report csc97008, 1997.

11. LENNART L., **System Identification: Theory for the user,** 2nd edition, Prentice Hall PTR, 1999.