# THE DESIGN AND TESTING OF A NEURAL CONTROLLER BASED ON ARTIFICIAL NEURAL NETWORK THEORY USING LABVIEW FACILITIES

Hamid ALSHAREEFI[1], Ciprian LUPU[2], Laith ISMAIL[3], Lich DUC LUU[4]

*This paper presents the design and software solution for the neural controller based on the artificial intelligence theories, the neural networks. Real-time implementation and experimental study performed on the position control system with demonstrating the calculations of feedforward and backpropagation algorithms for the neural controller as well as the front panel design of the main control program using the graphic image software LabVIEW.*

**Keywords**: neural network, feedforward, backpropagation, real-time, position control, LabVIEW

## 1. Introduction

The artificial neural network is used to inspire the biological neurons in the brain. Recently it's widely used in different industrial fields due to its portability to process and solve different high nonlinear data and fuzzy information. It has achieved many successes in image processing, robotics, and other industrial applications. ANN has been spread and developed after the development revolution on computers in high-speed processing and large capacity of memory. The backpropagation feature has been utilized to describe the ANN in the field of automatic control systems. Briefly, in ANN, data input signals are received through the nodes of the input layer and sent to the nodes of the hidden layer through neurons and stimulated by random weights that may be positive or negative. These neurons are collected in the hidden layer through activation functions and then sent to the nodes within the output layer and collecting through activation functions also, which may be linear or nonlinear, preferably non-linear because they have the property of derivation. Depending on the specified degree

[1] Ph.D. Student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: hamedgep@yahoo.com
[2] Professor, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: ciprian.lupu@acse.pub.ro
[3] Ph.D. Student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: laith_ismail@turath.edu.iq
[4] Ph.D. Student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: lanlich@gmail.com

of complexity as well as on the nature of the system, neural networks may contain several hidden layers or one hidden layer, or they may not contain hidden layers [1].

## 2. Artificial neural network architecture

There are many different structures and forms of neural networks. Recurrent neural networks represent the most widely used in the system control field, which are briefly including two main parts:

### 2.1. Feedforward

The part of the interconnections, where the neural network is constructed and the mathematical model of the network is defined. In this part the data is transmitted in one direction through the following layers as shown in Fig. 1.

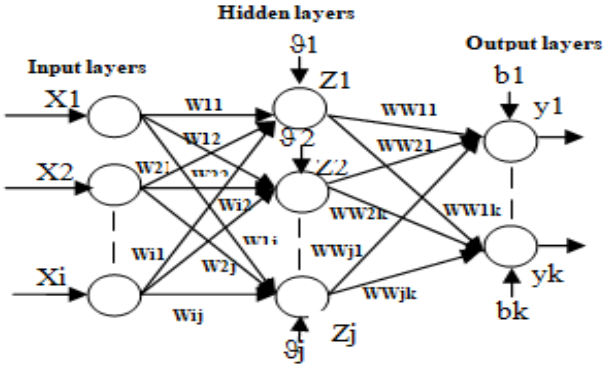



Fig. 1. Artificial neural network General architecture

a) Input layer, which contains one or a group of nodes that represent the number of network inputs that receive data and distribute it through neurons to the nodes of the hidden layer.

b) Hidden layer, resides in-between input and output layers and this is the primary reason why they imply that they are not visible to the external systems and are “private” to the neural network. There could refer to the word “hidden” which could be zero or more hidden layers in a neural network.

c) Output layer, responsible for producing the final output of the ANN model. There must always be one output layer in a neural network and contains one node or more depending on the number of the external outputs of the networks [2].

The data in feedforward transferred through the nodes, and each neuron collects all the input values multiplied by special weights and processes them with an activation function eq. (1).

$$Z(j) = f(x,w) \sum_{i=1,j=1}^{I,J} X(i) * w(i,j) \qquad (1)$$
$$y(k) = f(Z,ww) \sum_{i=1,j=1}^{I,J} Z(j) * ww(i,j)$$

There are many forms of the activation function (f (x), f (Z)), where could be sigmoid, linear, hyperbolic tangent sigmoid, or some other forms; in this paper, the sigmoid function eq. (2), used as an activation function

$$f(x) = \frac{1}{1 + exp\ (-x)} \qquad (2)$$

### 2.2. Backpropagation

Backpropagation is a short form for "backward propagation of errors which is considered the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration) based on the gradient descent of the loss function with respect to all the weights in the network [3].

$$E = y(desired) - y(actual) \dots (lost\ function)$$
$$Wnew = Wold + \Delta W \qquad (3)$$
$$\Delta w = \ learning\ rate * \frac{\partial E}{\partial w}$$

By using the chain rule then will be possible to find the optimal weights which optimize the error or the lost function.

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial out(y)} * \frac{\partial out(y)}{\partial y} * \frac{\partial y}{\partial w} \qquad (4)$$

## 3. Practical implementation

The real applied system on which the neural control algorithm was implemented and tested in this paper is a laboratory position control device, it consists of a tube contains two fans at its base and includes a free ball inside, as shown in Fig. 2. The goal is to control the position of the free ball inside the tube by adjusting the velocity of one fan, for example, fan (A) assuming a constant velocity at some level for the other fan (B) with the possibility of changing the velocity of the fan (B) in a certain ratio to reduce the load on the fan (A), as well as the possibility of exchanging the role between the two fans where the fan (B) is

controlled with the assumption of a constant velocity at a certain ratio of the fan (A)**.** For the data acquisition interface card, NI USB 6008 is used as shown in Fig. 2.

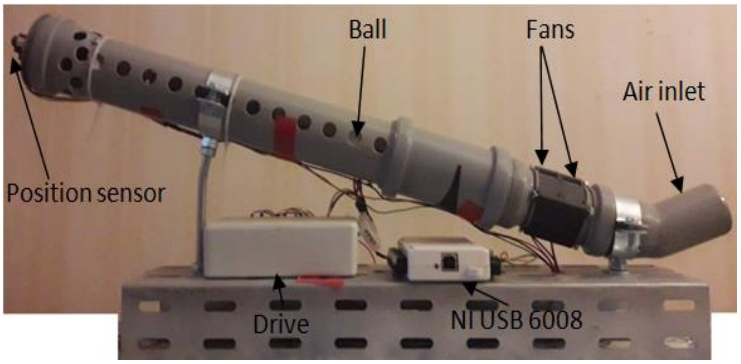



Fig. 2. Position control system

One of the practical solutions used in such kind of processes is the structure of multiple models, where different models of the system are defined by proposing specific domains for the operation level of fans, starting from 0% to 100%, and the modeling of the system is made for each of these operational domains, taking into the account the assumption of common zones between each of the two adjacent domains to avoid The big difference between the operational areas, which in turn leads to a clear difference in controllers designed for adjacent domains [4].

For high nonlinear systems, or systems that operate in many different operating conditions may it difficult to use the multiple models strategy due to the increased complexity in the main control program. Practically using a classical PID controller is not recommended for such kinds of systems if some performances are carefully required.

The presented solution is the neural controller, where the controller is designed based on the theory of neural networks. The neural controller can tune and train itself depending on the optimization of a cost function which represents the summation of squared error between the required reference and system output through the learning algorithm which optimizes controller parameters at each time sample. This type of controller considers a direct adaptive controller due to the direct adaptation of the controller parameters

### 3.1. Controller design

For the neural controller design, two inputs in the input layer are assumed. The reference and the delayed output y(k-1). In the hidden layer, four nodes (neurons) assumed (practically chosen according to the best performances gained

from trying more or less number of nodes), and one output node in the output layer that represents the control command for the plant as is shown in Fig.3.

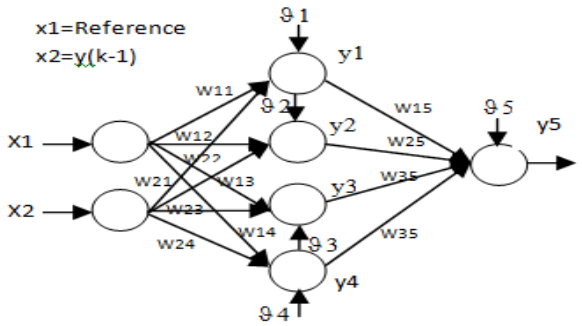



Fig. 3. Neural controller structure

The mathematical model of the controller above will be summarized by the following calculations:-

a) Feedforward computations for the node's output of the hidden layer (y1, y2, y3, y4) and the output node in the output layer which represent the controller's output (y5), and since the value of y5 is bounded in the period 0-1, therefore we have to multiply it by a specific gain whose value depends on some experimental factors and some performance requirements.

$$\begin{aligned} y1 &= sig(x1 * w11 + x2 * w21 - \vartheta 1) \\ y2 &= sig(x1 * w12 + x2 * w22 - \vartheta 2) \\ y3 &= sig(x1 * w13 + x2 * w23 - \vartheta 3) \\ y4 &= sig(x1 * w14 + x2 * w24 - \vartheta 4) \\ y5 &= sig(y1 * w15 + y2 * w25 + y3 * w35 + y4 * w45 - \vartheta 5) \\ u &= y5 * gain \end{aligned} \qquad (5)$$

Where (Wij, Wjk) represent the neuron's weights for the hidden and output layers respectively while $\vartheta j, \vartheta k$ represent the bias weights for hidden and output nodes respectively.

Fig. 4. Represent the software implementation for the equations above which is written by LabVIEW software.

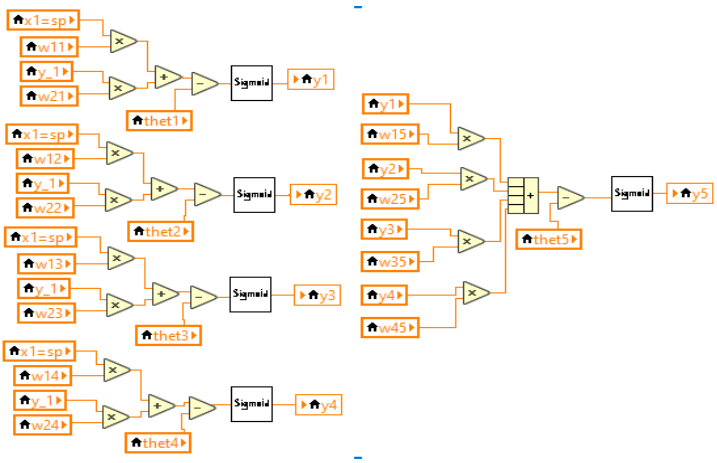



Fig. 4. Controller feedforward software by labVIEW

b) Backpropagation computations or learning algorithm to obtain the optimized weights for the output layer node (w15, w25, w35, w45) and node bias ($\vartheta 5$) through the backpropagation of the gradient error factor ($\delta 5$) as in eq. (6), (7).

$$\delta 5 = \frac{\partial sig\,(y5)}{\partial y5} * error \qquad 6)$$
$$\delta 5 = sig\,(y5) * \big(1 - sig(y5)\big) * error$$

Updating the neuron's weights by adding the old weight with a rate change of each weight ($\Delta w$), ($\Delta \vartheta 5$). As shown in eq. (7) and Fig. 5.

$$\begin{aligned} w15 &= w15 + \Delta w15,\ \Delta w15 = \alpha * y1 * \delta 5 \\ w25 &= w25 + \Delta w25,\ \Delta w25 = \alpha * y2 * \delta 5 \\ w35 &= w35 + \Delta w35,\ \Delta w35 = \alpha * y3 * \delta 5 \\ w45 &= w45 + \Delta w45,\ \Delta w45 = \alpha * y4 * \delta 5 \\ \vartheta 5 &= \vartheta 5 + \Delta \vartheta 5,\ \Delta \vartheta 5 = \alpha * (-1) * \delta 5 \\ \alpha &= \mathrm{mu} * \mathrm{error} \end{aligned} \qquad (7)$$

Where α is the training factor to be chosen with precision, where high value may lead to an unstable controller and a very small value leads to a slow controller, in this experiment study, α defined as a function of an error where the value of α is relatively large when the error value is relatively high To make the controller quickly converge towards achieving the target and decrease its value when the

error value is decreased. An experimental factor (mu) was added to control the speed of the controller.

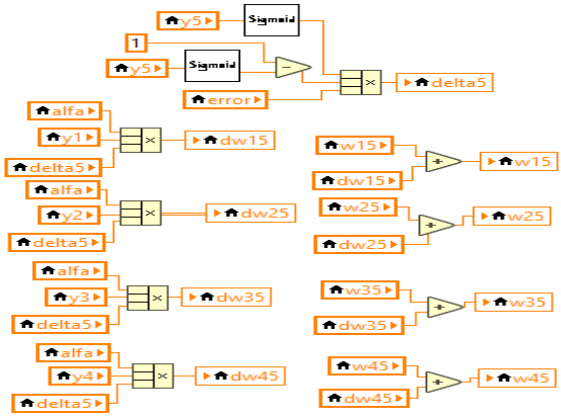



Fig. 5. Output layer weights learning

c) Computations of the gradient error (δ1,δ2,δ3,δ4), for the backpropagation of the hidden layer neurons weights, depending on the output error, where the factor (wij δ5), represents the alternative factor concerning the output error of the system which used to train the hidden layer neurons weights as in eq. (8) and Fig. 6. a.

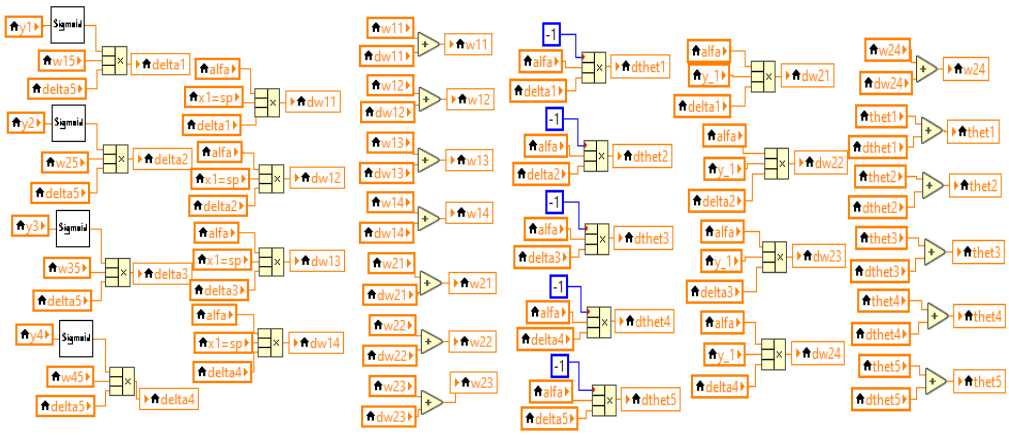
$$\begin{aligned} \delta 1 &= sig(y1) * w15 * \delta 5 \\ \delta 2 &= sig(y2) * w25 * \delta 5 \\ \delta 3 &= sig(y3) * w35 * \delta 5 \\ \delta 4 &= sig(y4) * w45 * \delta 5 \end{aligned} \tag{8}$$

d) Updating the hidden layer neurons weights, by adding the last weights with the rate of change of each weight as in eq. (9) and Fig. 6. a.

$$\begin{aligned} w11 &= w11 + \Delta w11,\ \Delta w11 = \alpha * x1 * \delta 1 \\ w12 &= w11 + \Delta w11,\ \Delta w12 = \alpha * x1 * \delta 2 \\ w13 &= w11 + \Delta w11,\ \Delta w13 = \alpha * x1 * \delta 3 \\ w14 &= w11 + \Delta w11,\ \Delta w14 = \alpha * x1 * \delta 4 \\ w21 &= w21 + \Delta w21,\ \Delta w21 = \alpha * x2 * \delta 1 \\ w22 &= w22 + \Delta w22, \Delta w22 = \alpha * x2 * \delta 2 \\ w23 &= w23 + \Delta w23,\ \Delta w23 = \alpha * x2 * \delta 3 \\ w24 &= w24 + \Delta w14,\ \Delta w24 = \alpha * x2 * \delta 4 \end{aligned} \tag{9}$$

Where x1 represents the system reference, x2 represents the delayed output y(k-1), and the updating of node bias weights ($\vartheta 1, \vartheta 2, \vartheta 3, \vartheta 4$), the same gradient factors ($\delta 1, \delta 2, \delta 3, \delta 4$) were used, as in eq. (10) and Fig. 6. b.

$$\begin{aligned} \vartheta 1 &= \vartheta 1 + \Delta\vartheta 1, \ \Delta\vartheta 1 = \alpha * (-1) * \delta 1 \\ \vartheta 2 &= \vartheta 2 + \Delta\vartheta 2, \ \Delta\vartheta 2 = \alpha * (-1) * \delta 2 \\ \vartheta 3 &= \vartheta 3 + \Delta\vartheta 3, \ \Delta\vartheta 3 = \alpha * (-1) * \delta 3 \\ \vartheta 4 &= \vartheta 4 + \Delta\vartheta 4, \ \Delta\vartheta 4 = \alpha * (-1) * \delta 4 \end{aligned} \tag{10}$$



a) b)

Fig. 6. Hidden layer and bias weights learning

For the smooth tracking of the changes in the system reference, an system first order added as a reference model, which can be represented in form of a delayed signal as shown in eq. no (11) and the LabVIEW software design as in Fig. 7

$$\boldsymbol{yr = 0.818 * yr(k-1) + 0.1818 * ur(k-1)} \tag{11}$$

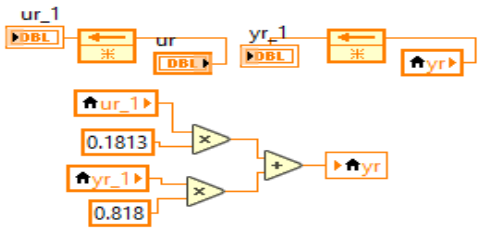



Fig.7. Reference model

### 3.2. Front panel and interface monitoring

Fig. 8. Shows the front panel design or the HMI interface monitor, designed using LabVIEW software, the initial parameters, and experimental factors adjusted and displaying the system response for any reference and disturbances.

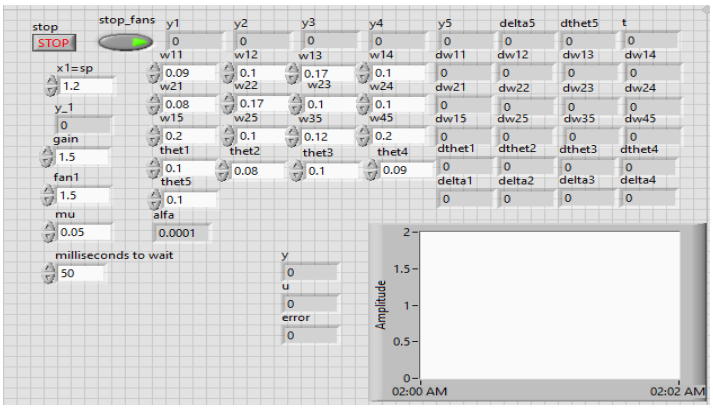
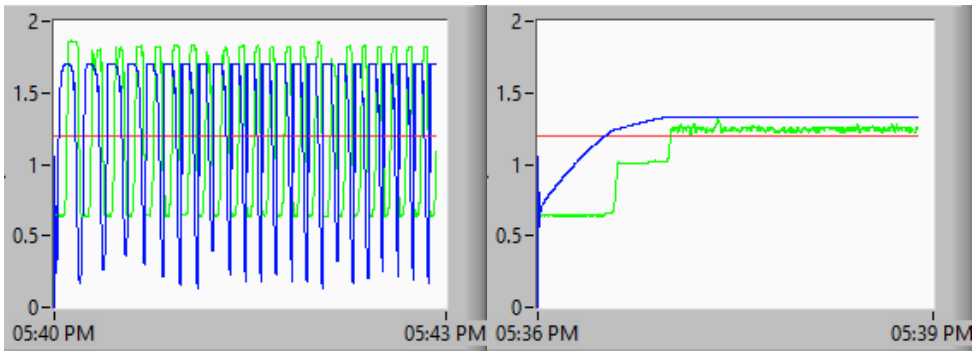



Fig. 8. Front panel interface by LabVIEW

## 4. Real-time results

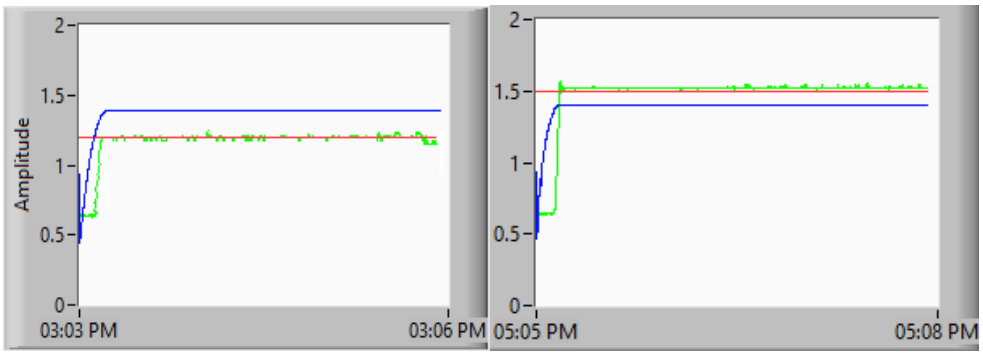
The real-time results for the position control system using a neural controller designed based on the theory of neural networks have illustrated good results in terms of stability and achieving the required performances. The stability computations and analyzes didn't discuss deep in this study, but concisely, the stability of the controller in particular and the system, in general, relies on the value of the learning factor $(\alpha)$, which represents the training speed of the neural controller parameters that directly affect the speed and stability of the controller. In general, choosing a high value for the learning factor may lead to instability of the system and its continuous oscillation, as shown in Fig. 9. a. While choosing a very small value of this factor will lead to a slow controller which may not satisfy the required performance of the controlled system, as shown in Fig. 9. b.




a) b)

Fig. 9. Output response for the position control system using a neural controller with fixed learning factor. (a) for large value of learning factor, (b) for small value of learning factor.

To adapt the learning factor (α). It is calculated as a function of the system output error and multiplies by an experimental factor (mu) which is tested experimentally according to some required performances for the controlled system. The learning factor will adapt according to the error between the system output value and the required reference that is mean, when the system error is high, the learning factor will be relatively high, and the system will quickly move towards reducing the error. Fig. 10. a and b shows the required system output response for a position control system with a neural controller with different values of the reference signal.

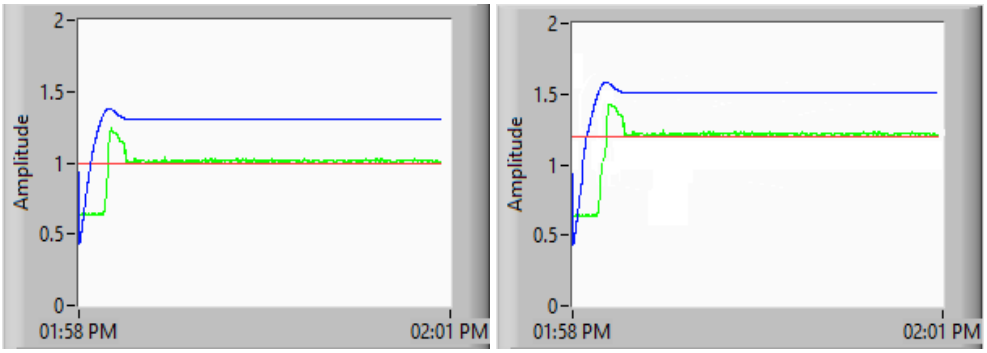



a) b)

Fig. 10. Output response for the position control system using a neural controller and adapting learning factor. (a) with 1.2 set-point, (b) with 1.5 set-point

Compared the same system controlled by using well-tuned classical PID controller. It is clear to see the smaller rising time and zero overshot of system output response in Fig. 10. a and b while using the neural controller, comparing

with slower rising time and high overshot while using the classical PID under the same conditions. As shown in Fig. 11. a and b.




a) b)

Fig. 11. The output response of the position control system using PID controller. (a) with 1 set-point, (b) with 1.2 set-point

## 5. Conclusions

Neural networks as an important part of artificial intelligent theories have become adopted in different industrial applications, not limited to the research application only. Because of their ability to manage many different data. In this practical study, the neural networks theory proposed to implement and test in real-time a neural controller in the laboratory using a position control system. It gave very good results in terms of reference tracking and stability for the controlled system. Especially for the systems that operate with different operational conditions which lead to a variation in the model of the system over time, Compared with classical PID controller which is not recommended for nonlinear systems or systems that operate with different operational conditions if there are some required performances for the controlled system some previous solutions used in such type of systems, like the theory of multiple models, where is the dynamical model of the system is required to be identified very accurately and for more than one operational conditions of the system, where the accuracy and robustness of the controllers used to depend on the accuracy of the identified models, in addition to the multiplicity of control units, which increases the complexity of the control program. As a result, using a neural controller may become a successful choice for many nonlinear systems according to the huge development in the speed of computers and data acquisition cards.

## REFERENCES

[1] Martin T. Hagan, Howard B. Demuth, M. Hudson Beale, O. DE JESÚS1 , “neural network design second edition”.

[2] A. van Eck Conradie, “Aneural paradgim for intelligent process control using evolutionary reinforcment learning ”, PhD Thesis, Department of Chemical Engineering, University of Stellenbosch.

[3] Plagianakos, V., D. Sotiropoulos and M. N. Vrahatis. “Automatic Adaptation of Learning Rate for Backpropagation Neural Networks.” (1998).

[4] D. Chrita, C. Lupu, “multiple modeles control solution for series fans processes (with variable load) ”, U.P.B. Sci. Bull., Series C, Vol. 78, Iss. 2, 2016.

[5] A. Zribi, A. Chtourou, M. Djemel, M.: A new PID neural network controller design for nonlinear processes. J. Circuits Syst. Comput. 27(04), 1850065 (2018)6.

[6] M. I. Mahmoud, B. A. Zalam, M. A. Gomah, “Real-Time Neural Network Speed Controller Implementation on the PLC for a DC Drive”. Article 5, Volume 16, Issue 1, Winter and Spring 2006, Page 45-59, 10.21608/MJEER.2006.64776

[7] Fadhil A. Ali, “Feed Forward Neural Network For Sine Function With Symmetric Table Addition Method Using Labview And Matlab Code”. International Journal on Computational Sciences & Applications (IJCSA) Vol.4, No.2, April 2014

[8] Tarek A. Tutunji, “Parametric System Identification using Neural Networks” Applied Soft Computing (Elsevier Publication) 47 pp 25-261, 2016.

[9] Murat Oduncuoglu, Halil Ibrahim Kurt, “ Application of a Neural Network Model for prediction of Wear properties of Ulterahigh Molecular Weight Polyethylene composites”, International Journal of Polymer Science · May 2015.

[10] T. Poggio and F. Girsi, "Networks for Approximation and Learning," proc. IEEE, vol. 78, no. 9, pp. 1481-1497, Sept. 1990.

[11] J. M. Zurada, Artificial Neural Networks, copright 1992 by west publishing company in the United States of America, pp. 185-208.

[12] P. Shamsollahi and O.P. Malik, "Real-time Implementation and Experimental Studies of a Neural Adaptive Power System Stabilizer”, IEEE Transactions on Energy Conversion, vol. 14, No. 3,September 1999.