Al-khwarizmi Engineering Journal

Al-Khwarizmi Engineering Journal, vol.1, no.1,pp 1-18, (2005)

Neuro-Self Tuning Adaptive Controller for Non-Linear Dynamical Systems

Ahmed Sabah Abdul Ameer Al-Araji
University of Technology

Abstract:

In this paper, a self-tuning adaptive neural controller strategy for unknown nonlinear system is presented. The system considered is described by an unknown NARMA-L2 model and a feedforward neural network is used to learn the model with two stages. The first stage is learned off-line with two configuration serial-parallel model & parallel model to ensure that model output is equal to actual output of the system & to find the jacobain of the system. Which appears to be of critical importance parameter as it is used for the feedback controller and the second stage is learned on-line to modify the weights of the model in order to control the variable parameters that will occur to the system. A back propagation neural network is applied to learn the control structure for self-tuning PID type neuro-controller. Where the neural network is used to minimize the error function by adjusting the PID gains. Simulation results show that the self-tuning PID scheme can deal with a large unknown nonlinearity.

Keyword: Self-Tuning, Neural Network, Adaptive Controller.

1. Introduction:

In many applications, the control engineers face a number of practical difficulties. The large dimensionality of many processes & the significant interaction between variables from the major obstacle to the successful attempts of extending the classical techniques for the design of controllers for monovaraible plants to multivariable ones. The development of computer-aided techniques to design controllers aim to reduce interaction before applying classical theory to the individual loops. Most existing techniques are based on the design of tunable set-point tracking controllers with the dominance PI (Proportional, Integral) & PID (Proportional, Integral, Derivative) controllers in industry & certain assumptions such as linearity & interactions with in the controlled process have to be made [1,2]. Neural networks have broad applicability to real world problems, such as in pattern recognition,diagnostic, optimization, system identification & control. They have already been successfully applied in many industries, as they are well suited for predication or forecasting because of their abilities in identifying patterns or trend in data [3,4].

The neural network model can be used in control strategies that require a global model of the system forward or inverse dynamics, and these models are available in the form of neural networks, which have been trained using neural based system identification techniques. Papers by: Narandra & Parthasarathy [5,6] are some of those that can be referred to as the application of neural networks for

system identification. The generalized learning method attempts to produce the inverse of a plant over the entire state space using off-line training while in the specialized architecture the training is on-line and uses error back-propagation through the plant to learn the plant inverse dynamics over a small operating region. Behera et al [7] in their paper are concerned with the design of a hybrid controller structure consisting of the adaptive control law and neural network based learning scheme for adaptation of time varying controller parameters. The global stability of the closed-loop feedback system is guaranteed provided the structure of the robot-manipulator dynamics model is exact. Generalization of the controller over the desired trajectory space has been established using an on-line weight learning scheme. The advantage of a neuron-adaptive hybrid control scheme is the high precision and better accuracy and computationally less intensive control scheme. Also for Self-Tuning Control (STC), Chen [8] used back-propagation trained neural network within a self-tuning control system to control Single-Input Single-Output (SISO) feedback linearizable system. Another approach is given in [9], where a neural network is used to tune the parameters of a conventional controller in an on-line way.

The organization of the paper is as follows: Section two describes the use of FNNs to learn to act as input-output model. Model (NARMA-L2) for system identification are examined with the corresponding neural nets and learning mechanism used for this purpose. Section three represents the core of the present paper. It is suggested using self-tuning PID neural controller. Illustrative example that clarify the features of the proposed strategy are given in section four, where the example is discussed in detail. Finally, section five contains the conclusions of the entire work.

2- Identification of Dynamical System:

The system identification and modeling is a very important step in control applications since it is a pre-requisitic for analysis and controller design. Due to the nonlinear nature of most of the processes encountered in many engineering applications there has been extensive research covering the field of nonlinear system identification [10]. This section focuses on nonlinear system identification using the model of multi-layered feedforward neural network, NARMA-L2 model. The neural network is trained using Back-Propagation Algorithm. To describe the process by using artificial neurons as basic building elements for the development of multi-layered and higher order neural network, the feedforward neural networks are widely used. The learning scheme for feedforward neural networks presented in this section includes the generalized Delta Rule based algorithms for Error Back Propagation for multi-layers neural networks [11]. A feedforward neural network can be seen as a system transforming a set of input patterns into a set of output patterns, and such a network can be trained to provide a desired response to a given input. The network achieves such a behavior by adapting its weights during the learning phase on the basis of some learning rules. The training of feedforward neural networks often requires the existence of a set of input and output patterns called the training set [11] and this kind of learning is called supervised learning. The feedforward network used here has two layers, the first is the hidden layer and the second is the output layer where

each unit in the hidden layer has a continuous sigmoidal nonlinearity [12] and the output node has linear activation function.

NARMA-L2 Model Identification:

Nonlinear input-output behavior can be well approximated by NARMA-L2 (Nonlinear Auto Regressive Moving Average-Linear) two model which can be expressed as [13]:

$$y_p(k+1) = f[y_p(k),...y_p(k-n+1),u(k-1),...u(k-n+1)] + g[y_p(k),..y_p(k-n+1),u(k-1),...u(k-n+1)]\times u(k) \quad ...(1)$$

where n is the order of the system.

The NARMA-L2 model requires only two neural networks to approximate the function f and g. Each of the two functions, however has (2n-1) inputs By using NARMA-L2 model the weights of the neural networks are adjusted in a similar manner when using the NARMA model.

$$y_p(k+1) = \bar{F}[y_p(k),...,y_p(k-n+1),u(k),...u(k-n+1)] \quad ...(2)$$

The difference between them is that NARMA-L2 model consists of two functions f[-] and g[-] in equation (1) while one neural network is needed for NARMA model.

The first step in the identification procedure using feedforward neural network is quite straightforward with serial parallel model and at each instant of time. The past inputs and the past outputs of the system are fed into the neural network as shown **Fig (1).**

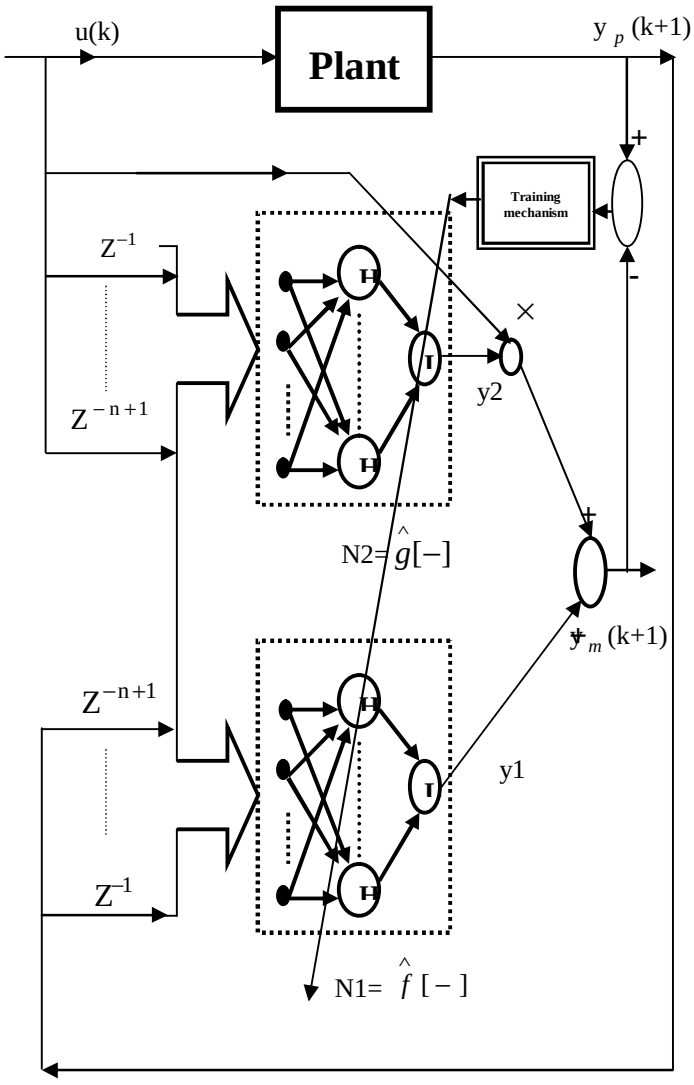

Fig (1): NARMA-L2 Identification Model

Serial-Parallel Configuration

The network's output yields the prediction error:

$$e(k+1) = y_p(k+1) - y_m(k+1) \quad \ldots(3)$$

the identification model for the NARMA-L2 model can be better illustrated as **Fig.1**, where $\overline{X}$ represents the input vector of the networks N1 and N2 (the argument of $\hat{f}[-]$ and $\hat{g}[-]$). The learning (training) algorithm is usually based on the minimization (with respect to the network weights) of the following objective cost function:

$$E = \frac{1}{2}\sum_{i=1}^{np}(e^i(k+1))^2 = \frac{1}{2}\sum_{i=1}^{np}(y_p^i(k+1) - y_m^i(k+1))^2$$

… (4)

Where np is number of patterns, e^i is the error of each step, y_p^i is the actual output of the plant of each step and y_m^i is the model output of the plant of each step. From **Fig.1**, it is important to note that the error between the desired output and the estimated neural network output needed to apply a supervised learning algorithm which is not available at the output N1 and N2. Hence, a little modification must be done to fit the algorithm to our case. This can be simply done by back-propagating the error at the output of the NARMA-L2 model (between $y_p(k+1)$ and $y_m(k+1)$) to the output of N2 after multiplying it by u(k) and to the output of N2 after multiplying it by u(k) and to the output of N1 directly. The second step in the identification procedure using the same feedforward neural network that its learned off-line with serial –parallel model. But now with parallel model and at each instant of time, the past inputs and the past model outputs of the neural network are fed into the same neural network as shown **Fig.2**. In order to minimize the error between the actual output & the model output and is equal to zero approximately then the model (NARMA-L2) will complete the same actual output response. When identification of the plant is complete then g[-] can be approximated by $\hat{g}[-]$ and f[-] by $\hat{f}[-]$ and the NARMA-L2 model of the plant can be described by equation (5) below:

$$y_m(k+1) = \hat{f}[y_p(k),\ldots y_p(k-n+1), u(k-1),\ldots u(k-n+1)] + \hat{g}[y_p(k),\ldots y_p(k-n+1), u(k-1),\ldots u(k-n+1)] \times u(k)$$

… (5)

Likewise if $\hat{g}[-]$ is sign definite in the operating region then the $\hat{g}[-]$ network can be used as the jacobain of the plant as given by equation (6).

$$\hat{g}[y_p(k),\ldots y_p(k-n+1), u(k-1),\ldots u(k-n+1)] \quad \ldots(6)$$

where the jacobain is:

$$jacobain = \frac{\partial y_p(k+1)}{\partial u(k)} = \hat{g}[-] \quad \ldots(7)$$

The sign definiteness of $\hat{g}[-]$ in the operating region (the region of interest) ensures the uniqueness of the plant inverse at that operating region [14]. Now by using equation (5) as the model of the plant identifier and equation (6) as the jacobain of the plant.

3- The Controller Design:

The control of nonlinear plants is considered in this section. The approach used to control the plant depends on the information available about the plant and the control objectives. The information of the unknown nonlinear plant can be known by the input-output data only and the plant is considered as (NARMA-L2 model). The first step in the procedure of the control structure is the identification of the plant from the input-output data, and then is used to

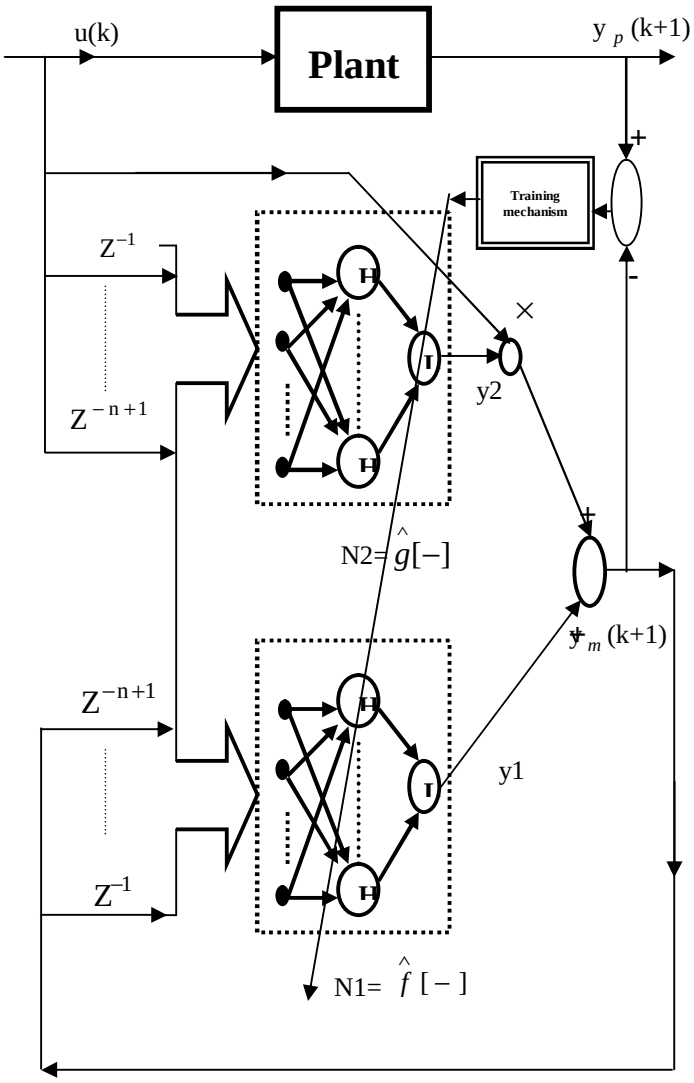

Fig (2): NARMA-L2 Identification Model

Parallel Configuration

find the jacobain of the plant as in section two. The feedback neural controller is used based on the minimization of the error between the desired "set-point" & the actual output plant in order to achieve good tracking of the reference signal and to use minimum effort. The integrated control structure that consists of the identifier of the plant and a self-tuning PID controller type neural networks thus brings together the advantages of the neural model with the robustness of feedback. The general structure of the neural controller type can be given in the form of the block diagram shown in **Fig. 3.**And this structure of the proposed controller can be applied to the nonlinear plants.It consists of:

1. Identifier as Feedforward Neural Networks (NARMA-L2) Model.
2. Self-Tuning PID Feedback Controller Type Neuro Controller.

In the following sections, the proposed controller will be explained in detail.

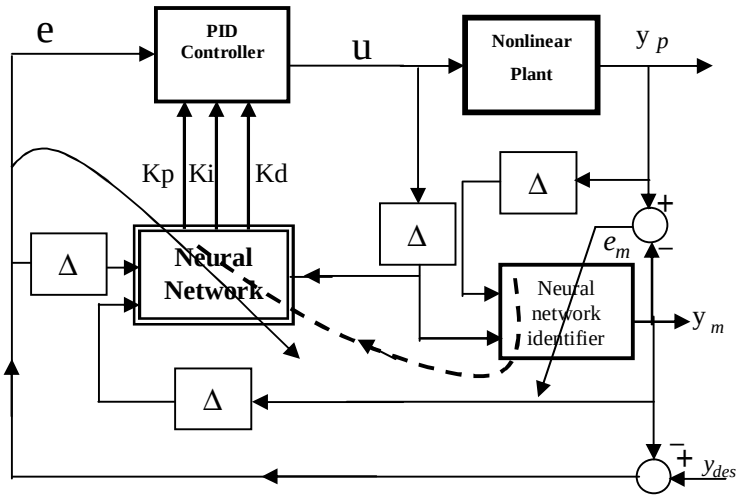

Fig (3): The general structure of the proposed controller

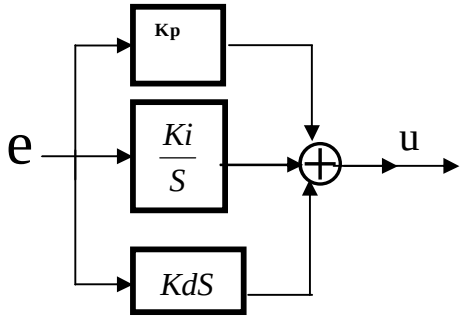

Fig (4): General Configuration of PID controller

Self-Tuning PID Type Neuro-Controller:

The feedback neural controller is very important because it is necessary to stabilize the tracking error dynamics of the system when the output of the plant is drifted from the input reference [14]. The adaptive Self-Tuning technique is to adjust the parameters of the PID feedback controller by using neural networks, so that, the output of the plant follows the output of the predefined desired model. In the following section, a self-tuning neuro-control scheme is discussed in which a neural network is used to tune the parameters of a PID controller referred to as the self-tuning PID neuro-control scheme. The PID control configuration is illustrated in **Fig. 4,** where Kp is the proportional gain, Ki is an integral gain, & Kd is the derivative gain, which are adjusted to achieve the desired output.The control input U(k) of the PID controller is given by equation (8):

$$U(k) = Kp e(k) + Ki\sum e(k) + kd(e(k) - e(k-1)) \quad (8)$$

The proposed control structure for the self-tuning PID learning where the network is used to minimize the error function by adjusting the PID gain. The discrete-time version of PID controller is described by:

$$U(k) = u(k-1) + Kp[e(k) - e(k-1)] + Ki\, e(k) + Kd[e(k) - 2e(k-1) + e(k-2)] \quad (9)$$

Where Kp, Ki, & Kd denote the PID gains.

$$e(k) = y_{des}(k) - y_m(k) \quad (10)$$

$y_{des}(k)$ is a desired output.

$y_m(k)$ is the model output.

In order to derive the self-tuning algorithm of the PID controller, a cost function **E** should be minimize and it is defined as::

$$E = \frac{1}{2} e^2(k+1) \quad (11)$$

Using two layers neural network as shown in **Fig.5**, that will realize the learning rule to find the suitable PID gains. The multi-layered feedforward neural network shown in **Fig.5** is composed of many interconnected processing units called neurons or nodes [10]. where:

V : Weight matrix.

W : Weight matrix.

L : Denotes linear node.

H : Denotes nonlinear node with sigmoidal function.

As can be seen the net consists of three layers: An input layer (buffer layer), a single hidden layer with biases and a linear output layer with bias too. The neurons in the input layer simply store the scaled input values. The hidden layer neurons perform two calculations. To explain these calculations, consider the general j'th neuron in the hidden layer shown in **Fig.6.** The inputs to this neuron consist of an ni – dimensional vector $\overline{X}$ (ni is the number of the input nodes) and a bias whose value is "-1"[10]. Each of the inputs has a weight $V_{j,i}$ associated with it. The first calculation within the neuron consists of calculating the weighted sum net_j of the inputs as:

$$net_j = \sum_{i=1}^{ni} V_{j,i} \times X_i + V_{j,ni+1} \times bias \quad (12)$$

Next the output of the neuron h_j is calculated as the continuous sigmoid function of the net_j as:

$$h_j = H(net_j) \quad (13)$$

$$H(net_j) = \frac{2}{1 + e^{-net_j}} - 1 \quad (14)$$

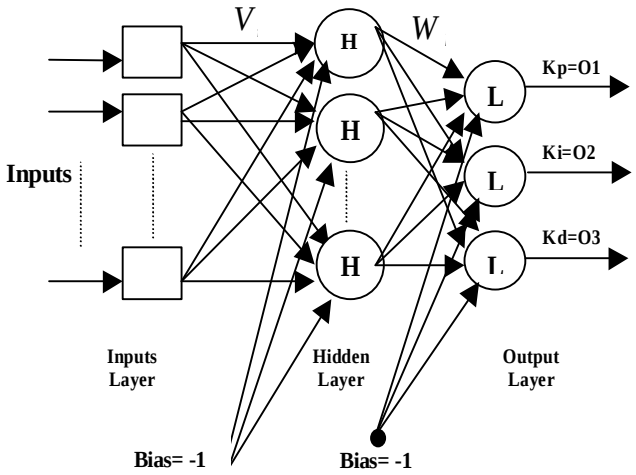

Fig (5): Neural network is used to determine the PID gains

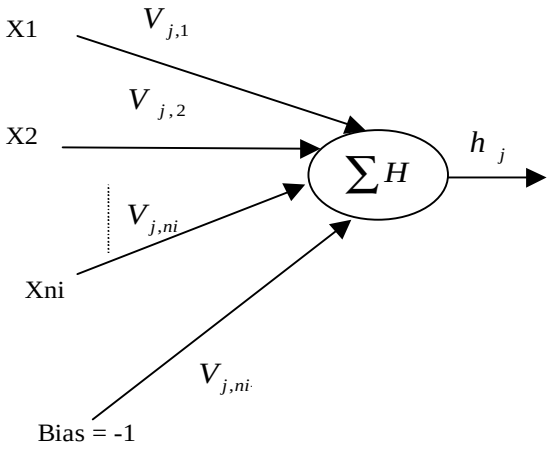

Fig (6): Neuron j in the hidden layer.

Once the outputs of the hidden layer are calculated, they are passed to the output layer. In the output layer, a single linear neuron is used to calculate the weighted sum (neto) of its inputs (the output of the hidden layer as in equation(15).

$$\text{neto}_k = \sum_{j=1}^{nh} W_{kj} \times h_j + W_{k,nh+1} \times bias \quad (15)$$

where nh is the number of the hidden neuro (nodes) and W_{kj} is the weight between the hidden neuron h_j and the output neuron.The single linear neuron, then, pass the sum (neto_k) through a linear function of slope 1 (another slope can be used to scale the output) as:

$$O_k = L(neto_k), \text{whereL(x)=x} \quad (16)$$

Thus the outputs at the output layer are Kp, Ki, & Kd which are denoted by O1, O2, & O3 respectively. Based on the steepest descent (gradient) method, at the output layer:

$$\Delta w_{kj}(k+1) = -h\frac{\partial E}{\partial w_{kj}} + a\Delta w_{kj} \qquad (17)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{w}_{\mathrm{kj}}} = \frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{k}}} \times \frac{\partial \mathrm{net}_{\mathrm{k}}}{\partial \mathrm{w}_{\mathrm{kj}}} \qquad (18)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{w}_{\mathrm{kj}}} = \frac{\partial \mathrm{E}}{\partial \mathrm{o}_{\mathrm{k}}} \times \frac{\partial \mathrm{o}_{\mathrm{k}}}{\partial \mathrm{net}_{\mathrm{k}}} \times \frac{\partial \mathrm{net}_{\mathrm{k}}}{\partial \mathrm{w}_{\mathrm{kj}}} \qquad (19)$$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial u(k)} \times \frac{\partial u(k)}{\partial o_k} \times \frac{\partial o_k}{\partial net_k} \times \frac{\partial net_k}{\partial w_{kj}} \qquad (20)$$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial y_m(k+1)} \frac{\partial y_m(k+1)}{\partial u(k)} \frac{\partial u(k)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} \qquad (21)$$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial y_m(k+1)} \frac{\partial y_m(k+1)}{\partial u(k)} \frac{\partial u(k)}{\partial o_k} f'(net_k) o_j \qquad (22)$$

$$jacobain = \frac{\partial y_m(k+1)}{\partial u(t)} = \hat{g}[-] \qquad (23)$$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial e(k+1)} \times \frac{\partial e(k+1)}{\partial y_m(k+1)} \times \hat{g}[-] \times \frac{\partial u(k)}{\partial o_k} \times f'(net_k) \times o_j \qquad (24)$$

from equation (10 & 11) substituted in equation (24)

$$\frac{\partial \mathrm{E}}{\partial \mathrm{w}_{\mathrm{kj}}} = -\mathrm{e}(\mathrm{t}+1) \times \hat{\mathrm{g}}[-] \times \frac{\partial \mathrm{u}(\mathrm{t})}{\partial \mathrm{o}_{\mathrm{k}}} \times \mathrm{f}'(\mathrm{net}_{\mathrm{k}}) \times \mathrm{o}_{\mathrm{j}} \qquad (25)$$

where:

$f'(net_k) = C$ for linear activation function with gain is limited between (0 to 1).

$$\frac{\partial u(k)}{\partial o_k} = \begin{bmatrix} e(k) - e(k-1) & k=1 \\ e(k) & k=2 \\ e(k) - 2e(k-1) + e(k-2) & k=3 \end{bmatrix} \qquad (26)$$

Then substituted equation (25) in equation (12)

$$\Delta w_{kj}(k+1) = he(k+1)\hat{g}[-]\frac{\partial u(k)}{\partial o_k} C \times o_j + a\Delta w_{kj} \qquad (27)$$

at the hidden layer:

$$\Delta v_{ji}(k+1) = -h\frac{\partial E}{\partial v_{ji}} + a\Delta v_{ji} \qquad (28)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{v}_{\mathrm{ji}}} = \frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{j}}} \times \frac{\partial \mathrm{net}_{\mathrm{j}}}{\partial \mathrm{v}_{\mathrm{ji}}} \qquad (29)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{v}_{\mathrm{ji}}} = \frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{j}}} \times \mathrm{o}_{\mathrm{j}} \qquad (30)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{v}_{\mathrm{ji}}} = \mathrm{o}_{\mathrm{j}} \times \sum_{\mathrm{k}=1}^{\mathrm{K}} \frac{\partial \mathrm{E}}{\partial \mathrm{o}_{\mathrm{j}}} \times \frac{\partial \mathrm{o}_{\mathrm{j}}}{\partial \mathrm{net}_{\mathrm{j}}} \qquad (31)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{v}_{\mathrm{ji}}} = \mathrm{o}_{\mathrm{j}} \times \sum_{\mathrm{k}=1}^{\mathrm{K}} \frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{k}}} \times \frac{\partial \mathrm{net}_{\mathrm{k}}}{\partial \mathrm{o}_{\mathrm{j}}} \times \frac{\partial \mathrm{o}_{\mathrm{j}}}{\partial \mathrm{net}_{\mathrm{j}}} \qquad (32)$$

$$\frac{\partial \mathrm{E}}{\partial \mathrm{v}_{\mathrm{ji}}} = \mathrm{o}_{\mathrm{j}} \times \sum_{\mathrm{k}=1}^{\mathrm{K}} \frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{k}}} \times \mathrm{w}_{\mathrm{kj}} \times \mathrm{f}'(\mathrm{net}_{\mathrm{j}}) \qquad (33)$$

from the derives of $\frac{\partial \mathrm{E}}{\partial \mathrm{net}_{\mathrm{k}}}$ in equation (19), we get:

$$\frac{\partial E}{\partial net_k} = -e(k+1) \times \frac{\partial y_m(k+1)}{\partial u(k)} \times f'(net_k) \times \frac{\partial u(k)}{\partial o_k} \qquad (34)$$

from equation (34) substituted in equation (37)

$$\frac{\partial E}{\partial v_{ji}} = -o_j \sum_{k=1}^{K} e(k+1) \frac{\partial y_m(k+1)}{\partial u(k)} f'(net_k) \frac{\partial u(k)}{\partial o_k} w_{kj} f'(net_j) \qquad (35)$$

then equation (35) substituted in equation (28)

$$\Delta v_{ji}(k+1) = ho_j \sum_{k=1}^{K} e(k+1)\hat{g}[-] C \frac{\partial u(k)}{\partial o_k} w_{kj} f'(net_j) + \alpha \Delta \mathrm{v}_{\mathrm{ji}} \qquad (36)$$

4- Case Study:

In this section, an example is taken to clarify the features of the neural controller explained in section three. In this example, the controller structure is applied to the plant whose difference equation is:

$$y_p(k+1) = 0.8Sin(2y_p(k)) + 1.2u(k) \quad (37)$$

This plant has been adopted from [8 & 14]. For the open loop response of the plant $y_p(k)$ to the input signal u(k) is shown in **Fig 7-a and b** respectively.

The plant response is very oscillatory when the input amplitude $|u(k)| \geq 0.4$.

To use the proposed controller first a neural network is trained for the identification the plant dynamics.

There are two stages the first is a series-parallel configuration NARMA-L2 model identification structure as that in **Fig.1** is used. The model is described by:

$$y_m(k+1) = N1[y_p(k)] + N2[y_p(k)]u(k) \quad (38)$$

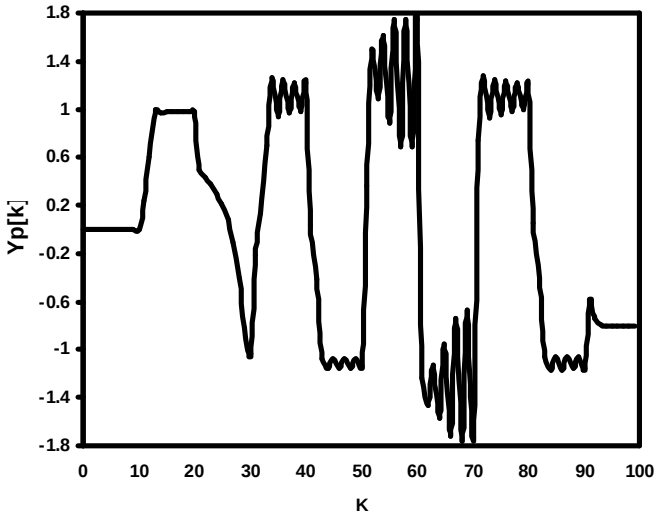

Fig (7-a): The open loop response

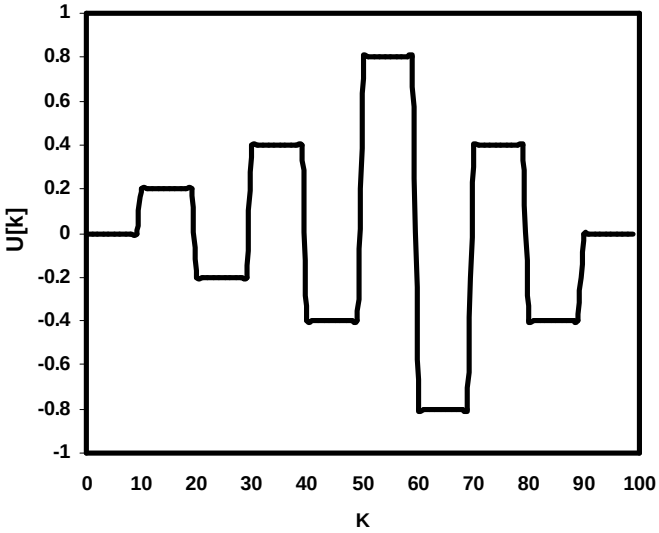

Fig (7-b): The corresponding input signal

where N1[-] and N2[-] are multi-layered neural networks which approximate $\hat{f}[-]$ and $\hat{g}[-]$ of equation (5), respectively. Since each of N1[-] and N2[-] has three inputs $y_p(k)$ (see equation (38)), the initial guess of the number of hidden nodes is three for each network. Using a random input sequence u(k) with $|u(k)| \leq 1$ a training set of 100 patterns input-output used with learning rate $h1$ for N1 and $h2$ for N2 and both were taken to be equal to 0.3. During the training phase, the training set has been presented to the network many times. A training event corresponding to a single pass over of the entire training set is called a training epoch or training cycle. However, for this example after 2000 epochs the Average System Error (ASE) computed for the latest epoch, which is described by equation (39) was 2.77×10^{-6}.

$$ASE = \frac{1}{2np}\sum_{i=1}^{np}\left(y_p{}^i(k+1) - y_m{}^i(k+1)\right)^2 \quad (39)$$

where np total number of patterns which is equal to 100 here. **Fig.8-a** compares the time response of the series-parallel model of equation (38) with the actual plant output for the input as a learning set. While **Fig.8-b** compares the time response of the series-parallel model of equation (38) with the actual plant output for the input applied as testing set generated from equation (40).

$$u(k) = 0.5\times\sin(\frac{2pk}{10}) + 0.5\times\sin(\frac{2pk}{20}) \quad (40)$$

The second stage is a parallel configuration NARMA-L2 model identification structure as that in **Fig 2** is used. To guarantee the model output is equal to the actual output and also to find the jacobain of the plant.

$$y_m(k+1) = N1[y_m(k)] + N2[y_m(k)]u(k) \quad (41)$$

where N1[-] and N2[-] are multi-layered neural networks which approximate $\hat{f}[-]$ and $\hat{g}[-]$ of equation (5), respectively. Since each of N1[-] and N2[-] has three input $y_m(K)$ (see equation (41)). Using the same random input sequence u(k) with $|u(k)| \leq 1$ a training set of 100 patterns input-output used on the same the neural networks N1[-] & N2[-] that there are learned off-line with serial-parallel identification with learning rate $h1$ for N1 and $h2$ for N2 and both were taken to be equal to 0.3. During the training phase, the training set has been presented to the network many times. However, for this example after 5000 epochs the Average System Error (ASE) computed for the latest epoch, which is described by equation (39) was 1.13×10^{-6}.

Fig.9-a compares the time response of the parallel model of equation (41) with the actual plant output for the input as a learning set, while **Fig.9-b** compares the time response of the parallel model of equation (36) with the actual plant output for the input u(k) applied as testing set generated from equation (35). Also **Fig.10** shows a plot of the coefficient of u(k) which is $\hat{g}[-]$ for the NARMA-L2 models as a function of time with values computed using the corresponding network $N2[y_p(k)]u(k)$, when a random input sequence u(k) with $|u(k)| \leq 1$ has been applied to the model. As shown in **Fig.10**, $\hat{g}[-]$ is sign definite in the region of interest. This means that the plant is invertable, or in other words; the model output $y_m(k+1)$ is monotonic with respect to

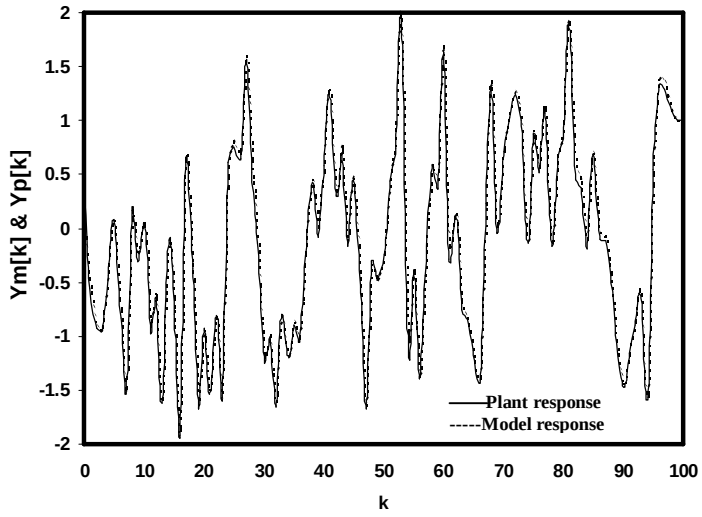

Fig (8-a): The response of the plant & the serial-parallel NARMA-L2 identification model for learning patterns

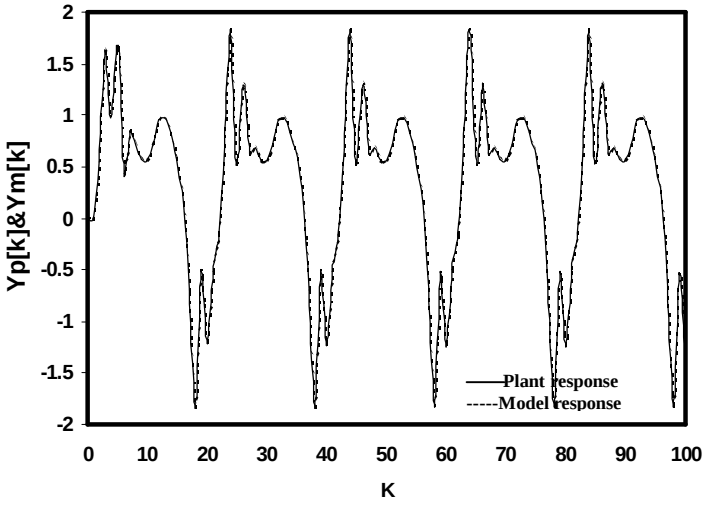

Fig (8-b): The response of the plant & the serial-parallel NARMA-L2 identification model for testing patterns

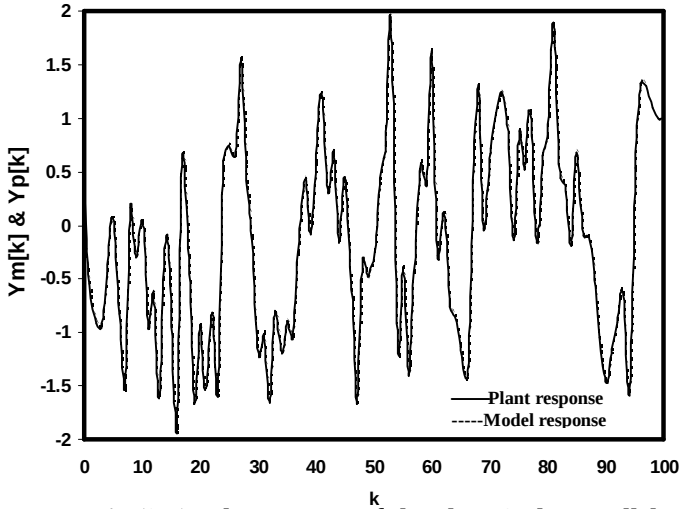

Fig (9-a): The response of the plant & the parallel NARMA-L2 identification model for learning patterns

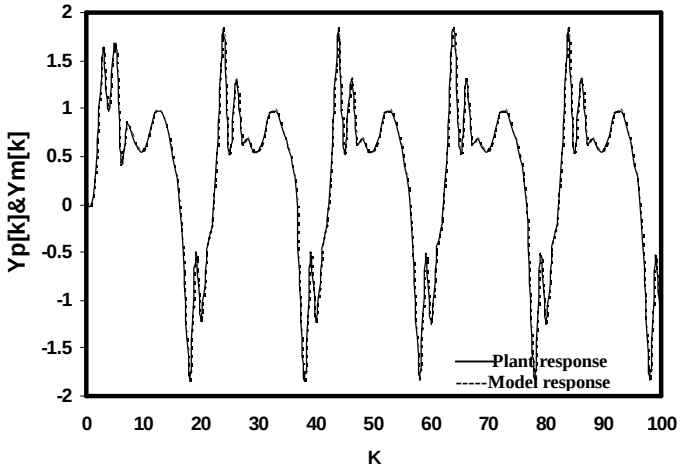

Fig (9-b): The response of the plant & the parallel NARMA-L2 identification model for testing patterns

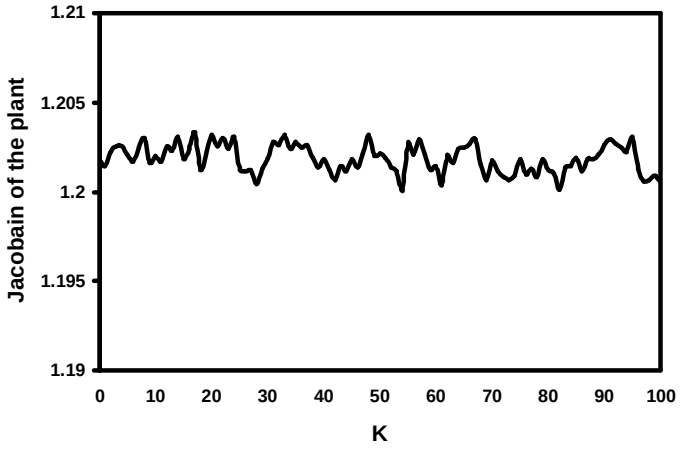

Fig (10): Estimated plant jacobain

u(k). The variation of $\hat{g}[-]$ is approximately around 1.2 as it is expected. This can be explained easily by noting the fact what $\hat{g}[-]$ resembles the plant jacobain is equal to as equation (7) and for this example $\frac{\partial y_p(k+1)}{\partial u(k)} = 1.2$.

To apply the proposed structure of controller after good learning of the identifier as $y_m \approx y_p$. It used the desired trajectory and the training done by repeating the desired trajectory cycles over 26000 times. The neural networks are used to minimize the performance error between the reference and the model output, where the model output is similar to the actual output. Convergence is achieved when the performance error falls below a pre-specified value. After training, it can be observed that the actual output of the plant is following the desired trajectory and the model output is the same as the actual output in **Figs.11 & 12.** And also, the gains of the PID self-tuning neural controller as shown in **Fig. 13 -a, b, & c** Kp, Ki, & Kd

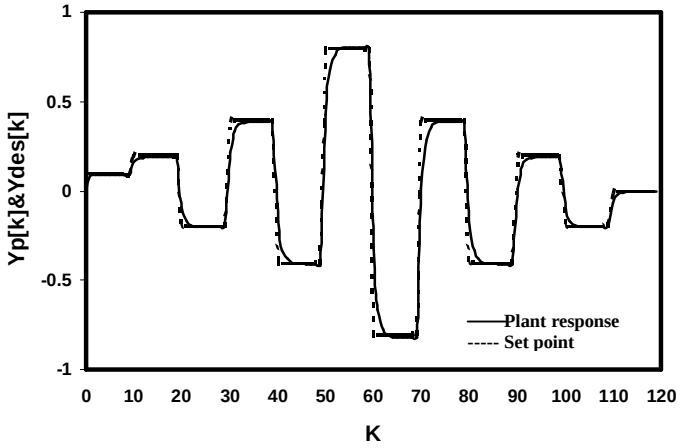

Fig (11): The response of the plant with the set point

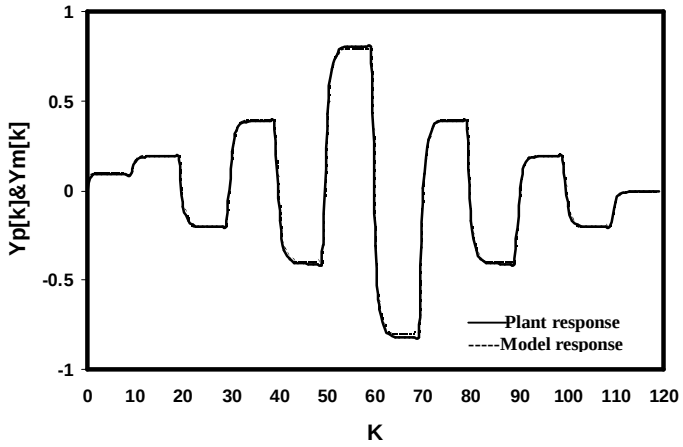

Fig (12): The response of the plant & the response of the model

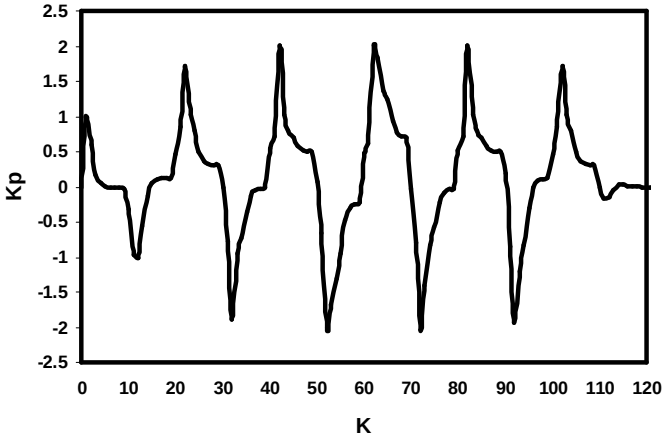

Fig (13-a): Kp gain of PID controller

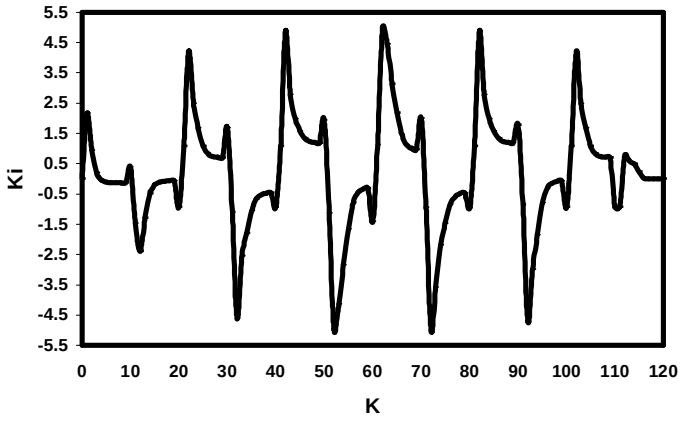

Fig (13-b): Ki gain of PID controller

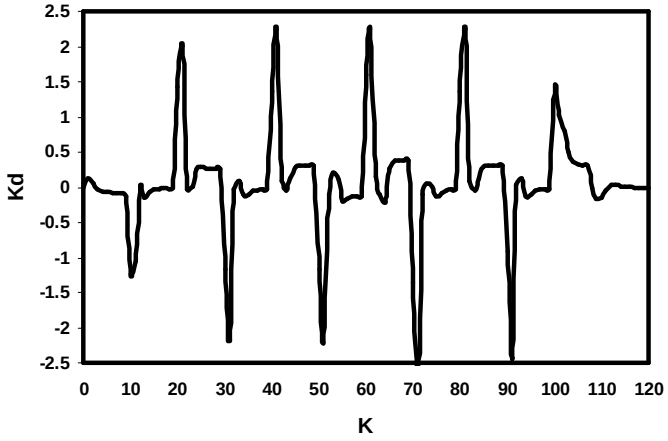

Fig (13-c): Kd gain of PID controller

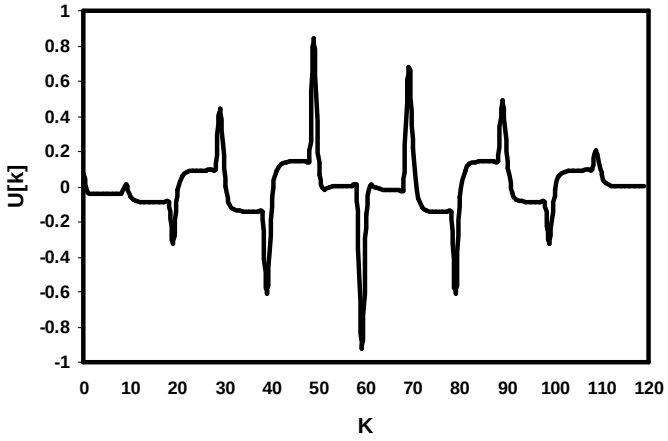

Fig (14): The control signal of the PID controller

respectively. And the feedback control action as shown in **Fig14.**

5- Conclusion:

The structure of the neural controller with an identifier based on neural NARMA-L2 model that is learned off-line with two configuration serial-parallel & parallel and applied the algorithm of the self-tuning PID neural controller as the proposed structure of controller and successfully simulated to nonlinear system as the example. Using neural NARMA-L2 model as a nonlinear model of the plant provides a simple check on the model jacobain, which appears to be of critical importance as it is used for the feedback controller. The on-line identifier NARMA-L2 model of the plant is used to updated of the weights of the identifier by using (BPA) in order to guarantee that model output approaches the actual output. Using PID feedback controller with self-tuning neural to adjust the parameters of the controller. So that, the output of the plant follows the output of the predefined desired input and (BP) algorithm is used to learn the model. The proposed control structure has shown the ability to minimize the error between the desired output and the actual output of the plant as well as the control action, excellent set point tracking, as it was clear when applied to the example. The simulation example in this paper is implemented using Turbo C++ programming language together with Microsoft Excel.

Nomenclature

Symbol	*Description*
E	Summation of error
e	The error between reference input and the model output
$\hat{f}[-], \hat{g}[-], N1[-], N2[-]$	Neural input-output mapping functions
H	Sigmoidal activation function of the hidden nodes
k	Discrete time instant
L	Linear activation function of the output node
n	Plant order
net_j	The weighted sum of the inputs of the node j in the hidden layer
$neto$	The weighted sum of the inputs of the output node
nh	Number of nodes in hidden layer
ni	Number of nodes in input layer
u	Manipulated input
V	Weight matrix between the input and the hidden layer
W	Weight matrix between the hidden and the output layer
$\overline{X}$	The input vector for the input layer
y_{des}	The desired output
y_p	Plant output
y_m	Model output
P	Number of patterns in the training set
h	Learning rate of the neural network

Abbreviations

ASE	Average System Error
BPA	Back Propagation Algorithm
FBNC	Feedback Neural Controller
FFNC	Feedforward Neural Controller
NARMA	Nonlinear Auto Regressive Moving Average
PID	Proportional Integral Derivative
SISO	Single-Input Single-Output
STC	Self-Tuning Control

References

1. Porter B. & Jones A.H.," Design of tunable digital set-point tracking PI controller using step response matrices for gas turbain" AlAA Guidance, Naving – 1986.

2. Porter B. & Jones A.H.," Genetic tuning of digital PID controllers" Electronics Letters: Vol. 28, PP 843-844, 1992.

3. Patterson D.W. "Artificial neural networks, theory & applications" Prentice Hall 1996.
4. Lightbody G. and Irwin G.W." Direct neural model reference adaptive control" IEEE Proce. On control theory and applications, Vol. 142; no. 1, pp. 31-42; Jun 1995.
5. K. S. Narendra and K. parthasarathy, "Identification and control of dynamical systems using neural networks," IEEE Trans. Neural Networks, vol. 1,pp. 4-27, 1990.
6. K. S. Narendra and K. parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," IEEE Trans. Neural Networks, vol. 2 no. 2, pp. 252-262, 1991.
7. L. Behera, S. Chaudhury, and M. Gopal, "Neuro-adaptive hybrid controller for robot-manipulator tracking control" IEE Proc. Control Theory Appl. Vol.143, no. 3, pp.270-275, 1996.
8. F. C. Chen, "Back-propagation neural networks for nonlinear self-tuning adaptive control," IEEE Control Systems Magazine, vol. 10, no. 3, pp. 44-48, 1990.
9. S. Omatu, M. Khalid, and R. Yusof, Neuro-Control and its Applications. London: Springer-Velag, 1995.
10. S. A. Billings, "Identification of nonlinear systems a survey," IEE Proc., vol. 3,pp. 272-285,1980.
11. N. B. Karayiannis and A.N. Venetsanopoulos, Artificial Neural Networks Learning Algorithms, Performance Evaluation, and Applications. London: Kluwer Academic Publishers. 1993.
12. K. J. Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, "Neural networks for control systems—A survey," Automatica, vol. 28, no. 6, pp. 1083-1112, 1992.
13. K. S. Narendra and S. Mukhopadhyay, "Adaptive control using neural networks and approximate models," IEEE Trans.

Neural Networks, vol. 8, no. 3, pp. 475-485, 1997.

14. Ahmed S. Al-Araji "A Neural controller with a pre-assigned performance index" M.Sc. Thesis, University of technology, November 2000.

المسيطر المتكيف ذو التنغيم التلقائي العصبي للانظمة الديناميكية اللاخطية

احمد صباح عبد الامير الاعرجي
الجامعة التكنولوجية

الخلاصة:

أن هيكلية المسيطر العصبي مع المعرف (Identifier) الذي أساسه النموذج العصبي (NARMA-L2) يتم تعليمه بطريقة (off-line)مع صيغتين التوالي المتوازي و المتوازي وتطبيق خوارزمية التنغيم التلقائي العصبي للمسيطر(PID) كمقترح لبناء هيكلية المسيطر.

أن النموذج العصبي (NARMA-L2) هو نموذج لأخطي يصف المنظومة أللاخطية ويستخدم لتحقق من (Jacobain) للمنظومة و التي تعتبر من العناصر المهمة و الحرجة في إيجاد إشارة التغذية العكسية.

أن المعرف (NARMA-L2) يتم أيضا تعليمه (on-line) لتحديث الأوزان (Weights) للنموذج بطريقة خوارزمية الانتشار العكسي العامة لكي يصبح النموذج مطابق الى المنظومة أللاخطية.

يستخدم المسيطر الراجع العصبي ذات التنغيم التلقائي لتعبير عناصر المسيطر (PID) لكي يتبع إخراج المنظومة الحقيقية الإدخال المطلوب وباستخدام أيضا" خوارزمية الانتشار العكسي العامة.

أن هيكلية المسيطر المقترح يستخدم لتقليل الخطاء بين الإخراج المرغوب و الإخراج الحقيقي للمنظومة.

لقد تم الحصول على نتائج ممتازة باستخدام المسيطر المقترح عندما طبق هذا المسيطر على المنظومة أللاخطية.