Neural Networks for Optimal Selection of The PID Parameters and Designing Feedforward Controller

Mr. Ahmed S. Al-Araji & Miss May N. Bunny

Received on :21/12/2005
Accepted on :4/5/2006

Abstract

A neural network-based feedforward controller and self-tuning PID controller with optimization algorithm is presented. The scheme of the controller is based on two unknown models that describe the system and optimization algorithm. These models are modified Elman recurrent neural network and NARMA-L2. The modified Elman recurrent neural network (MERNN) model and NARMA-L2 model are learned with two stages off-line and on-line, in order to guarantee that the output of the model accurately represents the actual output of the system. The aim from the NARMA-L2 model is to find the Inverse Feedforward Controller (IFC) which controls the steady-state output of the system. The MERNN model after being learned is called the identifier. The feedback PID self tuning control signal for N-step ahead can be calculated the PID parameters by using the optimization algorithm with the quadratic performance index which is quadratic in the error between the desired set point and the model output, as well as quadratic of the control action. The paper explains the algorithm for a general case, and then a specific application on non-linear dynamical plant is presented.

الخلاصة

أن الشبكة العصبية أساس المسيطر التغذية الأمامية(Feedforward Neural Controller) و المسيطرPID ذات التنغيم التلقائي مع الخوارزمية المثالية قدمت في هذا البحث.
أن هيكلية المسيطر المستخدم تتألف من نموذجين غير معرفين يصفان المنظومة مع خوارزمية المثلى هذان النموذجين هما (Modified Elman Neural Network & NARMA-L2) .
أن نموذجين يتعلماً بمرحلتين (On-line & Off-line) لكي يضمن أن إخراج النموذج يمثل ألا خراج الحقيقي وبصورة دقيقة.أن الهدف من النموذج (NARMA-L2) هو أيجاد معكوس مسيطر التغذية الأمامية (Inverse Feedforward Controller) و الذي يتحكم بالاستجابة النهائية للمنظومة.
ويطلق على النموذج (Modified Elman Neural Network) الناتج بعد التعلم "المعرف" (Identifier) ومن المعرف و الخوارزمية المثلية يمكن حساب القيم المثلى للعناصر المسيطر PID ومن ثم يمكن حساب إشارة التغذية العكسية لعدد (ن) من الخطوات اللاحقه ولكل لحظه من اجل السيطرة على الاستجابة العابرة للمنظومة عن طريق تقليل معامل الأداء وهو مربع الفرق بين ألا خراج المرغوب واخراج النموذج إضافة إلى مربع إشارة السيطرة. وتم شرح هذه الخوارزمية واخذ مثال ذات تصرف لاخطي.

1- Introduction

The application of intelligent techniques to control systems has been a matter of wide study in recent years. These methods are used to solve complex problems that, in many cases, do not have an analytical solution. Neural networks (NNs), due to their ability to learn, have become a powerful tool in the development of the control systems. In fact nowadays, a new branch in control theory has arisen: Neuro control. This discipline studies the design of control systems aided by NN. Although in the process industry simple conventional controllers such as the PID have largely been extended, and show good performance for many tasks, when the plant or the process under control is complex or has high non-linearities.

The control performance degrades notably [1]. This section gives a general overview of using neural networks in control systems and describes briefly a number of applications in this field. 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 andParthasarathy [2,3], Levin and Narandra [4] are some of those that can be referred to as the application of neural networks for system identification. Also Noriega & Wang [5] for general unknown nonlinear systems present a neural-network-based direct adaptive control strategy in the paper where a simplified formulation of the control signals is obtained of a feedforward neural network and an optimization scheme. The reason of study by the researchers is motivated by simplicity to implement the PID control in the industrial environment, by easiness of utilization by engineers and process operators, and by acceptance in the industrial sector [6]. Some approaches proposed in the literature for deriving PID controllers are using self-tuning control techniques based on recursive parameter estimation, others are using automatic control techniques, and others are using intelligent control techniques. Despite the huge development in control theory, the majority of industrial processes are controlled by the well-established proportional-integral-derivative (PID) control. The popularity of PID control can be attributed to its simplicity and to its good performance in a wide range of operating conditions. In the last decade years, significant development has been established in the process control area to adjust the PID controller parameters automatically, in order to ensure adequate servo and regulatory behavior for a closed-loop plant [7,8,9]

The organization of the paper is as follows: Section two describes the use of feedforward neural networks to learn to act as input-output model. Two models (modified Elman recurrent and NARMA-L2) for system identification is examined with the corresponding neural nets and learning mechanism used for this purpose. Section three represents the core of the present paper, and it is suggested using a feedforward neural controller and a feedback self tuning PID controller with optimization algorithm that will attain specific benefits towards a systematic engineering design procedure for neural control system. Illustrative example, that clarify the features of the proposed strategy are given in section four, where an example is discussed in detail. Finally, section five contains the conclusions of the entire work.

2- Identification of Dynamical Systems Using Neural Network Modeling

This section focuses on nonlinear system identification using two models of multi-layered feedforward neural network, the first one is modified Elman recurrent model and the second is NARMA-L2 model. The neural network is trained using Dynamic Back-Propagation Algorithm. 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.

2-1 Recurrent Neural Networks

The Recurrent neural networks RNN structures are suitable to channel equalization and multi-user detection applications, since they are able to cope with channel transfer functions that exhibit deep spectral nulls, forming optimal decision boundaries and are less computationally demanding than MLP networks for these applications [10]. Among the available recurrent networks, modified Elman networks as shown in Fig (1) is one of the simplest types that can be trained using dynamic BP algorithm and it used to minimize the oscillation or even instabilities to the training controller. The output of the context unit in the modified Elman network is given by:

$$h_c^o(k) = \alpha h_c^o(k-1) + \beta h_c(k-1) \quad (1)$$

where $h_c^o(k)$ and $h_c(k)$ are respectively the output of the context unit and hidden unit and α is the feedback gain of the self-connections and β is the connection weight from the hidden units (c'th)to the context units (c'th) at the context layer. The value of α and β are selected randomly between (0 and 1). From the figure (1) it can be seen that the following equations:

$$h(k) = F\{V1U(k), V2h^o(k)\} \quad (2)$$

$$O(k) = Wh(k) \quad (3)$$

where V1,V2 and W are weight matrices and F is a non-linear vector function. The multi-layered modified Elman neural networks shown in figure (1) that is composed of many interconnected processing units called neurons or nodes. where:

$V1$: Weight matrix of the hidden layers.
$V2$: Weight matrix of the context layers.
W: Weight matrix of the output layer.
L : Denotes linear node.
H: Denotes nonlinear node with sigmoidal function.

To explain these calculations, consider the general j'th neuron in the hidden layer shown in figure (2). The inputs to this neuron consist of an ni – dimensional vector and (ni is the number of the input nodes). Each of the inputs has a weight V1 and V2 associated with it. The first calculation within the neuron consists of calculating the weighted sum net_j of the inputs as [11]:

$$net_j = \sum_{i=1}^{nh} V1_{j,i} \times U_i + \sum_{c=1}^{C} V2_{j,c} \times h_c^o \quad (4)$$

Where
j=c.
nh=C number of the hidden nodes and context nodes.

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

$$h_j = \mathrm{H}(net_j) \quad (5)$$

$$\mathrm{H}(net_j) = \frac{2}{1+e^{-net_j}} - 1 \quad (6)$$

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 (7)).

$$\text{neto}_k = \sum_{j=1}^{nh} W_{kj} \times h_j \qquad (7)$$

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

$$O_k = L(neto_k), \text{ Where L (x)=x} \qquad (8)$$

The learning (training) algorithm is usually based on the minimization (with respect to the network weights) of the following objective cost function as equation (9).

$$E = \frac{1}{2}\sum_{i=1}^{np}(e^i(k+1))^2 = \frac{1}{2}\sum_{i=1}^{np}(y_p^i(k+1) - yl_m^i(k+1))^2 \qquad (9)$$

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 yl_m^i is the model output of the plant of each step.

2-2 NARMA-L2 Model Identification

Narendra and Mukhopadhyay in their paper [12] proposed two approximation input-output models (referred to by Narendra as NARMA-L1 and NARMA-L2) derived from the NARMA model, in which the control input appears linearly. The NARMA-L2 model requires only two neural networks to approximate the function f and g.

$$y_p(k+1) = f[y_p(k),\ldots y_p(k-n+1),u(k-1),\ldots u(k-n+1)] + g[y_p(k),\ldots y_p(k-n+1),u(k-1),\ldots u(k-n+1)]\times u(k) \qquad (10)$$

The identification model of NARMA-L2 model can be better illustrated as Fig (3), where $\overline{X}$ is represents the input vector of the networks N1 and N2 (the argument of $\hat{f}[-]$ and $\hat{g}[-]$). The same cost function in equation (9), is used for the learning algorithm that is usually based on the minimizing (with respect to the network weights) of the objective function.

$$E = \frac{1}{2}\sum_{i=1}^{np}(e^i(k+1))^2 = \frac{1}{2}\sum_{i=1}^{np}(y_p^i(k+1) - y2_m^i(k+1))^2 \qquad (11)$$

From Fig (3), 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 $y2_m$ (k+1)) to the output of N2 after multiplying it by u(k) and to the output of N1 directly. Figure (4) illustrates the error back-propagation and one can think of u(k) as a weight at link2.

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 (modified Elman networks model and NARMA-L2). The first step in the procedure of the control structure is the identification of the plant from the input-output data, and then a feedforward neural controller is used as the inverse of the plant. Also a feedback PID self tuning controller is

used based on the minimization of a quadratic performance index function of the error between the desired input and the actual output plant and of the feedback PID controller itself. An optimization algorithm is used to determine the control signal for N-steps ahead which will minimize the cost function in order to achieve good tracking of the reference signal and to use minimum effort. The integrated control structure that consists of the inverse of the plant, feedback self-tuning PID controller with an optimization algorithm and the series model reference, thus brings together the advantages of the inverse method with the robustness of feedback. The general structure of the neural controller can be given in the form of the block diagram shown in Fig (5).
In the following sections, each part of the proposed controller will be explained in detail.

3-1 Feedforward Neural Controller (FFNC)

The feedforward neural controller is very important in the structure of the controller, because of its necessity to keep the steady-state tracking error to zero. This means that the action of the (FFNC) $u_{ff}(k)$ is to put the output of the plant as the reference input in steady state. Hence the (FFNC) is supposed to learn the inverse dynamic of the plant and so it is called inverse feedforward controller (IFC). To achieve this a neural using NARMA-L2 model equation (10) as explained in section two uses network for identification of the plant. 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 (12) below:

$$y_m(k+1) = \hat{f}[y_p(k),...,y_p(k-n+1),u_{ff}(k-1),...,u_{ff}(k-n+1)] + \hat{g}[y_p(k),...,y_p(k-n+1),u_{ff}(k-1),...,u_{ff}(k-n+1)] \times u_{ff}(k) \quad (12)$$

Likewise if $\hat{g}[-]$ is sign definite in the operating region then the network can be used as the inverse of the plant as given by equation (13).

$$u_{ff}(k) = \frac{y_{ref}(k+1) - \hat{f}[y_p(k),...y_p(k-n+1),u_{ff}(k-1),..u_{ff}(k-n+1)]}{\hat{g}[y_p(k),...y_p(k-n+1),u_{ff}(k-1),...u_{ff}(k-n+1)]} \quad (13)$$

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 [12]. Now by using equation (12) as the model of the plant identifier and equation (13) as the inverse mapping of the model, then these form the feedforward neural controller. The training of the inverse dynamic is done off-line and on-line. After the neural network has learned the inverse dynamic then $u_{ff}(k)$ is the control action required to keep the output of the plant at the reference value at steady state, hence it will be called equivalently as u_{ref}.

3-2 Feedback Self-Tuning Controller (PID)

The feedback self-tuning PID controller is also 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. The feedback PID controller consists of an on-line neural identifier and an optimization algorithm. The goal is to find the feedback control action that minimizes the cumulative error between the reference input and the output of the plant as well as a weighted sum of the control signal. This can be achieved by minimizing the following quadratic performance index [13].

$$J=\frac{1}{2}\sum_{k=1}^{N} Q(y_{ref}(k+1)-y_p(k+1))^2+R(u_{ref}(k)-u(k))^2 \quad (14)$$

$u_{ref}(k)$ is the reference control action and it is equivalent to $u_{ff}(k)$.

$u(k)$ is the total control signal $=u_{ff}(k)+u_{fb}(k)$

$u_{fb}(k)$ is the feedback control action.

$y_{ref}(k+1)$ is the reference input.

(Q, R) are positive weighting factors

N is number of steps ahead

Hence:

$$u_{ref}(k)= u_{ff}(k) \quad (15)$$

$$u(k) = u_{ff}(k)+u_{fb}(k) \quad (16)$$

Substituting equation (15) and (16) in (14) then J will be given:

$$J=\frac{1}{2}\sum_{k=1}^{N} Q(y_{ref}(k+1)-y_p(k+1))^2+R(u_{ff}(k)-(u_{ff}(k)+u_{fb}(k)))^2 \quad (17)$$

$$J=\frac{1}{2}\sum_{k=1}^{N} Q(y_{ref}(k+1)-y_p(k+1))^2+R(u_{fb}(k))^2 \quad (18)$$

This quadratic cost function will not only force the output to follow the reference input by minimizing the cumulative error N steps ahead but also forces the control action in the transient period to be as close as possible to the reference control signal. Also J depends on (Q and R) which are positive weighting factors. Hence the control action found will be optimal with respect to the given set of values of the weighting factors Q and R [13 &14].

The on-line identifier of the plant is to be used to obtain the predicted values of the output of the plant N steps ahead instead of running the plant itself N steps. These values are needed to calculate the feedback PID control action from the parameters Kp, Ki and Kd by the optimization algorithm such that the quadratic performance index J will be minimized. Also on line identification is required to make $y1_m(k)$ the output of the identifier as close as possible to the plant output $y_p(k)$. A feedforward neural network will be used as an identifier and two stages of learning of this neural network will be performed. The first stage is an off-line identification and the second stage is an on-line modification of the weights of the obtained identifier to keep track of any possible variation of the plant parameters. Therefore it can be said $y1_m(k)\approx y_p(k)$, and the performance index of equation (18) can be put as:

$$J=\frac{1}{2}\sum_{k=1}^{N} Q(y_{ref}(k+1)-y1_m(k+1))^2+R(u_{fb}(k))^2 \quad (19)$$

$$J=\frac{1}{2}\sum_{k=1}^{N} Q(e(k+1))^2+R(u_{fb}(k))^2 \quad (20)$$

$$e(k+1)=y_{ref}(k+1)-y1_m(k+1) \quad (21)$$

In this work a one hidden layer feedforward neural network is used for the identifier, hence

$$y1_m(k+1)=L(\sum_{j=1}^{nh} W_j h_j + W_{nh+1}\times bias)=L(neto) \quad (22)$$

The single linear neuron, then, passes the sum (neto) through a linear function of slope (1). Where the activation function of the hidden layer is a sigmoidal function and the output layer is a linear function [15]. Dynamic back propagation algorithm (BPA) is used to adjust the weights of the MERNN to learn the dynamics of the plant, and a simple gradient decent rule is used. After the identifier learns the dynamics of the system then the whole structure of the controller as shown in Fig (5) will be implemented.

3-3 The Series Model Reference

The model reference is used to overcome the harmonics of the step change in the set point desired and to

reduce the spikes of the control action of the feedforward neural controller [13 and 16]. Therefore the transient time of the plant is reduced and the overshoot is decreased. The model reference in the structure of the controller may be chosen as the difference equation (23):

$$y_{ref}(k+1) = (1-\sigma) y_{des}(k+1) + \sigma \times y_{ref}(k) \tag{23}$$

And σ is the tuning parameter. To ensure that the model reference is stable and to avoid ringing, the tuning parameter should be chosen as $0 \le \sigma < 1$ [16 and 17].

Algorithm description of one step ahead control action

In this section, the feedback PID control signal $u_{fb}(k+1)$ will be derived for one-step ahead depended on the parameters of the PID controller, that is when N=1.

Where:

$$\begin{aligned} u_{fb}(k+1) = u_{fb}(k) &+ Kp(k+1)[e(k+1) - e(k)] \\ &+ Ki(k+1)\, e(k+1) \\ &+ Kd(k+1)[e(k+1) - 2e(k) + e(k-1)] \end{aligned} \tag{24}$$

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

$$Kp(k+1) = Kp(k) + \Delta Kp(k) \tag{25}$$

$$Ki(k+1) = Ki(k) + \Delta Ki(k) \tag{26}$$

$$Kd(k+1) = Kd(k) + \Delta Kd(k) \tag{27}$$

$$\Delta Kp(k) = -\eta \frac{\partial J}{\partial Kp(k)} \tag{28}$$

But:

$$-\eta \frac{\partial J}{\partial Kp(k)} = -\eta \left[\frac{\frac{1}{2} Q \partial (e(k+1))^2}{\partial Kp(k)} + \frac{\frac{1}{2} R \partial (u_{fb}(k))^2}{\partial Kp(k)} \right] \tag{29}$$

Where:

$$e(k+1) = y_{ref}(k+1) - y1_m(k+1) \tag{30}$$

By the chain rule of differentiation we have

$$\frac{\partial (e(k+1))^2}{\partial Kp(k)} = \frac{\partial (e(k+1))^2}{\partial y1_m(k+1)} \times \frac{\partial y1_m(k+1)}{\partial Kp(k)}$$

$$= -2e(k+1) \times \frac{\partial y_m(k+1)}{\partial Kp(k)} \tag{31}$$

And

$$\frac{\partial (u_{fb}(k))^2}{\partial Kp(k)} = e(k+1) - e(k) \tag{32}$$

Hence equation (29) becomes

$$-\eta \frac{\partial J}{\partial Kp(k)} = -\eta [Qe(k+1) \frac{\partial y1_m(k+1)}{\partial Kp(k)} + R(e(k+1) - e(k))] \tag{33}$$

For the two-layer modified Elman on-line neural network identifier shown in Fig (1) we have:

$$\frac{\partial y1_m(k+1)}{\partial Kp(k)} = \frac{\partial L(neto)}{\partial neto} \times \frac{\partial neto}{\partial Kp(k)} \tag{34}$$

For linear activation function output

$$\frac{\partial L(neto)}{\partial neto} = 1$$

$$\frac{\partial y1_m(k+1)}{\partial Kp(k)} = 1 \times \frac{\partial neto}{\partial h_j} \times \frac{\partial h_j}{\partial Kp(k)} \tag{35}$$

$$\frac{\partial y1_m(k+1)}{\partial Kp(k)} = 1 \times \sum_{j=1}^{nh} W_j \frac{\partial h_j}{\partial net_j} \times \frac{\partial net_j}{\partial Kp(k)} \tag{36}$$

Where $\frac{\partial h_j}{\partial net_j} = \frac{1}{2}\left[1 - (h_j)^2\right] = H'(net_j)$

$$\frac{\partial y1_m(k+1)}{\partial Kp(k)} = 1 \times \sum_{j=1}^{nh} W_j \times H'(net_j) \times \frac{\partial net_j}{\partial u(k)} \times \frac{\partial u(k)}{\partial Kp(k)} \quad (37)$$

$$\frac{\partial y1_m(k+1)}{\partial Kp(k)} = 1 \times \sum_{j=1}^{nh} W_j \times H'(net_j) \times V_{ji} \times \frac{\partial (u_{ff}(k) + u_{fb}(k))}{\partial Kp(k)} \quad (38)$$

$$\frac{\partial y1_m(k+1)}{\partial Kp(k)} = 1 \times \sum_{j=1}^{nh} W_j \times H'(net_j) \times V_{ji} \times [e(k+1) - e(k)] \quad (39)$$

Where the V_{ji}'s are the weights of the u(k) only [17].
Hence: -

$$\Delta Kp(k) = [\eta Qe(k+1) \sum_{j=1}^{nh} (W_j \times H'(net_j) \times V_{ji} \times (e(k+1) - e(k)] - [\eta \times R \times (e(k+1) - e(k)] \quad (40)$$

$$\Delta Ki(k) = [\eta Qe(k+1) \sum_{j=1}^{nh} (W_j \times H'(net_j) \times V_{ji} \times (e(k+1)] - [\eta \times R \times (e(k+1)] \quad (41)$$

$$\Delta Kd(k) = [\eta Qe(k+1) \sum_{j=1}^{nh} (W_j \times H'(net_j) \times V_{ji} \times (e(k+1) - 2e(k) + e(k-1)] - [\eta \times R \times (e(k+1) - 2e(k) + e(k-1)] \quad (42)$$

For N steps-ahead algorithm of the feedback PID control action can be descripted in appendix.

4- Case Study

In this section, an example is taken to clarify the features of the neural controller explained in section three and applied the algorithm for 0ne-step ahead and five-steps ahead.
The plant to be controlled is described by the difference equation:

$$y_p(k+1) = \frac{-0.9 y_p(k) + u(k)}{1 + y_p^2(k)} \quad (43)$$

This plant has been adopted from [13 and 18]. For the open loop response of the plant $y_p(k)$ to the input signal u(k) is shown in Fig (6-a and b) respectively. The plant response is very oscillatory for the low amplitude input and shows limit cycle oscillation for $|u(k)| > 0.6$ [13 and 18]. To use the proposed controller first a neural network is trained for the feedforward controller, then the feedback controller is established.

The Feedforward Controller

To identify the plant dynamics, series-parallel identification structure as that in Fig (3). The model is described by:

$$y2_m(k+1) = N1[y_p(k)] + N2[y_p(k)] \times u(k) \quad (44)$$

Where N1[-] and N2[-] are multi-layered neural networks which approximate $\hat{f}[-]$ and $\hat{g}[-]$ of the equation (10) respectively. Since each of N1[-] and N2[-] has one inputs (see equation (44)), the initial guess of the number of hidden nodes was three for each network. An input-output training pattern is needed to provide enough information about the plant to be modeled. This can be achieved by injecting a sufficiently rich input signal to excite all process modes of interest while also ensuring that the training patterns adequately covers the specified operating region. A hybrid excitation signal has been used for the plant that the input signal consisted of random amplitude signal with range (-1 to +1). After identification series-parallel configuration for many times a neural network with three hidden nodes gives fairly good generalization capabilities as shown in Fig (7-a and b). When the training is continued up to 4000 epochs ASE equals 3.1×10^{-6}. The plant Jacobian N2[-] $= \hat{g}[-]$ is sign definite in the region of interest. And then applied parallel configuration identification with initial the same finishing weights in a

neural network with three hidden nodes in the series-parallel configuration gives fairly very good generalization capabilities when the training is continued up to 3000 epochs ASE equals 1.1×10^{-6}. An on-line updating of the weights of the neural network will be carried out to ensure the output of the model will be equal to that of the plant, so that calculation of u_{ff} will be fairly accurate. This means that the plant is invertable and a controller of the form of equation (45) can be implemented.

$$u_{ff}(k)=\frac{y_{ref}(k+1)-N1[y2_m(k)]}{N2[y2_m(k)]} \quad (45)$$

Where $y_{ref}(k+1)$ is the output of the model reference.

Feedback Self-Tuning PID Controller

For off-line identification with series-parallel configuration a model described by modified Elman neural networks as shown in Fig (1). BPA with learning rate $\eta=0.2$ for N[-] and the input-output patterns as a learning set, then after 5000 epochs the ASE is equal to 3.5×10^{-6}. Figure (8-a and b) compares the time response of the model with the actual plant output for the u(k) as learning set and testing set respectively. And then applied parallel configuration identification with initial the same finishing weights in a neural network series-parallel configuration gives fairly very good generalization capabilities when the training is continued up to 4000 epochs ASE equals 1.1×10^{-6}. An on-line updating of the weights of the neural network will be carried out to ensure the output of the model will be equal to that of the plant, so that calculation of u_{fb} by the optimization algorithm will be fairly accurate.

Simulation Results

In this simulation, the proposed control scheme is applied to the plant model.

For one step ahead N=1

The equation of the model reference is taken from [13 and 17] is:

$$y_{ref}(k+1)=0.3y_{ref}(k)+0.7y_{des}(k+1) \quad (46)$$

When the tuning parameter of the model reference is equal to (0.3). The response of the plant model is fast without overshoot and steady-state error is zero as shown in Fig (9). The response of the NARMA-L2 model and modified Elman model as shown in Fig (10). The feedforward control action is reach to 0.6 amplitude valve as shown in Fig (11) that without the output plant model became oscillatory. The feedback PID control action has small value with respect to the feedforward control action but it has small spikes when the output desired is step change as shown in Fig (12). The total control action as shown in Fig (13). Figures (14-a, b, and c) the values of the PID controller parameters Kp, Ki, and Kd respectively. That calculated from the optimization algorithm for one-step ahead and there are depended on the parameter of the quadratic performance index (Q=1 and R=1) and the error between the reference output and the modified Elman model output. To study the effect of the parameters (Q and R) on the calculation the PID parameters Kp, Ki, and Kd as equations (40, 41, and 42). Then find the response of the feedback PID control action. The parameter of the quadratic performance index (Q=1 and R=0) the response of the plant model is fast with small overshoot for ten samples. And steady-state error is zero as shown in Fig (15). The feedback PID control action has small value but it has small spikes as

shown in Fig (16). The total control action as shown in Fig (17). The PID controller parameters Kp, Ki, and Kd as shown in Figures (18-a, b, c). When the parameter of the quadratic performance index (Q=0 and R=1) the response of the plant model is fast with small overshoot for fourteen samples. And has steady-state error as shown in Fig (19). The feedback PID control action has small negative value with very small spikes as shown in Fig (20). The total control action as shown in Fig (21).The PID parameters Kp, Ki, and Kd as shown in Figures (22-a, b, c).

For Five step ahead N=5

The response of the plant model is fast without overshoot and steady-state error is zero when the optimization algorithm is generate five steps ahead with parameter of the quadratic performance index (Q=1 and R=1) as shown in Fig (23). The feedforward control action as shown in Fig (24). The feedback PID control action as shown in Fig (25) with very small spikes. The PID controller parameters Kp, Ki, and Kd as shown in Fig (26-a, b, c). After each sampling time the weights of the on-line identifier "modified Elman Neural networks" and NARMA-L2 model are updated in order to minimize the error between y_p, $y1_m$ and $y2_m$ by using Back Propagation Algorithm.

5- Conclusion

The structure of the neural controller with an optimization algorithm based on modified Elman and NARMA-L2 neural models as well as the calculation of feedforward and feedback PID control action has been designed and successfully simulated to the nonlinear system. The calculation of feedback PID control action for N steps ahead is based on the minimization of a quadratic performance index of the error between the desired output and the model output (modified Elman neural network) as well as the control action. The on-line identifier modified Elman neural neural network model of the plant is used to generate the parameters of the PID controller (Kp, Ki, and Kd) control action with on-line updating of the weights of the identifier by using (BPA) in order to guarantee that model output approaches the actual output. Using neural NARMA-L2 model as a nonlinear model of the plant provides a simple check on the model invertability, which appears to be of critical importance as it is used for the inverse feedforward controller. On-line updating of the weights of the NARMA-L2 model using dynamic (BPA) guarantees that model output approaches the actual output.

Appendix

N Steps Ahead Optimization Algorithm

The N steps estimation of u_{fb} will be calculated for each sample. Since the modified Elman neural network model as given by equations (2 and 3) represents the plant to be controlled asymptotically, it can be used to predict future values of the model output for the next N steps, and can be used to find the optimal value of u_{fb} using an optimization algorithm that calculated the parameters of the PID feedback controller Kp, Ki, and Kd. For this purpose, let N be a pre-specified positive integer and denote:

$$Y_{ref\,t,N} = [y_{ref}(t+1), y_{ref}(t+2), ..., y_{ref}(t+N] \quad (46)$$

as the future values of set point and (t) represents the time instant, and

$$Y1_{m\,t,N} = [y1_m(t+1), y1_m(t+2), ..., y1_m(t+N] \quad (47)$$

as the predicted outputs of the model of the plant using the modified Elman neural network model. Then define the following error vector:

$$E_{t,N} = [e(t+1), e(t+2), ..., e(t+N]$$ (48)

where:

$$e(t+i) = y_{ref}(t+i) - y1_m(t+i))$$
$$i = 1,2,...,N$$ (49)

Defining the feedback control signals to be determined as:

$$U'_{fb\,t,N} = [u'_{fb}(t), u'_{fb}(t+1), ..., u'_{fb}(t+N-1)]$$ (50)

$$Kp'_{fb\,t,N} = [Kp'(t), Kp'(t+1), ..., Kp'(t+N-1)]$$ (51)

$$Ki'_{fb\,t,N} = [Ki'(t), K'i(t+1), ..., Ki'(t+N-1)]$$ (52)

$$Kd_{fb\,t,N} = [Kd'(t), Kd'(t+1), ..., Kd'(t+N-1)]$$ (53)

And assuming the following objective function:

$$J1 = \frac{1}{2} QE_{t,N} E_{t,N}{}^T + \frac{1}{2} RU'_{fb\,t,N} U'_{fb\,t,N}{}^T$$ (54)

Then our purpose is to find Kp', Ki', Kd' such that J1 is minimized using the gradient descent rule, so that the new parameters of the PID control action will be given by:

$$Kp'_{t,N}{}^{K+1} = Kp'_{t,N}{}^{K} + \Delta Kp'_{t,N}{}^{K}$$ (55)

$$Ki'_{t,N}{}^{K+1} = Ki'_{t,N}{}^{K} + \Delta Ki'_{t,N}{}^{K}$$ (56)

$$Kd'_{t,N}{}^{K+1} = Kd'_{t,N}{}^{K} + \Delta Kd'_{t,N}{}^{K}$$ (57)

where k here indicates that calculations are done at the kth sample; and

$$\Delta Kp'_{t,N}{}^{K} = -\eta \frac{\partial J1}{\partial Kp'_{t,N}{}^{K}} = [\Delta Kp'(t), \Delta Kp'(t+1), .. \Delta Kp'(t+N-1)]$$ (58)

$$\Delta Ki'_{t,N}{}^{K} = -\eta \frac{\partial J1}{\partial Ki'_{t,N}{}^{K}} = [\Delta Ki'(t), \Delta K'i(t+1), .. \Delta Ki'(t+N-1)]$$ (59)

$$\Delta Kd'_{t,N}{}^{K} = -\eta \frac{\partial J1}{\partial Kd'_{t,N}{}^{K}} = [\Delta Kd'(t), \Delta Kd'(t+1), .. \Delta Kd'(t+N-1)]$$ (60)

$$-\eta \frac{\partial J1}{\partial Kp'_{t,N}{}^{K}} = \eta QE_{t,N} \frac{\partial Y1_{m\,t,N}}{\partial Kp'_{t,N}{}^{K}} - \eta R \frac{\partial U_{fb\,t,N}}{\partial Kp'_{t,N}{}^{K}}$$ (61)

Where:

$$\frac{\partial Y1_{m\,t,N}}{\partial kp'_{t,N}{}^{K}} = \begin{bmatrix} \frac{\partial y1_m(t+1)}{\partial kp'(t)} & \frac{\partial y1_m(t+2)}{\partial kp'(t)} & \frac{\partial y1_m(t+3)}{\partial kp'(t)} & \cdots & \frac{\partial y1_m(t+N)}{\partial kp'(t)} \\ 0 & \frac{\partial y1_m(t+2)}{\partial kp'(t+1)} & \frac{\partial y1_m(t+3)}{\partial kp'(t+1)} & \cdots & \frac{\partial y1_m(t+N)}{\partial kp'(t+1)} \\ 0 & 0 & \frac{\partial y1_m(t+3)}{\partial kp'(t+2)} & \cdots & \frac{\partial y1_m(t+N)}{\partial kp'(t+2)} \\ : & : & : & : & : \\ 0 & 0 & 0 & 0 & \frac{\partial y1_m(t+N)}{\partial kp'(t+N-1)} \end{bmatrix}$$ (62)

$$\frac{\partial U_{fb\,t,N}}{\partial Kp'_{t,N}{}^{K}} = = \begin{bmatrix} \frac{\partial u_{fb}(t+1)}{\partial kp'(t)} & \frac{\partial u_{fb}(t+2)}{\partial kp'(t)} & \frac{\partial u_{fb}(t+3)}{\partial kp'(t)} & \cdots & \frac{\partial u_{fb}(t+N)}{\partial kp'(t)} \\ 0 & \frac{\partial u_{fb}(t+2)}{\partial kp'(t+1)} & \frac{\partial u_{fb}(t+3)}{\partial kp'(t+1)} & \cdots & \frac{\partial u_{fb}(t+N)}{\partial kp'(t+1)} \\ 0 & 0 & \frac{\partial u_{fb}(t+3)}{\partial kp'(t+2)} & \cdots & \frac{\partial u_{fb}(t+N)}{\partial kp'(t+2)} \\ : & : & : & : & : \\ 0 & 0 & 0 & 0 & \frac{\partial u_{fb}(t+N)}{\partial kp'(t+N-1)} \end{bmatrix}$$ (63)

It can be seen that each element in the above matrix can be found by differentiating (2 and 3) with respect to each element in (51) as a result, it can be obtained that:

$$\frac{\partial y1_m(t+n)}{\partial Kp'(t+j-1)} = \frac{\partial \hat{F}(p)}{\partial Kp'(t+j-1)} + \sum_{i=m}^{n-1} \frac{\partial \hat{F}(p)}{\partial y1_m(t+i)} \times \left[\frac{\partial y1_m(t+i)}{\partial Kp'(t+j-1)} \right]$$
$$n = 1,2,3,...,N$$
$$j = 1,2,3,..N$$ (64)

where P is the input pattern

P=[u(t),u(t-1),,u(t-n+1), y1m(t), y1m(t1), …,y1m(t-n+1)]

To calculate the $-\eta \frac{\partial J1}{\partial Ki'^{K}_{t,N}}$ and $-\eta \frac{\partial J1}{\partial Kd'^{K}_{t,N}}$

as the same equations (61, 62, 63 and 64). Equation (62) must be calculated using equation (64) every time a new control signal has to be determined. This could result in a large computation for a large N.

Therefore a recursive method for calculating the gain matrix is developed in the following, so that the algorithm can be applied to real-time systems. After completing the procedure from n=1 to N and from j=1 to N the new control action for the next sample will be

$$u'^{k}_{fb}(t+N) = u'^{k}_{fb}(t+N-1) + Kp'^{k}_{t,N}(e^{k+1}_{t,N} - e^{k}_{t,N}) + Ki'^{k}_{t,N}\, e^{k}_{t,N} + Kd'^{k}_{t,N}(e^{k+1}_{t,N} - 2e^{k}_{t,N} + e^{k-1}_{t,N}) \quad (65)$$

$$u(k+1) = u_{ff}(k+1) + u'^{K}_{fb}(t+N) \quad (66)$$

Where $u'^{k}_{fb}(t+N)$ is the last value of the feedback PID controlling signal calculated by the optimization algorithm, that is N-step ahead of control signal is calculated. This is calculated at each sample time k so that u(k+1) is applied to that plant and the model at the next sampling time. Then we continue to apply this procedure at the next sampling time (k+1) until the error between the desired input and the actual output becomes lower than a specified value.

Reference

[1] J. A. Mendez and L. Acosta"An application of a neural self-tuning controller to an over head crane", Neural comput and Application. Vol. 8, pp. 143-150. 1999.

[2] 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.

[3] 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.

[4] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks__ Part II: Obsrvability, Identification, and Control," IEEE Trans. Neural Networks vol. 7, no. 1, pp. 30-42, 1996.

[5] J. R. Noiega and H. Wang, "A direct adaptive neural-network control for unknown nonlinear systems and its application," IEEE Trans. Neural Networks, vol. 9, no. 1,pp. 27-34,1998.

[6] L.S.Co, O.M.Al."Design and Tuning of Intelligent and Self-Tuning PID Controller". Control Engineering International. University of Ceara. 2000.

[7] K.J. Astrom and B. Wittenmark, "Adaptive Control" Addison-Wesley Publishing Company, 1989.

[8] T.L. Chia," Some basic approaches for self-tuning controllers" Control Engineering International, pp. 49-52, Dec. 1992.

[9] D.P. Kwok, P.Tam, C.K. Li. And P. Wang,"Linguistic PID controller" in Proc. of 11th IFAC world Congress, Tallinn, Estonia, USSR, 1990,vol.7, pp.192-197.

[10] S. Haykin, "Neural Networks:A Comprehensive Foundation" 2nd Edition, Prentice-Hall, 1999.

[11] May N. Bunny "Development of controller algorithms for a dual-spin satellite" M.Sc. Thesis, University of technology, November 2002.

[12] 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.
[13] Ahmed S. Al-Araji "A Neural controller with a pre-assigned performance index" M.Sc. Thesis, University of technology, November 2000.
[14] Y. M. Park, M. S. choi, and K. Y. Lee, "An optimal tracking neuro-controller for nonlinear dynamic systems," IEEE Trans. Neural Networks, vol. 7, no. 5, pp. 1099-1109, 1996.
[15] J. M. Zurada, Introduction to Artificial Neural Systems. Jaico Publishing House, 1993.
[16] E. P. Nahas, M. A. Henson and D. E. Seborg, "Nonlinear internal model control strategy for neural network models," Computers Chem. Eng., vol. 16, no. 12, pp. 1039- 1057, 1992.
[17] Ahmed S. Al-Araji "Adaptive Neuro-Controller Based Model-Reference" engineering and technology journal, University of technology, November 2004.
[18] M. S. Ahmed and I. A.Tasadduq,"Neural-net controller for nonlinear plants: design approach through linearisation" IEEE Proc. Control Theory Appl., vol. 141 no. 5, pp. 315 322, 1994.

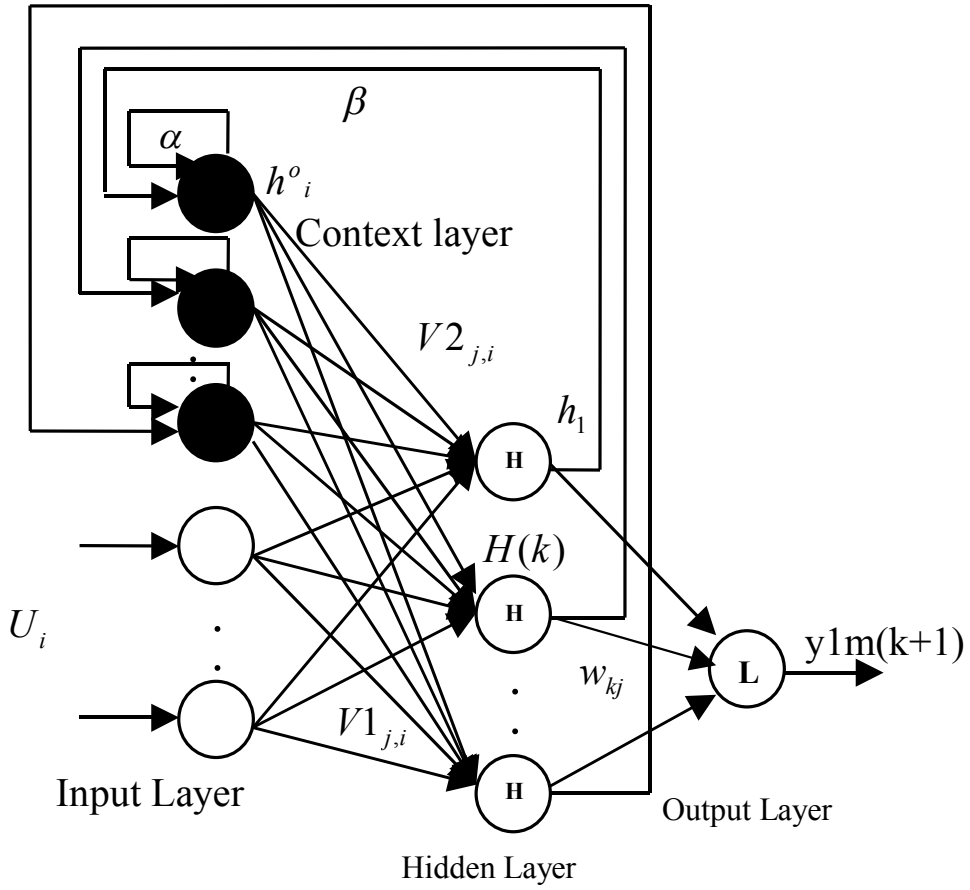


Fig (1): The Modified Elman Recurrent Neural Networks

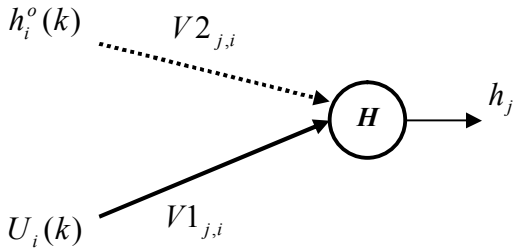
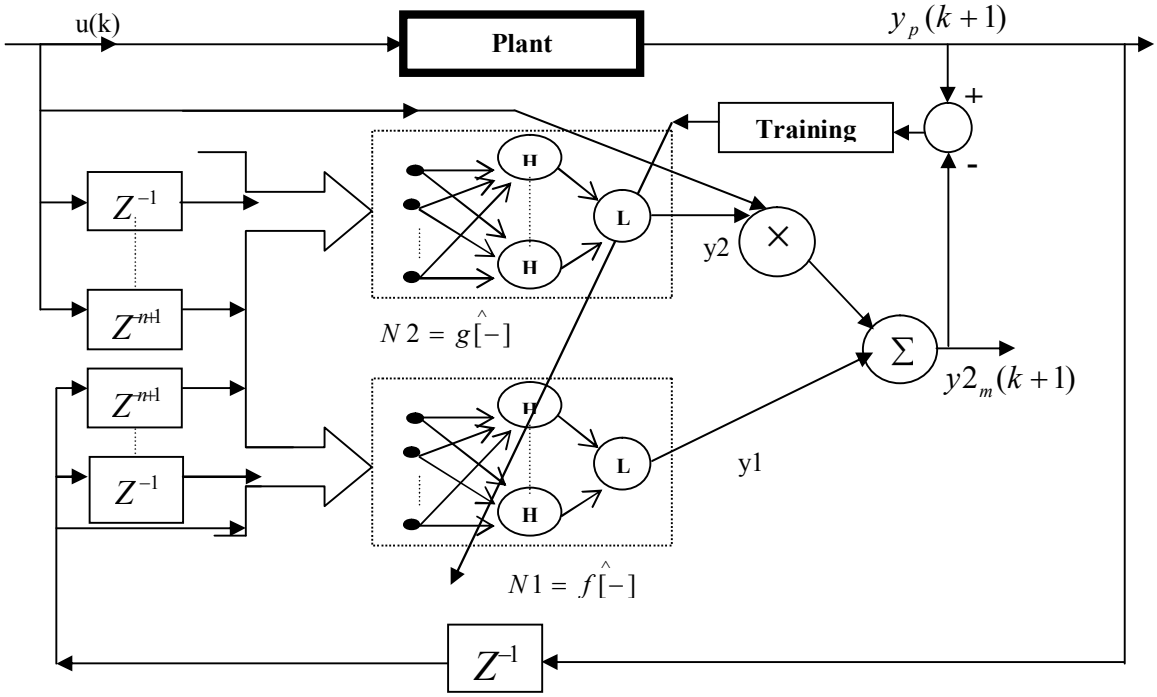


Fig (2): Neuron j in the hidden layer.



$\overline{X}$ **=[u(k-1),…,u(k-n+1),y** $_p$ **(k),…,y** $_p$ **(k-n+1)]**

Fig (3): NARMA-L2 identification model

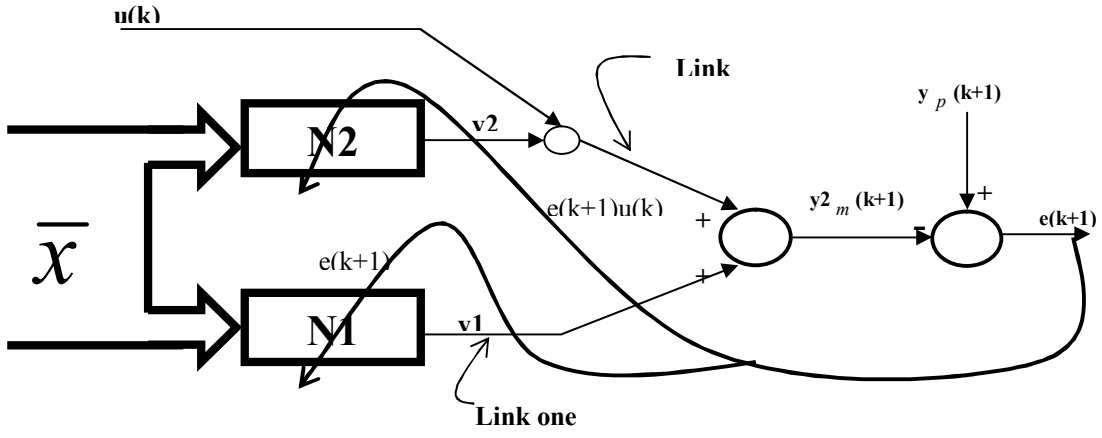
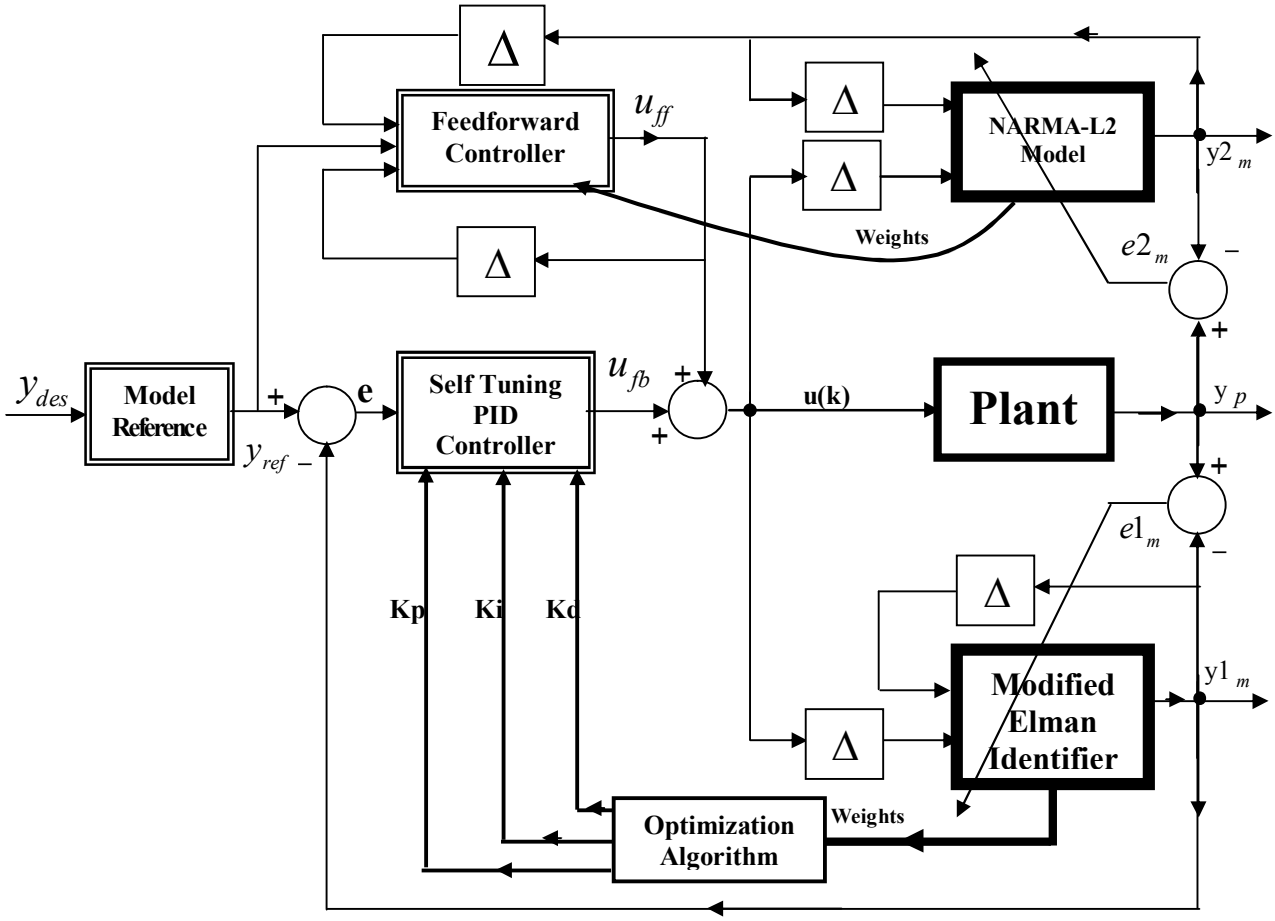


Fig (4): Block diagram illustrating backward error flow



Δ is defined as a delay mapping from a sequence of scalar inputs and outputs.

Fig (5): General structure of neural controller

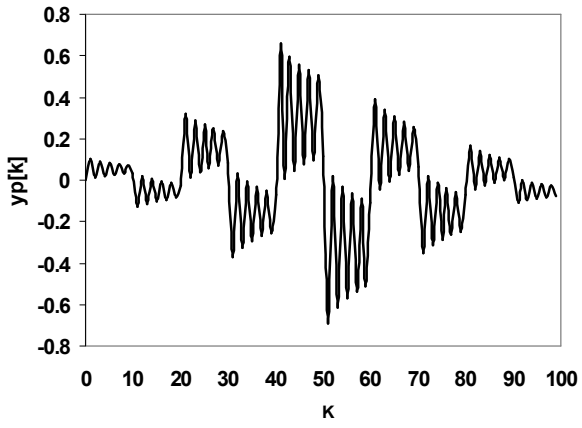


Fig (6-a): The open loop plant response

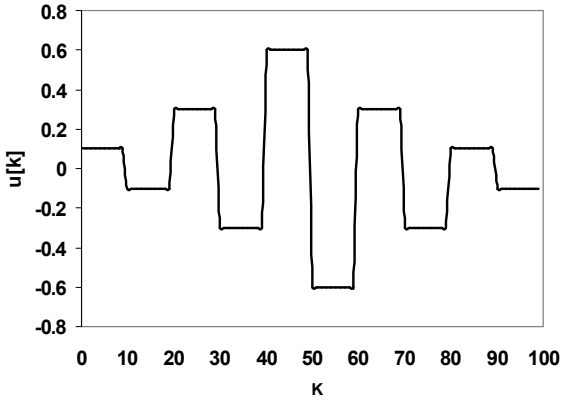


Fig (6-b): The corresponding input signal

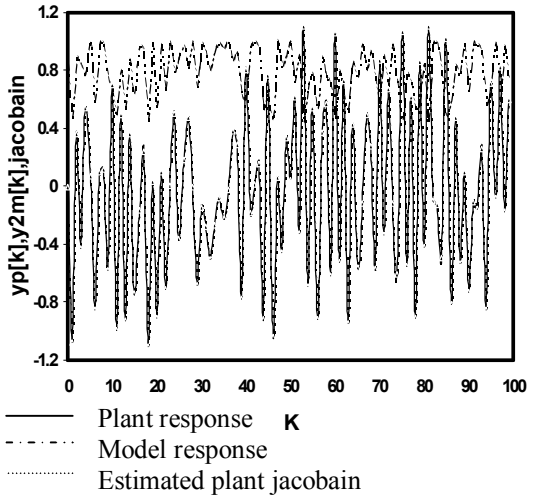


Fig (7-a): The response of the plant and of the series-parallel NARMA-L2 identification model for learning patterns and the estimated plant Jacobain

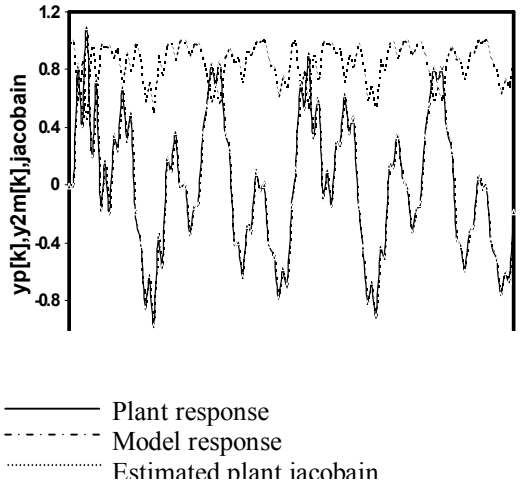


Fig (7-b): The response of the plant and of the series-parallel NARMA-L2 identification model for testing patterns and the estimated plant Jacobain

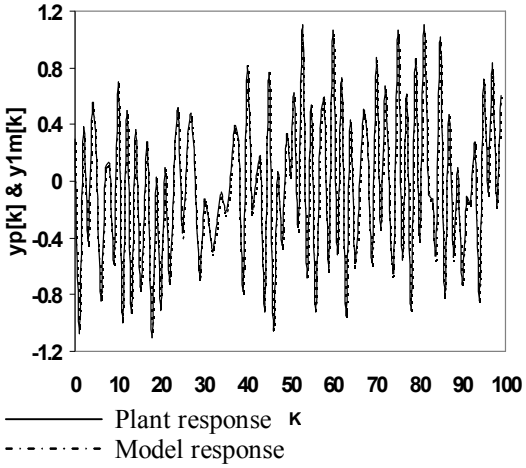


Fig (8-a): The response of the plant and of the series-parallel modified Elman identification model for learning patterns

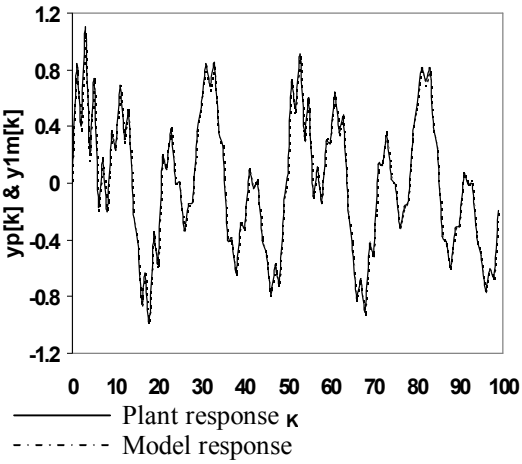


Fig (8-b): The response of the plant and of the series-parallel modified Elman identification model for testing patterns

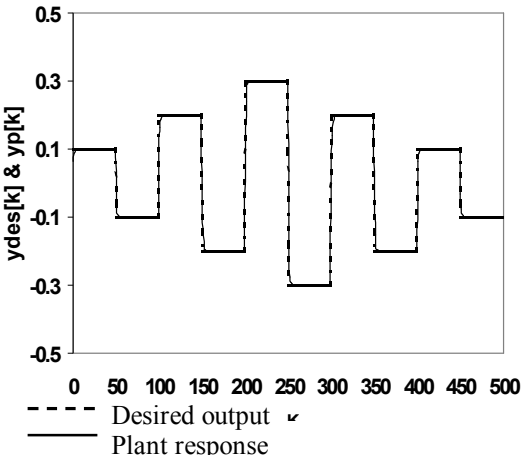


Fig (9): Desired output tracking for one-step ahead with Q=1 & R=1

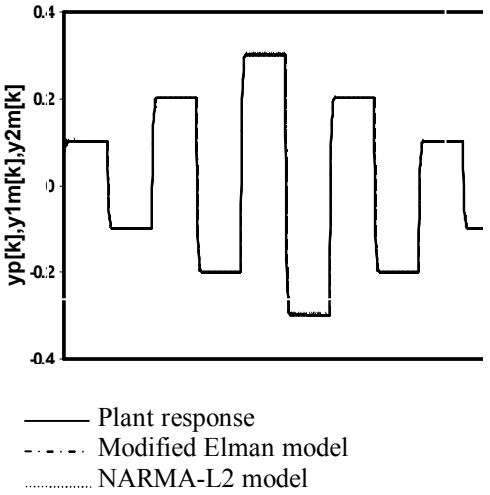


Fig (10): The response of the plant, Modified Elman model, & NARMA-L2 model

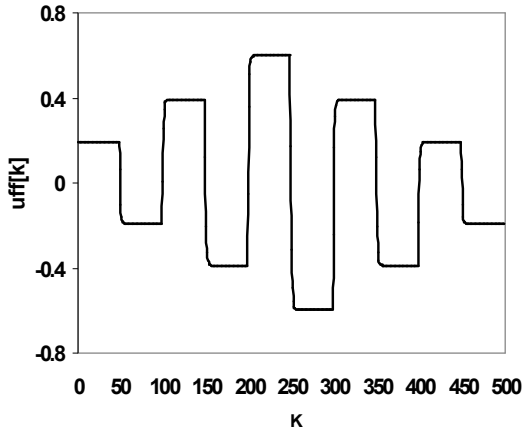


Fig (11): The feedforward control signal for Q=1 & R=1

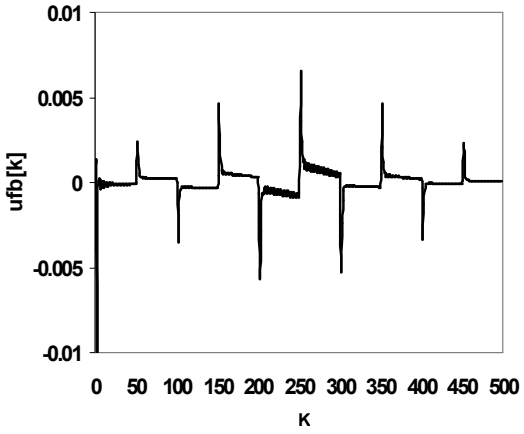


Fig (12): The feedback control signal for Q=1 & R=1

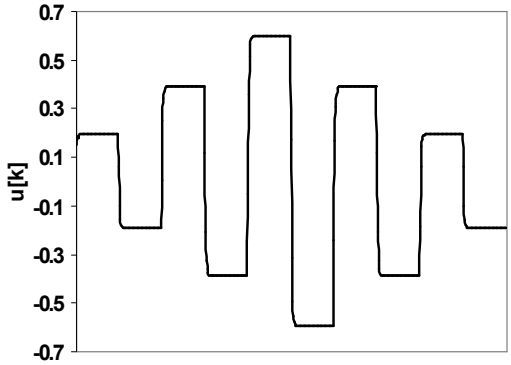


Fig (13): The total control signal for Q=1 & R=1

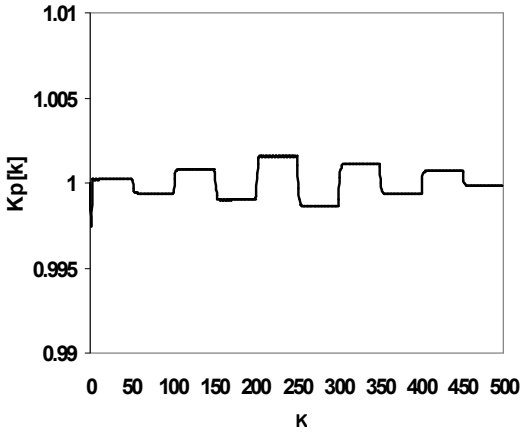


Fig (14-a): Kp gain of the PID Controller for Q=1 & R=1

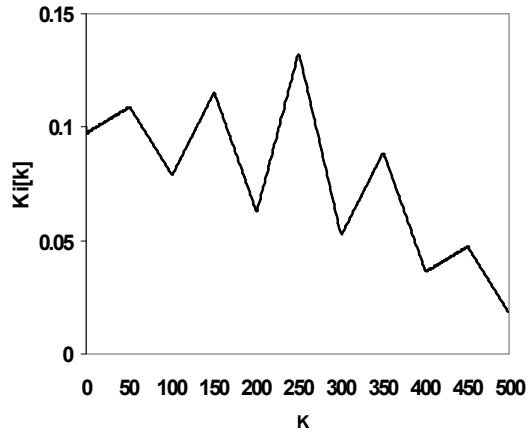


Fig (14-b): Ki gain of the PID Controller for Q=1 & R=1

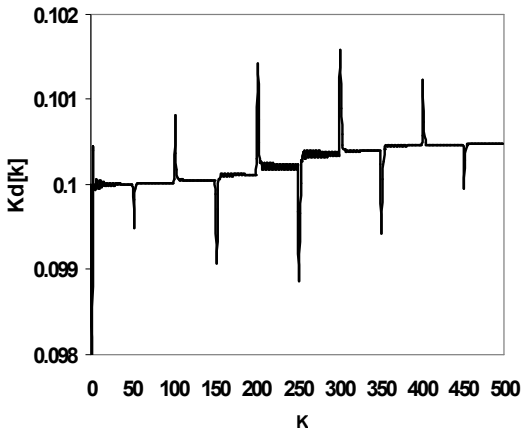


Fig (14-c): Kd gain of the PID Controller for Q=1 & R=1

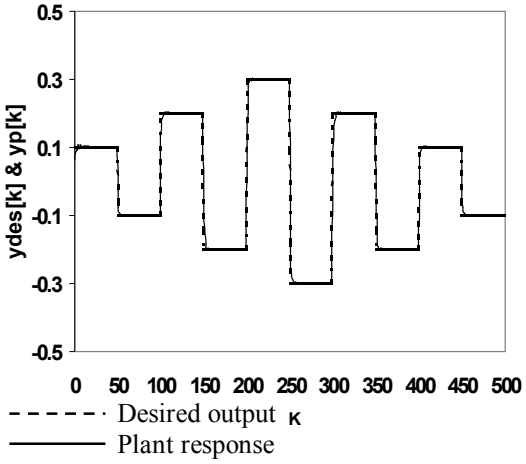


Fig (15): Desired output tracking for one-step ahead with Q=1 & R=0

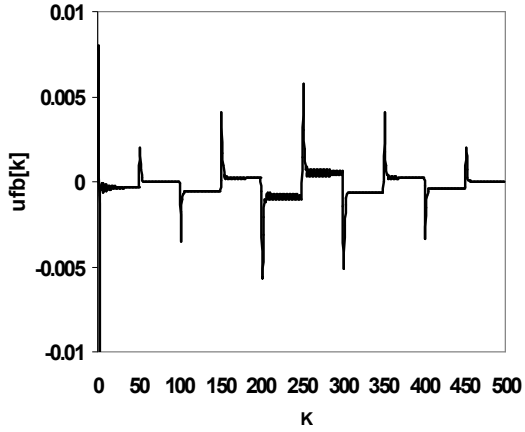


Fig (16): The feedback control signal for Q=1 & R=0

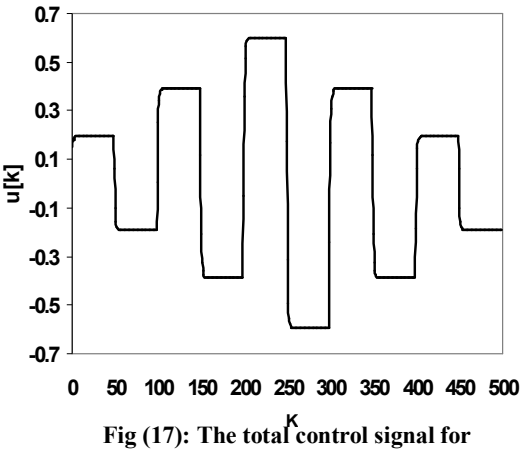


Fig (17): The total control signal for Q=1 & R=0

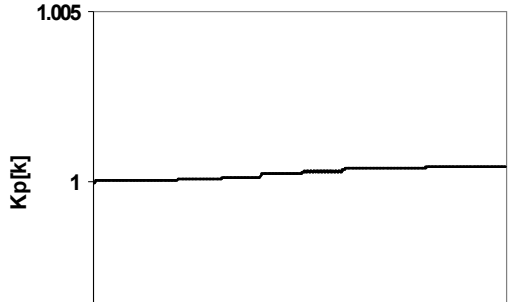


Fig (18-a): Kp gain of the PID Controller for Q=1 & R=0

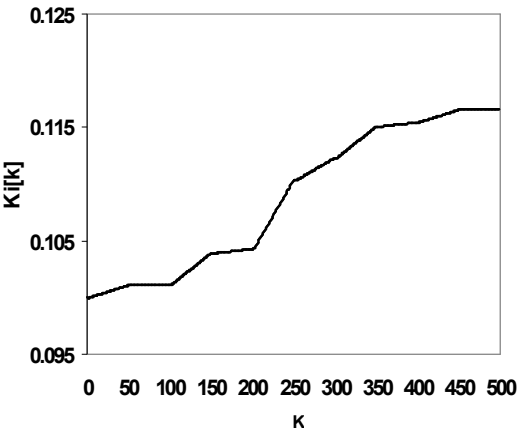


Fig (18-b): Ki gain of the PID Controller for Q=1 & R=0

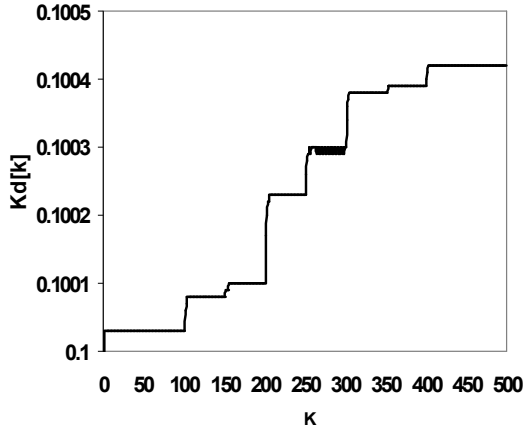


Fig (18-c): Kd gain of the PID Controller for Q=1 & R=0

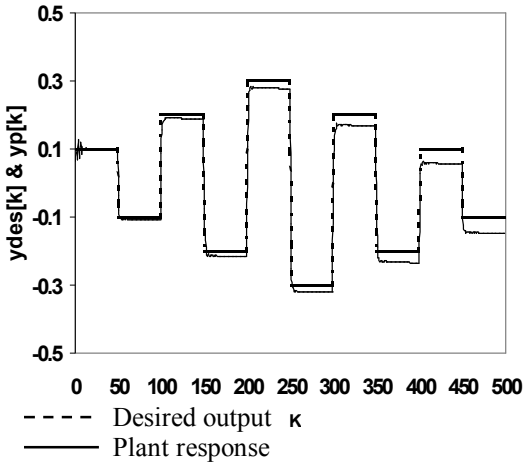


Fig (19): Desired output tracking for one-step ahead with Q=0 & R=1

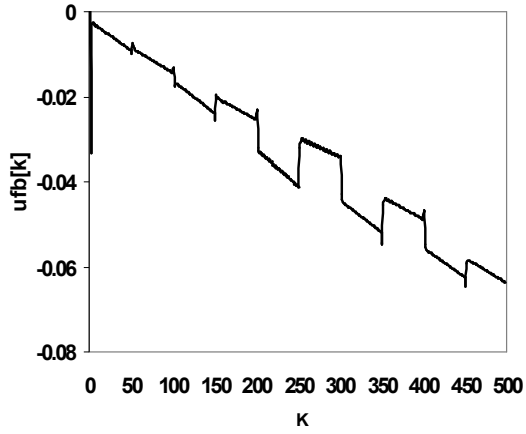


Fig (20): The feedback control signal for Q=1 & R=0

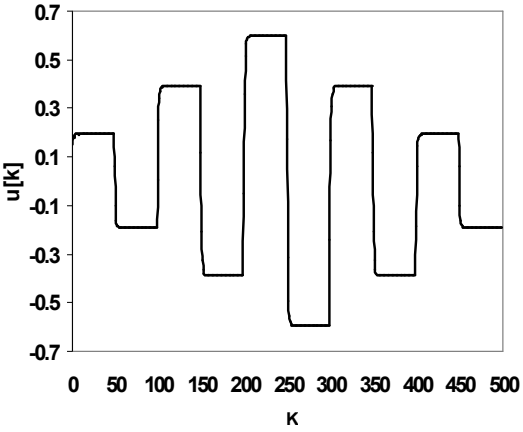


Fig (21): The total control signal for Q=1 & R=0

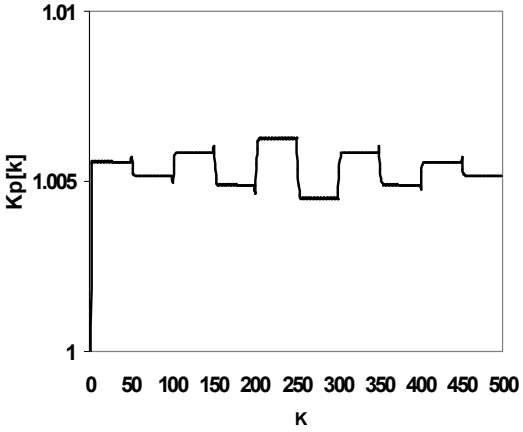


Fig (22-a): Kp gain of the PID Controller for Q=0 & R=1

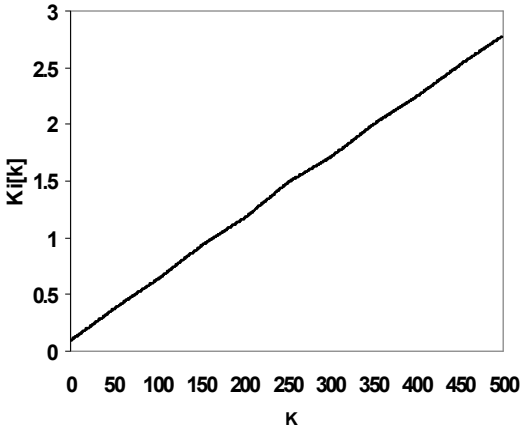


Fig (22-b): Ki gain of the PID Controller for Q=0 & R=1

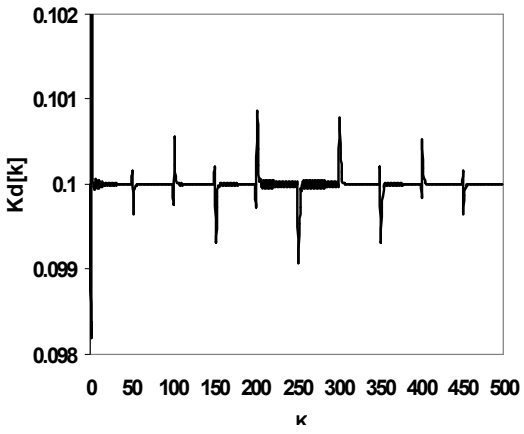


Fig (22-c): Kd gain of the PID Controller for Q=0 & R=1

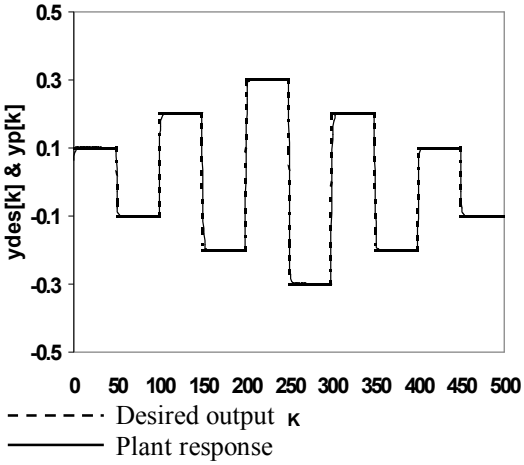


Fig (23): Desired output tracking for five-step ahead with Q=1 & R=1

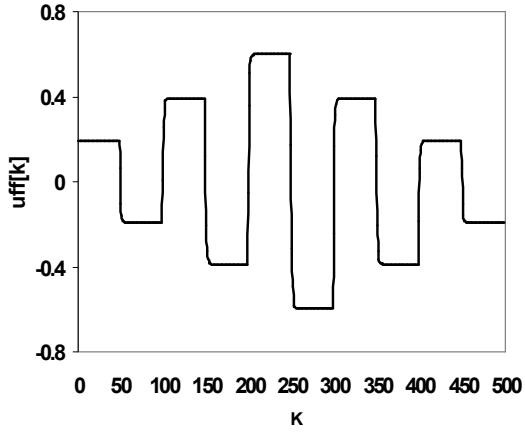


Fig (24): The feedforward control signal for five-step ahead with Q=1 & R=1

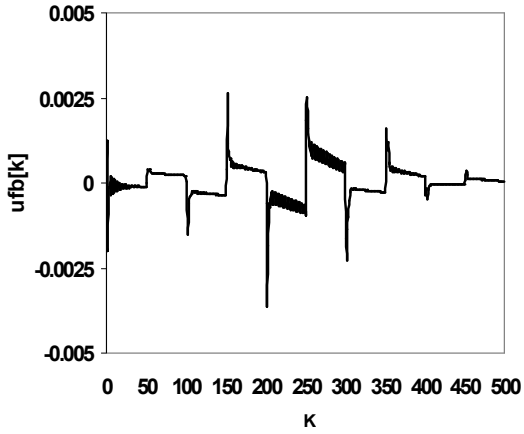


Fig (25): The feedback control signal for five-step ahead with Q=1 & R=1

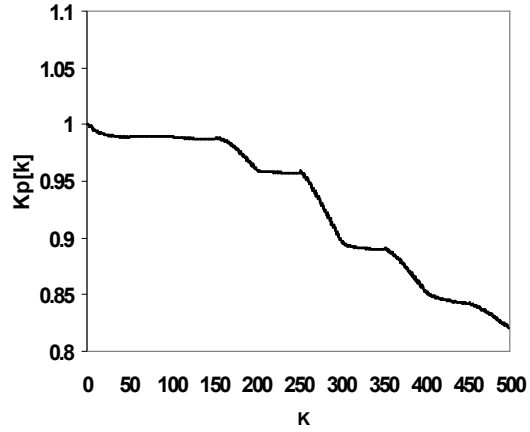


Fig (26-a): Kp gain of the PID Controller for five-step ahead with Q=1 & R=1

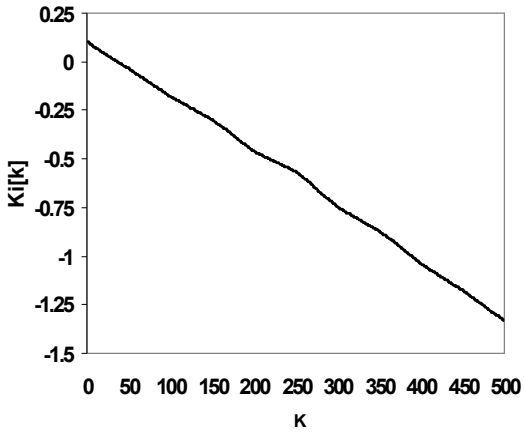


Fig (26-b): Ki gain of the PID Controller for five-step ahead with Q=1 & R=1

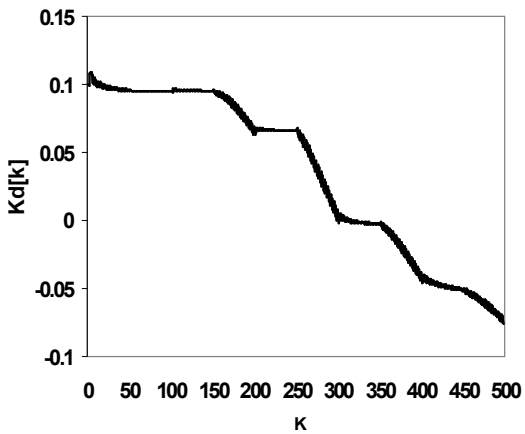


Fig (26-c): Kd gain of the PID Controller for five-step ahead with Q=1 & R=1