Neural Internal Model Control for tracking Unknown Nonaffine Nonlinear Discrete-time Systems under External Disturbances

T. A. A. Alzohairy
Assistant Proffessor
Community College in Arriyadh
King Saud University

ABSTRACT
Tracking of nonlinear dynamic plants under external disturbances is presently an active area of research. The main difficulty felt in the tracking of nonlinear dynamic plants under external disturbances is the computational complexity in control design. This paper presents a simple neural network internal model control (IMC) approach for off-line tracking of unknown nonaffine nonlinear discrete time systems subject to external disturbances. The proposed control scheme is based on the neural network plant model and the inverse model. These models are determined using input output data. The final neural network IMC approach, including network plant model and the inverse model, is work off-line for tracking nonaffine nonlinear discrete time systems subject to external disturbances. Simulation results have been presented toward the end of the paper to illustrate the effectiveness of the proposed control strategy for tracking unknown nonaffine nonlinear discrete-time systems with and without external disturbances.

General Terms
Control systems, Neural networks, Nonlinear systems.

Keywords
Input-output approximation, Inverse model, Multilayer perceptron, Nonlinear Internal Model Control (IMC), Nonaffine nonlinear discrete-time system, Single input single output (SISO) systems.

1. INTRODUCTION
Artificial neural networks (ANNs) have shown an excellent ability to model any nonlinear function to a desired degree of accuracy [8]. Because of this property, they are suitable for the identification and control of nonlinear plants [14]. From the different classes of networks, feedforward neural networks and particularly multi-layer perceptrons (MLPs) are the most frequently used for nonlinear control.

In recent years there has been a significant increase in the number of neural network based control techniques. One such method is the neural network internal model control (IMC) strategy. This strategy is one of the important techniques used for the control of unknown nonlinear discrete-time systems subject to uncertainainties (model mismatch and disturbances) because of its robustness against uncertainties [2],[4], [5], [9], [11] ,[21]. The IMC-based ANN strategy consists of training a network to learn the plant dynamics. Another neural network is trained to learn the inverse dynamics so that it can be used as a nonlinear controller.

In general the inversion of nonlinear models is not an easy task and analytical solutions may not exist, so solutions have to be found numerically. One important point is that the inversion of the plant model may lead to unstable controllers when the plant has unstable zeros. Fortunately, there are several strategies for obtaining the inverse model so that the nonlinear performance can be fully exploited in order to cope with a complex plant [3].

It is well known that nonaffine nonlinear plant models represent a much broader group of both real-world and academic systems. For an unknown nonaffine nonlinear discrete-time system, its output depends nonlinearly on its input. Therefore, it is no longer a simple task to determine the control input of such a nonaffine nonlinear system.

Most frequently used nonaffine nonlinear discrete-time systems are described by the nonlinear autoregressive moving average with eXogenous input (NARMAX) representation [1].

Recently, neural network control for unknown nonaffine nonlinear discrete-time systems has received considerable attention for its academic challenge and its practical interest. Adaptive neural network for such nonlinear systems based on online identification with backpropagation is given in [16]. The study of adaptive inverse control in which dynamic gradient methods were used to adjust the neural networks weights is given in [17]. Control of nonaffine nonlinear discrete-time systems using reinforcement learnining based linearly parameterized neural networks is given in [19].

All of the above mentioned results and others are limited to on-line control. In this paper the plant NARMAX representation and IMC based on artificial neural networks are used to produce an efficient off-line control scheme for tracking unknown nonaffine nonlinear discrete-time systems under external disturbances. As we will see in the simulation results, the neural network IMC strategy shows satisfactory performance when it is used to control unknown nonaffine nonlinear discrete-time systems with and without disturbances.

This paper is organized as follows: In section 2 the problem under consideration is stated. Section 3 gives details of the nonlinear internal model control (IMC) based neural networks. The input-output representation used for unknown nonaffine nonlinear plant modeling, the network architecture used, and full details of the training algorithm used for modeling are given in Section 4. In Section 5, the nonlinear relation used by neural network to find model of the plant inverse, the network architecture used, and training algorithm used for network weights learning are given. The block

diagram of the internal model control (IMC) structure with details of the implementation of model and inverse is given in section 6. Section 7 presents the simulation results for tracking SISO nonaffine nonlinear discrete-time systems with or without disturbances. A comparison study when control SISO nonaffine nonlinear discrete-time systems done with and without disturbances is shown in section 8. Finally, some conclusions are remarked in Section 9.

2. STATEMENT OF THE PROBLEM

Consider the following SISO nonaffine nonlinear discrete-time system [20]:

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= x_3(k) \\ &\vdots \\ x_n(k+1) &= s[X(k), u(k)] \\ y(k+1) &= h[X(k)] \end{aligned} \tag{1}$$

where $X(k) = [x_1(k), x_2(k), \ldots, x_n(k)]^T \in R^n$ is the internal state of the system, $u(k) \in R$ is the input to the system, $y(k) \in R$ is the measured output, and $s(.)$ and $h(.)$ are unknown smooth nonlinear functions.

The control objective is to design a control input $u(k)$, such that the system output $y(k)$ subjected to disturbances d follows a known and bounded trajectory $r(k) \in R$.

3. NONLINEAR IMC USING NEURAL NETWORKS

For the control of nonlinear discrete-time systems subject to uncertainties (model mismatch and disturbances)nonlinear internal model control (IMC) using neural networks has received much attention [10], [12], [18]. Nonlinear IMC was proposed by [6], as shown in Fig. 1. The key characteristic of this type of control strategy is to have the inverse controller and the internal model. In Fig. 1, the model of the nonlinear plant is needed as the internal model. Using this internal model, the effect of uncertainties can be suppressed with the feedback signal generated. In nonlinear IMC, the nonlinear model and its inversion play a crucial role. Fig. 2 gives a basic neural network IMC structure [15], which is an extension of nonlinear IMC. For the control of unknown nonlinear discrete systems using the nonlinear IMC structure, neural network model is employed as the internal model and neural network inverse controller is used to replace the inverse controller, as shown in Fig. 2. The control structure given in Figs. 1 and 2 has been shown to have good robustness against uncertainties [10], [11], [12].

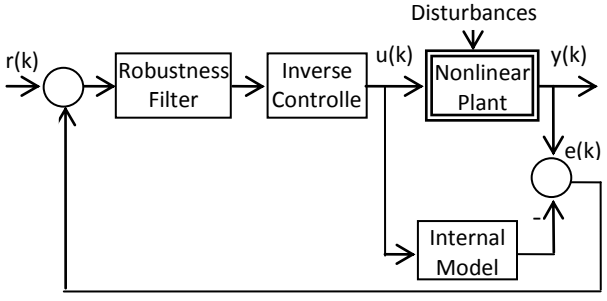

Fig.1: Basic Structure of nonlinear IMC.

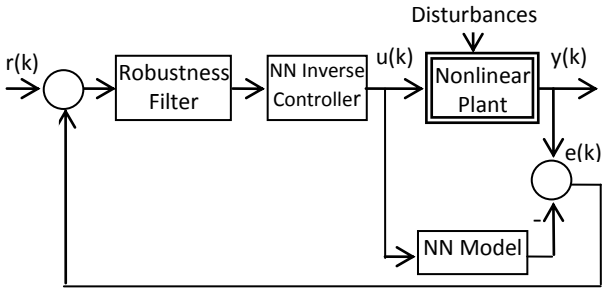

Fig. 2: Basic Structure of NN IMC.

4. NN MODELING

The first step in performing control using IMC strategy is to find model of the plant. The model of the plant should capture the dynamics of the plant well enough so that a controller designed to control the plant model will also control the plant very well.

For convenience of analysis, the future output of the SISO nonaffine nonlinear discrete-time system (1) is determined by a number of past observations of the inputs and outputs. An equivalent input-output representation can be written as the nonlinear auto regressive moving average with exogenous inputs (NARMAX) system:

$$\begin{aligned} y(k+1) = f[&y(k), y(k-1), \ldots, y(k-n+1), \\ &u(k), u(k-1), \ldots, u(k-m+1)] \end{aligned} \tag{2}$$

where n and m are the highest orders of the output and input, respectively, $f(.)$ is the smooth nonlinear function.

The series-parallel identification model corresponding to a plant represented by (2) has the form shown in Fig. 3. TDL in Fig. 3 denotes a tapped delay line whose output vector has for its elements the delayed values of input signal. Hence the past values of the input and the output of the plant form the input vector to a neural network whose output $\hat{y}(k+1)$ corresponds to the estimate of the plant output at any instant of time $k+1$. The series-parallel model enjoys several advantages: 1) since the plant is assumed to be BIBO stable, all the signals used in the identification procedure are bounded. 2) since no feedback loop exists in the model, static back propagation can be used to adjust parameters reducing the computational overhead substantially. 3) assume that the output error tends to small value asymptotically i.e., $\hat{y}(k+1) \approx y(k+1)$, the series-parallel model may be replaced by a parallel model without serious consequences. This has practical implications if the identification model is to be used off line.

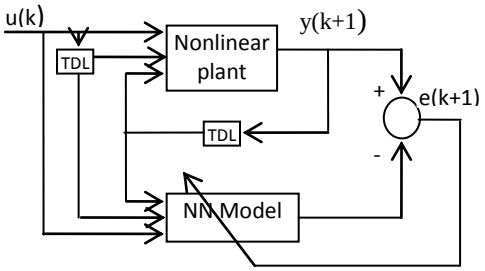

Fig. 3: Architecture for finding neural network internal model of the nonaffine plant represented by a NARMAX model (2).

In this paper, a multilayer network with an input layer, an output layer with one neuron, and two hidden layers is used to model the nonaffine plant represented by a NARMAX model (2).

The multilayer network performs the calculations such that the output of the jth neuron in 1st and the output of the rth neuron in 2nd hidden layers are expressed as

$$I_j = f_j^{h1}\left[\sum_i w_{ji}^{h1} X_i + b_j^{h1}\right] \quad (3)$$

$$Z_r = f_r^{h2}\left[\sum_j w_{rj}^{h2} I_j + b_r^{h2}\right] \quad (4)$$

Where X_i is the *i*th network input, w_{ji}^{h1} is the connection weight from the ith input to the *j*th neuron in the 1st hidden layer, w_{rj}^{h2} is the connection weight from the *j*th neuron in the 1st hidden layer to the *r*th neuron in the 2nd hidden layer, b_j^{h1} is the weight from the bias to the *j*th neuron in the 1st hidden layer, b_r^{h2} is the weight from the bias to the *r*th neuron in the second hidden layer, $f_j^{h1}(.)$ and $f_r^{h2}(.)$ are nonlinear sigmoid activation functions defined as

$$f(\text{netinput}) = \frac{1}{1+e^{-\text{netinput}}} \quad (5)$$

The network output is calculated by the following equation

$$\hat{y}(k+1) = f^o\left[\sum_r w_r^o Z_r + b^o\right] \quad (6)$$

where w_r^o is the weight connection of the neuron in the output layer to the *r*th neuron in the 2nd hidden layer, b^o is the bias weight for the output neuron, and f^o is the transformation function between 2nd hidden layer and output layer and it is a linear function.

The popular backpropagation algorithm for training the multilayer network is gradient descent-based algorithm to minimize the following cost function

$$J = \frac{1}{2}(y(k+1) - \hat{y}(k+1))^2 \quad (7)$$

A recursive algorithm, starting at the output nodes and working back to the first hidden layer is used to adjust weights, is given by

$$\begin{cases} w_r^o(k+1) = w_r^o(k) + \eta\delta^o Z_r \\ w_{rj}^{h2}(k+1) = w_{rj}^{h2}(k) + \eta\delta_r^{h2} I_j \\ w_{ji}^{h1}(k+1) = w_{ji}^{h1}(k) + \eta\delta_j^{h1} X_i \end{cases} \quad (8)$$

where η is the learning rate $0 < \eta < 1$ and

$$\begin{cases} \delta^o = y(k+1) - \hat{y}(k+1) \\ \delta_r^{h2} = Z_r(1-Z_r).\delta^o w_r^o \\ \delta_j^{h1} = I_j(1-I_j)\sum_r \delta_r^{h2} w_{rj}^{h2} \end{cases} \quad (9)$$

Full details of the training algorithm used are given in [7], [14].

After training, the plant output measurements in the series-parallel model

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-m+1)] \quad (10)$$

are replaced by the model outputs in the parallel model

$$\hat{y}(k+1) = \hat{f}[\hat{y}(k), \hat{y}(k-1), \ldots, \hat{y}(k-n+1), u(k), u(k-1), \ldots, u(k-m+1)] \quad (11)$$

This parallel model is used for neural network inverse modeling as we will see in the following section.

5. NN INVERSE MODELING

Inverse modeling plays a crucial role in IMC structure. The inverse learning structure is shown in Fig. 4. In this structure the plant model precedes the network to be trained as the inverse. We use the plant model because the plant is unknown. Using this architecture, we are able to explicitly calculate the partial derivatives of the plant model output with respect to its input. From (2), the inverse function f^{-1} leading to the generation of $u(k)$ would require knowledge of the future value $y(k+1)$. To overcome this problem, we replace this future value with the value of the reference signal $r(k+1)$, which we assume is available at time k. Thus, the nonlinear relation of the network modeling the plant inverse is given by

$$u(k) = \hat{f}^{-1}(\hat{y}(k), \ldots, \hat{y}(k-n+1), r(k+1), u(k-1), \ldots, u(k-m+1)) \quad (12)$$

The condition governing the inevtibility of known system is that the linearization of the plant (1) around the equilibrium state is controllable.

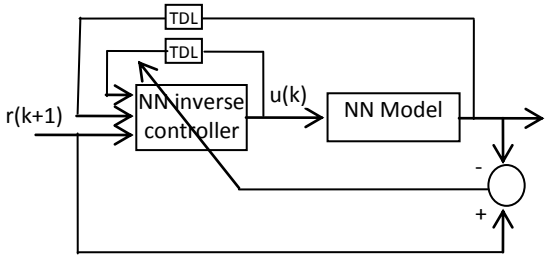

Fig. 4: Plant model inverse modeling using neural networks.

The structure of inverse neural network is similar to that used for plant modeling. A multilayer network with an input layer, an output layer with one node, and two hidden layers is used to model the inverse model represented by (12).

The inverse multilayer network performs the calculations

$$I_j^c = f_j^{h1}\left[\sum_i w_{ji}^{h1} X_i^c + b_j^{h1}\right] \quad (13)$$

$$Z_r^c = f_r^{h2}\left[\sum_j w_{rj}^{h2} I_j^c + b_r^{h2}\right] \quad (14)$$

$$u(k) = f^{o}\left[\sum_{r} w_{r}^{o} Z_{r}^{c} + b^{o}\right] \tag{15}$$

where the input vector is given by

$$\begin{aligned} X^{c} &= (X_{0}^{c}, X_{1}^{c}, \ldots, X_{n+m}^{c})^{T} \\ &= (\hat{y}(k), \ldots, \hat{y}(k-n+1), r(k+1), u(k-1), \ldots, u(k-m+1))^{T} \end{aligned} \tag{16}$$

The criterion to be minimized is given by:

$$J_{c} = \frac{1}{2}(r(k+1) - \hat{y}(k+1))^{2} \tag{17}$$

A recursive algorithm, used to adjust weights, is given by

$$\begin{cases} w_{r}^{o}(k+1) = w_{r}^{o}(k) - \eta \dfrac{\partial J_{c}}{\partial w_{r}^{o}} \\ w_{rj}^{h2}(k+1) = w_{rj}^{h2}(k) + \eta \dfrac{\partial J_{c}}{\partial w_{rj}^{h2}} \\ w_{ji}^{h1}(k+1) = w_{ji}^{h1}(k) + \eta \dfrac{\partial J_{c}}{\partial w_{ji}^{h1}} \end{cases} \tag{18}$$

where

$$\begin{cases} \dfrac{\partial J_{c}}{\partial w_{r}^{o}} = \dfrac{\partial J_{c}}{\partial \hat{y}(k+1)} \cdot \dfrac{\partial \hat{y}(k+1)}{\partial u(k)} \cdot \dfrac{\partial u(k)}{\partial w_{r}^{o}} \\ \dfrac{\partial J_{c}}{\partial w_{rj}^{h2}} = \dfrac{\partial J_{c}}{\partial \hat{y}(k+1)} \cdot \dfrac{\partial \hat{y}(k+1)}{\partial u(k)} \cdot \dfrac{\partial u(k)}{\partial w_{rj}^{h2}} \\ \dfrac{\partial J_{c}}{\partial w_{ji}^{h1}} = \dfrac{\partial J_{c}}{\partial \hat{y}(k+1)} \cdot \dfrac{\partial \hat{y}(k+1)}{\partial u(k)} \cdot \dfrac{\partial u(k)}{\partial w_{ji}^{h1}} \end{cases} \tag{19}$$

and

$$\frac{\partial J_{c}}{\partial \hat{y}(k+1)} = -(r(k+1) - \hat{y}(k+1)) \tag{20}$$

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{r}\sum_{j} w_{r}^{o} . Z_{r}^{c} . (1 - Z_{r}^{c}) . w_{rj}^{h2} . I_{j}^{c} . (1 - I_{j}^{c}) . w_{ji}^{h1} \tag{21}$$

$$\begin{cases} \dfrac{\partial u(k)}{\partial w_{r}^{o}} = Z_{r}^{c} \\ \dfrac{\partial u(k)}{\partial w_{rj}^{h2}} = w_{r}^{o} . Z_{r}^{c} . (1 - Z_{r}^{c}) * I_{j}^{c} \\ \dfrac{\partial u(k)}{\partial w_{ji}^{h1}} = \sum_{r} w_{r}^{o} . Z_{r}^{c} . (1 - Z_{r}^{c}) * w_{rj}^{h2} * I_{j}^{c} . (1 - I_{j}^{c}) * X_{i}^{c} \end{cases} \tag{22}$$

The training process is continued until the control sequence leads to minimize the difference between the plant model and the reference signal.

6. IMC CONTROLLER DESIGN

After training the plant neural network and inverse neural network models, as given in sections 4 and 5, we can incorporate them in the NN IMC structure given in section 3, Fig. 2.

Because plant neural network and inverse neural network models are implemented according to equations (11) and (12)

$$\begin{aligned} \hat{y}(k+1) = \hat{f}[&\hat{y}(k), \hat{y}(k-1), \ldots, \hat{y}(k-n+1), \\ &u(k), u(k-1), \ldots, u(k-m+1)] \end{aligned}$$

$$\begin{aligned} u(k) = \hat{f}^{-1}(&\hat{y}(k), \ldots, \hat{y}(k-n+1), r(k+1), \\ &u(k-1), \ldots, u(k-m+1)) \end{aligned}$$

the block diagram given in Fig. 5 gives more details about the Internal Model Control structure.

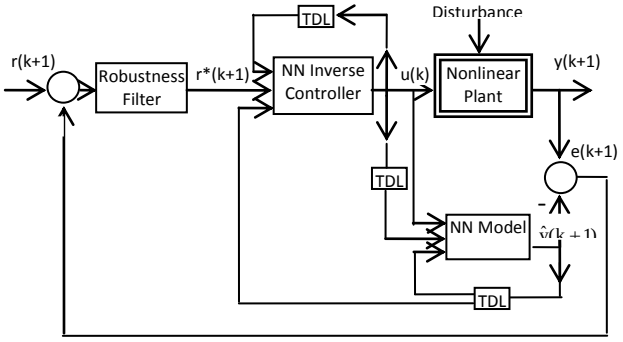

Fig. 5: Detailed structure of Internal Model Control

The difference between the nonlinear plant output and plant neural network model (internal model) output represents the effect of disturbances and internal model mismatch.

If the NN inverse controller static gain is equal to the inverse of the internal model static gain and if the overall closed loop system is stable, this structure presents robustness properties against disturbances and modeling mismatch; it allows having an offset-free response by canceling the noise at the process output.

In practice perfect model cannot be obtained. In addition, the infinite gain required by perfect control would lead to sensitivity problem under model uncertainty.

A filter is usually added at the input of the NN inverse controller to attenuate uncertainties in the feedback, generated by the difference between plant and model outputs and serves to moderate excessive control effort. In [11] it was found that to ensure robust stability, whether the model order, it is always possible to use first order filter of the form

$$r^{*}(k+1) = \frac{(1-\alpha)z^{-1}}{1-\alpha z^{-1}}[r(k+1) - e(k+1)] \tag{23}$$

where α is the filter tuning parameter and $r^{*}(k+1)$ is the filter output.

According to the Fig. 5, the models need to match, that is the inverse model should be the inverse of the forward model instead of the inverse of the system.

7. SIMULATION RESULTS

This section demonstrates the application of the IMC strategy on the unknown nonaffine nonlinear systems for both tracking and disturbance rejection studies.

Example 1:

Consider the liquid level system described by the following equation [20]:

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= 0.9722x_2(k) - 0.3513x_2^2(k)u(k-1) \\ &\quad - 0.3103x_2(k)u(k) + 0.1633x_1(k)u(k-1) \\ &\quad - 0.03259x_2^2(k)x_1(k) - 0.1295u(k-1) \\ &\quad + 0.3084x_2(k)x_1(k)u(k-1) - 0.04228x_1^2(k) \\ &\quad + 0.1087x_1(k)u(k)u(k-1) + 0.3578u(k) \\ y(k) &= x_2(k) \end{aligned} \tag{24}$$

The NARMAX model of the process should be written as:

$$\begin{aligned} y(k+1) &= 0.9722y(k) + 0.3578u(k) - 0.1295u(k-1) \\ &\quad + 0.3084y(k)y(k-1)u(k-1) - 0.04228y^2(k-1) \\ &\quad + 0.1633y(k-1)u(k-1) - 0.03259y^2(k)y(k-1) \\ &\quad - 0.3513y^2(k)u(k-1) - 0.3103y(k)u(k) \\ &\quad + 0.1087y(k-1)u(k)u(k-1) \end{aligned} \tag{25}$$

The control objective is to make the liquid level $y(k+1)$ to follow a desired trajectory

$$r(k+1) = 0.3\,\mathrm{sign}(\sin(\pi k/50)) + 0.4 \tag{26}$$

In order to explain the disturbance-rejection capabilities of the Internal model control (IMC) structure, additive noise plus the influence of unknown plant dynamics are considered. The following additive noise is assumed to corrupt the output:

$$d(k) = 0.01*(\sin(8t) + \sin(12t) + \sin(23.66t) + \sin(35.49t)) \tag{27}$$

To control this unknown nonlinear plant, the control process is divided into the following steps:

Step 1: Plant modeling

Using random input signal uniformly distributed over the interval $[-1,1]$ and input/output patterns generated from the unknown plant, neural network model of the plant with 10 nodes in the 1st hidden layer and 5 neurons in the 2nd hidden layer can be trained using the rules given in section 4 to model the plant (24) represented by NARMAX model given by

$$y(k+1) = f[y(k), y(k-1), u(k), u(k-1)] \tag{28}$$

The initial network weight values are selected randomly from the interval $[-1,1]$ and the learning rate (η) used is equal 0.2. Fig. 6 shows the output of the plant and the output of the model for a randomly selected input signal uniformly distributed over the interval $[-1,1]$. Fig. 7 shows the error between model and plant which proves the accuracy of the modeling process. So the neural network model is used to find neural network inverse controller.

Step 2: Using neural network plant model obtained from step 1 and the architecture given in Fig. 4, the neural network inverse controller represented by

$$u(k) = f^{-1}[y_m(k), y_m(k-1), r(k+1), u(k-1)] \tag{29}$$

with 10 nodes in the 1st hidden layer and 5 neurons in the 2nd hidden layer can be obtained. The initial network weight values are selected randomly from the interval $[-1,1]$ and the learning rate (η) used is equal 0.08. The rules used for the learning process are given in section 5. Fig. 8 shows the relation between input to the controller and output of the model. It can be seen from Fig. 8 that the relation is linear, as desired. Fig. 9 shows the tracking capabilities of the IMC strategy when the desired trajectory is given by (26) and the using the first-order filter F of the form given by

$$F = \frac{0.5z^{-1}}{1 - 0.5z^{-1}} \tag{30}$$

Fig. 10 gives the error between the reference signal $r(k+1)$ and the plant output $y_p(k+1)$ result when IMC strategy is used for tracking. The control input for the system (24) obtained using IMC strategy is given in Fig. 11.

Fig. 12 shows the disturbance given by (27). Fig. 13 shows that the performance of the IMC structure is satisfactory when controlling the system subjected to disturbance given by (27). The error between the reference signal $r(k+1)$ and the plant output $y_p(k+1)$, result when IMC strategy is used to control system subjected to disturbance (27), is shown in Fig. 14.

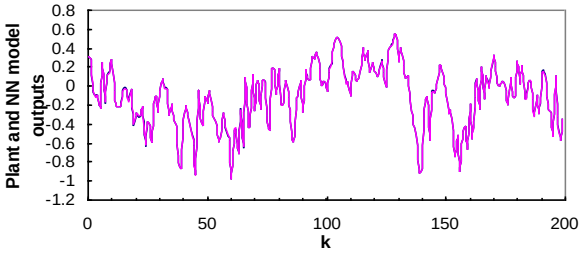

Fig. 6: Model and Plant response using random input uniformly distributed over the interval $[-1,1]$.

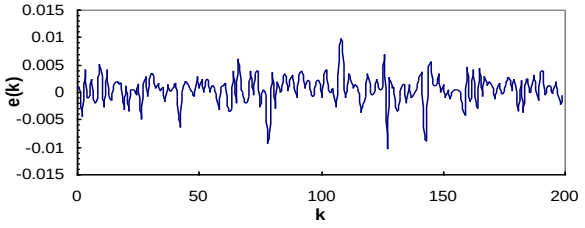

Fig. 7: Error between model and plant response when using random input uniformly distributed over the interval $[-1,1]$.

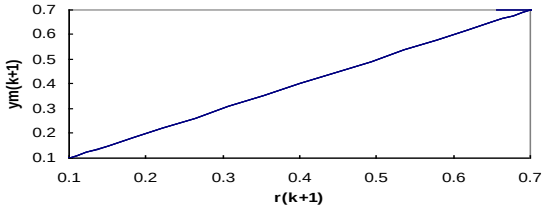

Fig. 8: Relation between input to the controller and output of the model.

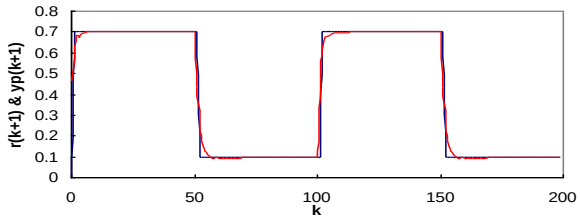

Fig. 9: IMC tracking performance without disturbance.

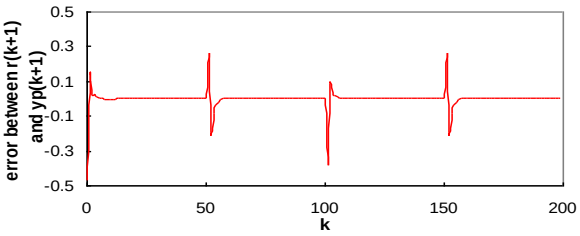

Fig. 10: Error between reference signal and output of the system.

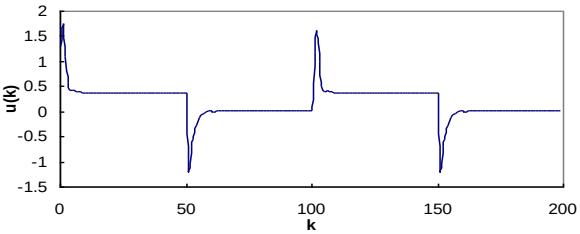

Fig. 11: Control signal for the system (24) in case of reference model (26) when using IMC strategy.

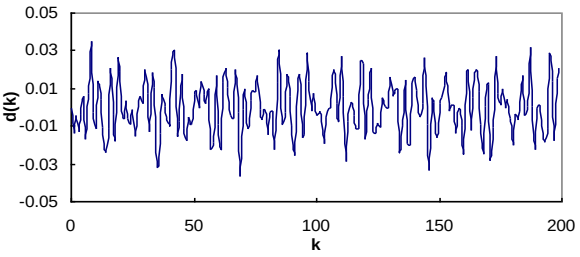

Fig. 12: Disturbance of the plant (24).

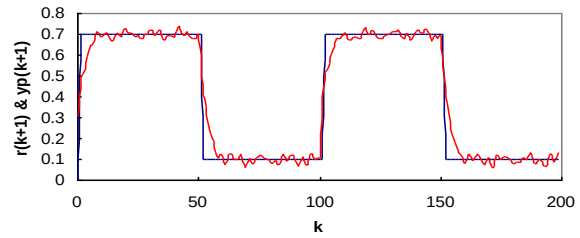

Fig. 13 IMC tracking performance under disturbance (27).

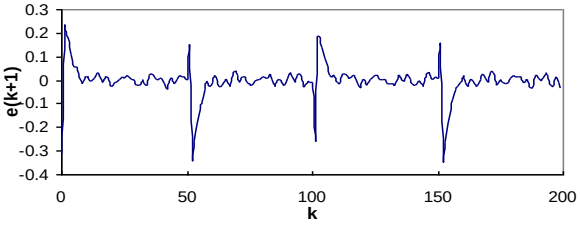

Fig. 14: Error between reference signal and output of the system when the system under disturbance (27).

Example 2:

Consider the first order SISO system as described in the following difference equation [13]:

$$y(k+1) = \sin[y(k)] + u(k)(5 + \cos[y(k)u(k)]) \qquad (31)$$

The objectives of simulations include the following:

1) Test the approximation performance of the NN input-output approximation model under the following test signal with amplitude.

$$u(k) = 0.5\sin\left(\frac{2\pi k}{50}\right) + 0.5\sin\left(\frac{2\pi k}{25}\right) \qquad (32)$$

2) Track the following trajectories with magnitude:

$$r(k) = 2\sin\left(\frac{2\pi k}{50}\right) + 2\sin\left(\frac{2\pi k}{100}\right) \qquad (33)$$

when using the neural network IMC strategy.

3) Explain the disturbance-rejection capabilities of the Internal model control (IMC) structure, additive noise plus the influence of unknown plant dynamics are considered. The following additive noise is assumed to corrupt the output:

$$d(k) = 0.01*(\cos(3t)) \qquad (34)$$

To control this unknown nonlinear plant, the control process is divided into the following steps:

Step 1: Plant modeling

using random input signal uniformly distributed over the interval $[-1,1]$ and input/output patterns generated from the unknown plant, neural network model of the plant with 20 nodes in the 1st hidden layer and 11 neurons in the 2nd hidden layer can be trained as seen in section 4 to model the plant (31) represented by NARMAX model as given by

$$y(k+1) = f[y(k), u(k)] \qquad (35)$$

The initial network weight values are selected randomly from the interval $[-1,1]$ and the learning rate (η) used is equal 0.1. Fig. 15 shows the output of the plant and the output of the model for a randomly selected input signal uniformly distributed over the interval $[-1,1]$. Fig. 16 shows the error between model and plant which proves the accuracy of the modeling process. So the neural network model is used to find neural network inverse controller.

Step 2: Using neural network plant model obtained from step 1 and the architecture given in Fig. 4, the neural network inverse controller represented by

$$u(k) = f^{-1}[y_m(k), r(k+1)] \qquad (36)$$

with 20 nodes in the 1st hidden layer and 11 neurons in the 2nd hidden layer can be obtained. The initial network weight values are selected randomly from the interval $[-1,1]$ and the learning rate (η) used is equal 0.02. Section 5 gives the details of the learning process. Fig. 17 shows the relation between input to the controller and output of the model. It can be seen from Fig. 17 that the relation is linear, as desired. Fig. 18 shows the tracking capabilities of the IMC strategy when the desired trajectory is given by (33) and using the first-order filter F of the form given by (30).

Fig. 19 gives the error between the reference signal $r(k+1)$ and the plant output $y_p(k+1)$ result when IMC strategy is used for tracking. The control input for the system (31) obtained using IMC strategy is given in Fig. 20.

Fig. 21 shows the disturbance given by (34). Fig. 22 shows that the performance of the IMC structure is satisfactory when controlling the system subjected to disturbance given by (34). The error between the reference signal $r(k+1)$ and the plant output $y_p(k+1)$, result when IMC strategy is used to control system (31) subjected to disturbance (34), is shown in Fig.23.

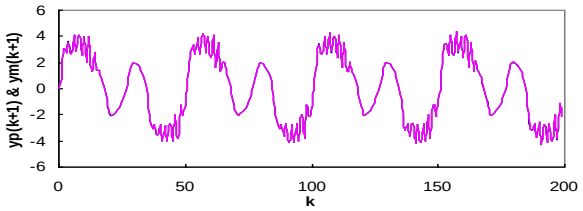

Fig. 15: Model and Plant response using random input uniformly distributed over the interval $[-1,1]$.

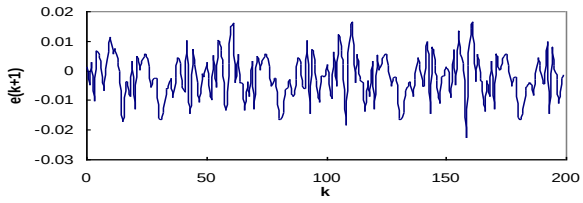

Fig. 16: Error between model and plant response when using random input uniformly distributed over the interval $[-1,1]$.

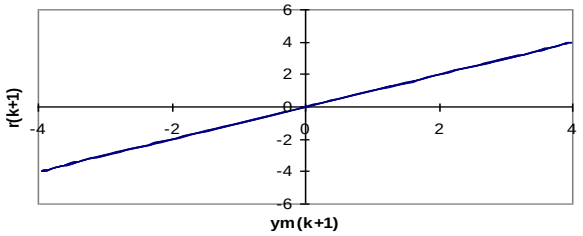

Fig. 17: Relation between input to the controller and output of the model.

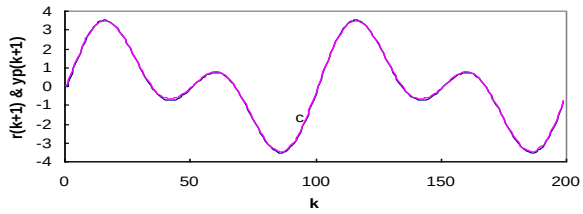

Fig. 18: IMC tracking performance without disturbance.

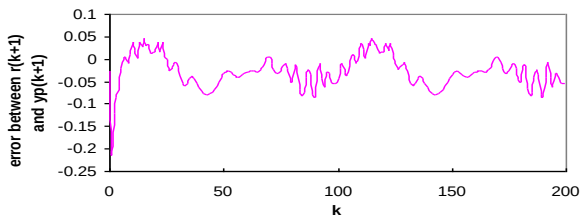

Fig. 19: Error between reference signal and output of the system.

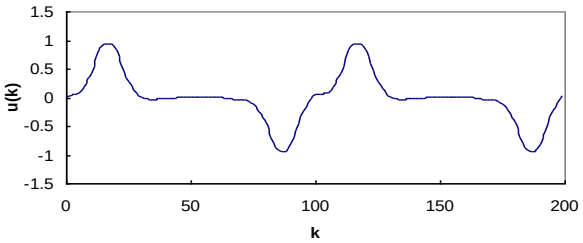

Fig. 20: Control signal for the system (31) in case of reference model (33) when using IMC strategy.

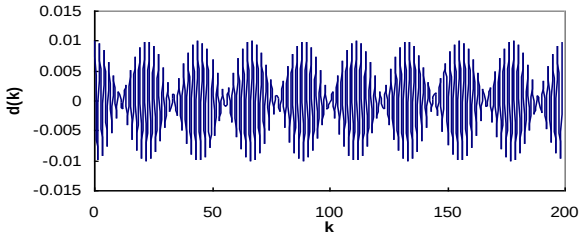

Fig. 21: Disturbance of the plant (31).

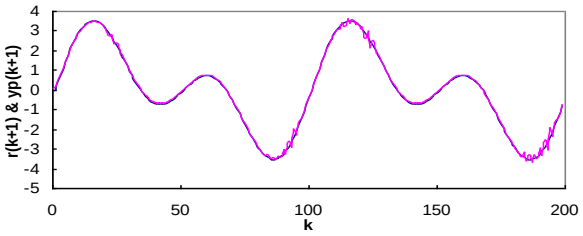

Fig. 22: IMC tracking performance when the system is under disturbance (34).

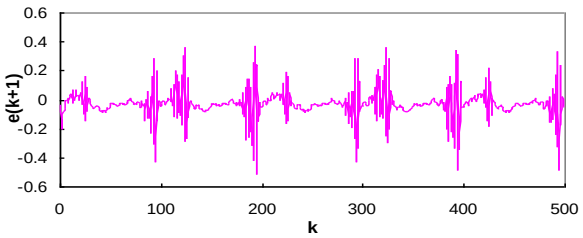

Fig. 23: Error between reference signal and output of the system when the system is under disturbance (34).

8. COMPARISON STUDY

In this section, we will make a comparison between two control cases 1) when nonaffine nonlinear discrete-time systems are subject to disturbances and 2) when nonaffine nonlinear discrete-time systems are without any disturbances. Related to example 1, Mean square error, between the reference model $r(k+1)$ and the plant output $y_p(k+1)$, in the first case is given by 4.8×10^{-3} and in the second case is given by 3×10^{-3}. Related to example 2, Mean square error, between the reference model $r(k+1)$ and the plant output $y_p(k+1)$, in the first case is given by 1.2×10^{-2} and in the second case is given by 1.9×10^{-3}. Its clear that the difference is very small which prove the ability of the internal model control technique based neural networks to track unknown nonaffine nonlinear discrete-time systems under external disturbances.

9. CONCLUSION

This paper described the use of nonlinear internal model control (IMC), based on two feedforward neural networks, to control unknown nonaffine nonlinear systems. The neural network models (internal model and inverse model) are obtained from input-output data using a three-layer feedforward network trained with a backpropagation algorithm. The nonlinear IMC controller consists of a model inverse controller and a robustness filter with a single tuning parameter. Simulation results for two unknown nonaffine nonlinear systems demonstrate the ability of the IMC strategy to perform tracking control. In view of the flexibility that neural networks provide in modeling unknown nonlinear systems, the nonlinear IMC is potentially applicable to control unknown nonaffine nonlinear systems subjected under external disturbances.

10. ACKNOWLEDGEMENT

The author would like to thank King Saud University for its academic and financial support throughout the research work.

11. REFERENCES

[1] Adetona O., Sathananthan S. and Keel L. H. 2004. Robust adaptive control of nonaffine nonlinear plants with small input signal changes. IEEE Trans. Neural Networks, l5(2), 408-416.

[2] Awais M. M. 2005. Application of internal model control methods to industrial combustion. Applied Soft Computing. 5, 223-233.

[3] Carotenuto R. 2001. An iterative system inversion technique. International Journal of Adaptive Control and Signal Processing. 15, 85-91.

[4] Chidrawar S. K. and Patre B. M. 2008. Implementation of neural network for internal model control and adaptive control. International Conference on Computer and Communication Engineering. Kuala Lumpur, Malaysia.

[5] Colin G., Chamaillard Y., Bloch G. and Corde G. 2007. Neural control of fast nonlinear systems – application to a turbocharged SI engine with VCT. IEEE Trans. Neural Networks, l8(4), 1101-1114.

[6] Economou, C. G., Morari, M., and Palsson, B. O. 1986. Internal model control. 5. Extension to nonlinear systems. Ind. Eng. Chem. Press Des. Dev., 25, 403-411.

[7] Haykin, S., 1999. Neural networks. Printice-Hall.

[8] Hornik K, Stincheombe M and White H. 1989. Multilayer feedforwaed networks are universal approximators. Neural Networks, 2, 359-366.

[9] Li H. X. and Deng H. 2006. An approximate internal model based neural network control for unknown nonlinear discrete time systems. IEEE Trans. Neural Networks, l7(3), 659-670.

[10] Lightbody, G., and Irwin, G. W. 1997. Nonlinear control structures based on embedded neural system models. IEEE Trans. Neural Networks, 8(3), 553-567.

[11] Morari M. and Zafiriou E. 1989. Robust process control. Prentice-Hall, Englewood cliffs, NJ.

[12] Nahas, E. P., Henson, M. A., and Seborg, D. E. 1992. Nonlinear internal model control strategy for neural network models. Computa. Chem. Eng., 16(12), 1039-1057.

[13] Narendra K. S. and Mukhopadhyay S. 1997. Adaptive control using neural networks and approximate models. IEEE Trans. Neural Networks, 8, 475-485.

[14] Narendra K. S. and Parthasarathy K., 1990. Identification and control of dynamical systems using neural networks. IEEE Trans. Neural Networks, l(1), 4-27.

[15] Norgaard, M., Ravan, O., Poulsen N. K., and Hansen L. K. 2000. Neural networks for modeling and control of dynamical systems. London, U. K.:Spring-Verlag.

[16] Noriega J. R. and Wang H. 1998. A direct adaptive neural-network control for unknown nonlinear systems and its applications. IEEE Trans. Neural Networks, 9(1), 27-34.

[17] Plett G. L. 2003. Adaptive inverse control of linear and nonlinear systems using dynamic neural networks. IEEE Trans. Neural Networks, 14(2), 360-376.

[18] Rivals, I., and Personnaz, L. Nonlinear internal model control using neural networks: Applications to processes with delay and design issues. IEEE Trans. Neural Networks, 11(1), 80-90.

[19] Yang Q., Vance J. B. and Jagannathan S. 2008. Control of nonlinear discrete-time systems using reinforcement learning based linearly parameterized neural networks. IEEE Trans. Neural Networks, 38(4), 994-1001.

[20] Zhai L. F., Chai T. Y. and Ge S. S. 2007. Stable adaptive neural network control of nonaffine nonlinear discrete-time systems and application. 22th IEEE International Symposium on Intelligent Control, Singapore. 602-607.

[21] Zhao Z., Liu Z., Wen X. and Zhang J. 2008. A new multi-model internal model control scheme based on neural network. Proceeding of the seventh world congress on Intelligent Control and Automatica, Chongqing, china.