Engineering Journal of the University of Qatar, Vol. 13, 2000, pp.

A NEURAL CONTROLLER USING IMC STRATEGY

K. I. Abdul-lateef and K. B. Mirza
Department of Control and Computer Engineering
University of Technology
Baghdad, Iraq.

ABSTRACT

In this paper a nonlinear Internal Model Control (IMC) strategy based on a modified NARMA model is proposed. The IMC controller consists of a model inverse controller and a robustness filter with a single tuning parameter. Under such a controller, the control quality and the robustness can be influenced in a direct manner. From the input-output observation data of the plant, a neural network is trained off-line using Back-Propagation algorithm to emulate the feed-forward dynamics of the plant. This neural plant model provides a simple check of the model invertability, which plays a critical role in the IMC strategy. Using the same network used in the feed-forward model of the plant, an exact model inverse is obtained directly assuming invertability of the plant model. The exact inverse of the plant model is very important to achieve offset-free performance. Simulation examples demonstrate the simplicity of the design procedure and the good performance characteristics of the proposed nonlinear IMC controller. The examples show that the proposed controller is superior to conventional PID controller and it can also be applied to systems having time delay.

INTRODUCTION

Model based control strategies for nonlinear processes usually require local linearization and linear control design based on the linearized model. Since reasonably accurate nonlinear models are available for a variety of processes, control strategies in which the nonlinear process model serves as the basis for controller design can be expected to yield significantly improved performance. There are a number of nonlinear system identification techniques available, however, neural networks offer some of the most versatile ways of modeling nonlinear processes of a diverse nature [1].

Although many different neuro-controller architectures exist yet they have not achieved wide spread acceptance in industrial applications. The lack of general stability theory prevents their use in life-critical situations and slow training algorithms can cause problems when using adaptive systems. Moreover, for real applications of control theories, the control algorithm should be simple enough to be implemented and to be understood. This explains why until 1993, about 84% of the control industry in Japan used the conventional PID controllers [2].

This paper aims at presenting a control algorithm which is simple to be implemented and to be understood, furthermore, it has some desirable properties such as robustness, flexibility, and coping with nonlinearities. One particular control approach for which a well-understood framework exists for the structured utilization of nonlinear plant models is the IMC strategy.

The idea of using neural networks for nonlinear IMC was considered for the first time by Bhat and McAvoy [3]. However, they do not appear to have implemented a neural network based nonlinear IMC controller. Hunt and Sbarbaro [4] implemented a nonlinear IMC based on two neural networks: a neural network is trained to learn the feed-forward dynamics of the plant and a separate neural network is used to learn the inverse dynamics of the plant which is employed as a nonlinear controller. They also investigated invertability conditions for a class of nonlinear dynamical systems. Nahas et al. [5] proposed two alternative implementations of the nonlinear IMC strategy: one is based on the plant model and the other is based on a model of the inverse plant dynamics. Lightbody and Irwin [6] proposed a nonlinear IMC strategy, that utilized a nonlinear neural model of the plant to generate parameter estimates over the nonlinear operating region for an adaptive linear internal model, without the problems associated with the recursive parameter identification algorithms. In this paper it is suggested using an approximate model (NARMA-L2) within the IMC strategy that will attain specific benefits towards a systematic engineering design procedure for neural control systems.

The paper is structured as follows. Section 2 gives an overview of using feed-forward neural networks to learn to act as input-output models. The NARMA-L2 model used in this study is also shown in Section 2. The specific use of this model within the IMC framework is demonstrated in Section 3. Finally, simulation results are presented in Section 4.

NEURAL NETWORK MODELING OF DYNAMICAL SYSTEMS

It is well known that the design of effective analog or digital controllers depends very heavily on how well we know the dynamics of the controlled plant (process). Recently there has been a lot of research into the approximation properties of multi-layered neural networks. It has been proved that a continuous function can be arbitrarily well approximated by a feed-forward network with only a single internal hidden layer, where each unit in the hidden layer has a continuous sigmoidal nonlinearity [7]. The apparent ability of universal approximation by multi-layered neural networks is of interest as it would be very useful in several applications including system identification and control.

Process Input-Output Models

General nonlinear input-output behavior can be well approximated by a NARMA (Nonlinear AutoRegressive Moving Average) model, which can be expressed as:

$$\begin{aligned} y_p(k+1) = \overline{F}[&y_p(k), y_p(k-1),\ldots, y_p(k-n+1), u(k), \\ &u(k-1),\ldots, u(k-n+1)] \end{aligned} \tag{1}$$

Here, n is the order of the system. Thus, the system output (y_p) at time instant $k+1$ depends (in the sense defined by nonlinear map $\overline{F}$) on the past inputs and outputs [8,9]. Then $\overline{F}$ in (1) can be realized by a feed-forward neural network, resulting in the identification model of the form:

$$\begin{aligned} y_m(k+1) = \hat{\overline{F}}[&y_p(k), y_p(k-1),\ldots, y_p(k-n+1), u(k), \\ &u(k-1),\ldots, u(k-n+1)] \end{aligned} \tag{2}$$

Where y_m is the output of the neural network model, and $\hat{\overline{F}}$ represents the nonlinear input-output map of the network (the approximation of $\overline{F}$).

The availability of a NARMA model for a nonlinear plant does not automatically imply a method of determining the control input to track a desired output. This is because of the nonlinear dependence of the output on the input. Narendra and Mukhopadhyay in their paper [10], have proposed two approximate 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. This makes the control problem tractable as in the linear case, since $u(k)$ is merely the ratio of two time-functions. The equations for the two proposed nonlinear models are given below.

NARMA-L1 model:

$$\begin{aligned} y_p(k+1) = &f_0[y_p(k), y_p(k-1),\ldots, y_p(k-n+1)] \\ &+ \sum_{i=0}^{n-1} g_{0_i}[y_p(k), y_p(k-1),\ldots, y_p(k-n+1)] \cdot u(k-i) \end{aligned} \tag{3}$$

NARMA-L2 model:

$$\begin{aligned} y_p(k+1) &= f[y_p(k),\ldots, y_p(k-n+1), u(k-1),\ldots, u(k-n+1)] \\ &+ g[y_p(k),\ldots, y_p(k-n+1), u(k-1),\ldots, u(k-n+1)]\cdot u(k) \end{aligned} \tag{4}$$

If the NARMA-L1 model is used to approximate a given plant, $(n+1)$ networks are needed to approximate the functions f_0 and $g_{0_i}(i = 0,1,\ldots, n-1)$. Each of the functions is scalar-valued and has n arguments $y_p(k), y_p(k-1),\ldots, y_p(k-n+1)$. In contrast to this, the NARMA-L2 model requires only two neural networks to approximate the functions f and g. Each of the functions, however, has $(2n-1)$ inputs. From a practical standpoint, the NARMA-L2 model is simpler to be realized than the NARMA-L1 model, and thus NARMA-L2 model will be used in this paper. In fact, NARMA-L2 model was used previously by F. C. Chen [11], but the mathematical derivation and the conditions under which this model can be used with confidence in a control problem have been proposed by Narendra and Mukhopadhyay [10].

The identification procedure using a feed-forward neural network is quite straightforward. At each instant of time, the past n inputs and the past n outputs of the system (altogether $2n$) are fed into the neural network (Fig. 1). The network's output is compared with the next observation of the plant's output to yield the prediction error:

$$e(k+1) = y_p(k+1) - y_m(k+1) \tag{5}$$

The weights of the network are then adjusted using back-propagation training algorithm to minimize the sum of the squared errors:

$$E = \frac{1}{2}\sum_{i=1}^{Q}(e^{(i)}(k+1))^2 = \frac{1}{2}\sum_{i=1}^{Q}(y_p{}^{(i)}(k+1) - y_m{}^{(i)}(k+1))^2 \tag{6}$$

where Q is the number of training points in the training set. Note that since the learning is performed off-line, the speed of the learning algorithm will not be an issue.

Once the identification is achieved, two modes of operation are possible:

- Series-Parallel Mode: In this mode, the outputs of the actual system are used as inputs to the model (the network has no feedback). This scheme can be used only in conjunction with the plant. It can also generate only one step ahead prediction. The architecture is identical to the one used for identification (Fig. 1).

- Parallel Mode: If we assume that after a suitable training period the network gives a good representation of the plant (i.e. y_m ? y_p), the network output itself (and its delayed values) can be fed-back and used as a part of the network input (resulting in a recurrent network). In this way the network can be used independently of the plant. The parallel mode must be used if more than one step ahead prediction is required, or to achieve certain performance (as will be seen in Section 3).

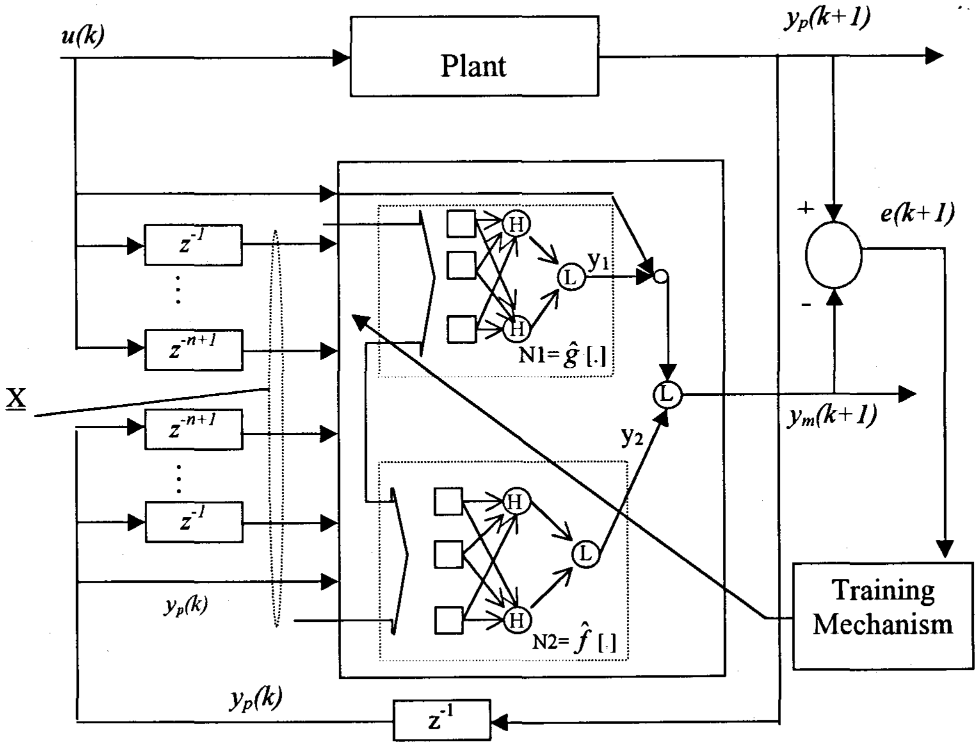


Fig. 1. NARMA-L2 identification model.

Where

$$\underline{X} = \left[u(k-1)\cdots u(k-n+1)\, y_p(k-n+1)\cdots y_p(k-1)\, y_p(k)\right]^T$$

Network Architecture

The multi-layered feed-forward neural network shown in Fig. 2 is composed of many interconnected processing units called neurons or nodes. As can be seen the net consists of three layers: an input layer (buffer layer), a single hidden layer with biases, and an unbiased linear output layer. The neurons in the input layer simply store the scaled input values. The hidden layer carries out two calculations. To explain these calculations, consider the general j'th neuron in the hidden layer shown in Fig. 3.

The inputs to this neuron consists of an *ni*-dimensional vector $\underline{X}$ (*ni* is the number of the input nodes) and a bias with its value as 1. Each of the inputs has a weight $v_{j,l}$ associated with it. The first calculation within the neuron consists of calculating the weighted sum $neth_j$ of the inputs as:

$$neth_j = \sum_{l=1}^{ni} v_{j,l} \cdot x_l + v_{j,ni+1} \tag{7}$$

Next the output of the neuron h_j is calculated as the hyperbolic tangent of $neth_j$ as:

$$h_j = \tanh(neth_j) \tag{8}$$

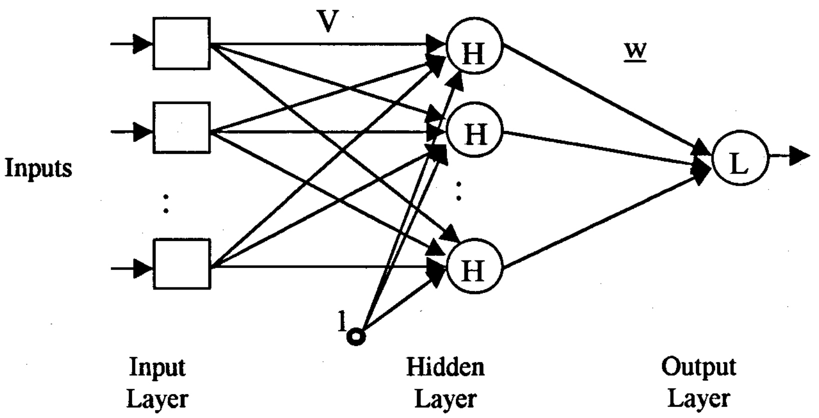
Once the outputs of the hidden layer are calculated, they are passed to the output layer. In the output layer, a single linear neuron is used to calculate the weighted sum *neto* of its inputs (the output of the hidden layer) as in (9).

$$neto = \sum_{j=1}^{nh} w_j \cdot h_j \tag{9}$$

where *nh* is the number of the hidden neurons (nodes) and w_j is the weight between the hidden neuron h_j and the output neuron .The single linear neuron, then, passes the sum *neto* through a linear function of slope 1 (another slope can be used to scale the output) as:

$$y = neto \tag{10}$$

Using such a network, each of the functions $g[.]$ and $f[.]$ in equation (4) can be approximated separately by $\hat{g}[.]$ and $\hat{f}[.]$ respectively, where $\underline{X}$ represents the input vector of the networks N1 and N2 (the argument of $\hat{g}[.]$ and $\hat{f}[.]$) as shown in Fig.1.



V: Weight matrix.
w: Weight vector.
L: Denotes linear node.
H: Denotes nonlinear node with hyperbolic tangent function

Fig. 2. A multi-layered feed-forward neural network

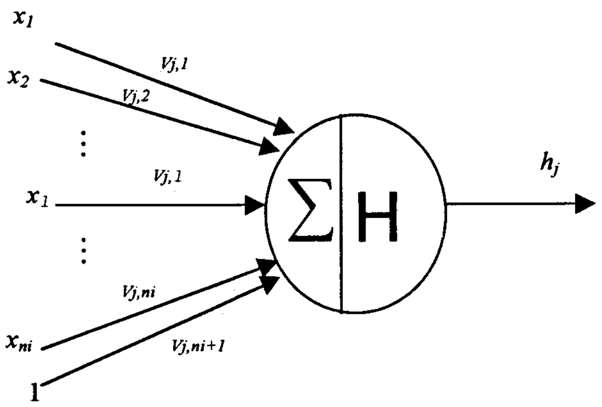


Fig. 3. Neuron j in the hidden layer

A suitably trained network has the ability to generalize; it provides the right outputs in response to inputs not appearing in the training set. An oversized network is expected to model the process with poor generalization capability, whereas the undersized network will not be able to represent the process dynamics faithfully. Although some results are available for predicting the optimal number of nonlinear hidden neurons [1], the approach proposed in [5,12] is easier to apply, this is because of the special structure of NARMA-L2 model. This approach attempts to obtain a model with the best generalization capability by generating the simplest model that is compatible with the training (learning) data set.

CONTROLLER DESIGN

The Concept of the IMC Strategy

The IMC strategy is now well known and has been widely applied to process control. For linear systems, which are described by transfer function models, the IMC represents a powerful controller design strategy [13]. The basic linear IMC structure is shown in Fig. 11, where $P(z)$, $M(z)$, $C(z)$, and $F(z)$ represents the plant, the plant model, the controller, and the robustness filter respectively.

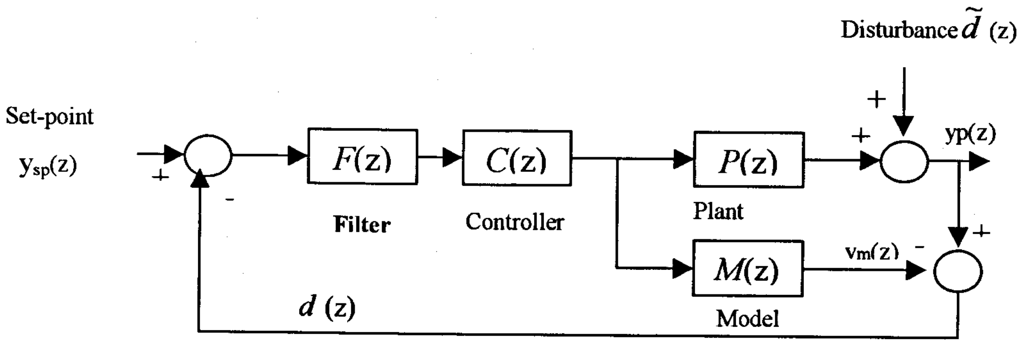


Fig. 4. Linear IMC structure

Initially, neglecting the robustness filter $F(z)$, and referring to the block diagram of Fig.3 the following relationships can be obtained:

$$\frac{y_p(z)}{\tilde{d}(z)} = \frac{1 - C(z)M(z)}{1 + C(z)\left(P(z) - M(z)\right)} \tag{11}$$

$$\frac{y_p(z)}{y_{sp}(z)} = \frac{C(z)P(z)}{1 + C(z)\left(P(z) - M(z)\right)} \tag{12}$$

From (11) and (12), it is apparent that if the model is perfect ($M(z)=P(z)$) then the necessary and sufficient conditions to ensure the stability of the overall structure is that both the controller and the plant are stable. Likewise, perfect modeling amounts to feed-forward control for set-point tracking, with feedback control for disturbance rejection, and hence, the IMC strategy provides a suitable design structure for the development of robust control techniques.

An important step towards a practically applicable nonlinear feedback controller design technique was achieved by proving that the attractive properties of the IMC, as studied in linear control system design, carry over to the general nonlinear case. Input-output operators were used in [14] to show that nonlinear IMC technique satisfies the same stability, perfect control, and zero-offset properties as linear IMC. Moreover, since disturbances were treated as modeling errors, no explicit techniques were proposed to incorporate measured disturbances in a feed-forward / feedback control scheme. For the general nonlinear IMC, its important characteristics are summarized in the following properties [14]:

- Property 1 (Stability): Assume that C and P are input-output stable and that a perfect model of the plant is available, i.e., $M=P$. Then the closed-loop system is input-output-stable.

- Property 2 (Perfect Control): Assume that the right inverse of the model operator M^{1} exists, that $C=M^{1}$, and that the closed-loop system is input-output-stable with this controller, then the control will be perfect, i.e., $y_p = y_{sp}$.

- Property 3 (Zero Offset): Assume that the right inverse of the steady-state model operator M_{∞}^{-1} exists, that the controller satisfies $C_{\infty} = M_{\infty}^{-1}$, and that the closed-loop system is input-output-stable with this controller, then offset free control is attained for asymptotically constant inputs.

Nonlinear IMC Implementation Using Neural Networks

Existing approaches

The idea of using neural networks for nonlinear IMC has been considered by a number of researchers [3,4,5,6]. The tendency was to employ a NARMA model of the plant. Moreover, it has been verified in [5] that offset-free performance cannot be achieved if the controller design is based on a series-parallel model of the plant and it is only the parallel model that can provide offset-free control. Thus for a NARMA model, the series-parallel model in (2) is replaced by its parallel form as in (13).

$$y_m(k+1) = \hat{F}[y_m(k), y_m(k-1),\ldots, y_m(k-n+1), u(k), u(k-1),\ldots, u(k-n+1)] \tag{13}$$

To obtain perfect control, the controller *C* must be chosen as the inverse of the internal model *M* (see Property 2), thus, a neural network may form a nonlinear approximation to the inverse of the previously trained neural plant model of (13) [4]. Assuming that the desired model output is $y_{des}(k)$, the general nonlinear inverse of the model (13) is defined in (14)

$$u(k) = \hat{\bar{G}}[y_{des}(k+1), y_m(k), y_m(k-1),..., y_m(k-n+1) \\ u(k-1),..., u(k-n+1)] \tag{14}$$

Assuming that the closed-loop system under neural IMC was stable (see Fig. 5), there would then be zero offset for asymptotically constant set-point y_{sp} and disturbances (which are treated here as modeling errors), if the control law inverts the steady-state gain of the model exactly.

Alternatively, using the plant model of (13), iterative methods can be used to calculate the inverse of the model at each sample time [4,5].

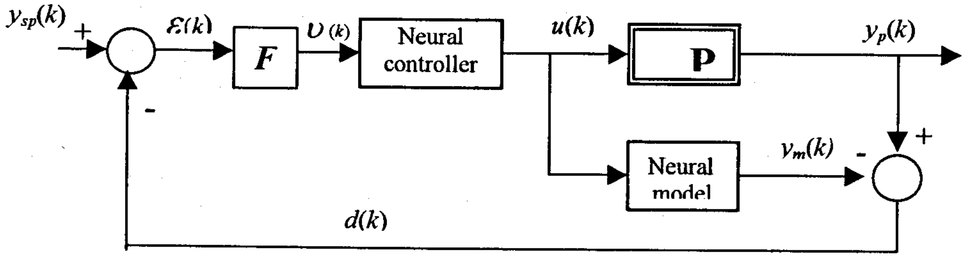


Fig. 5. IMC implemented using neural network model and its neural network inverse as a controller

Likewise, in [5] another implementation of the IMC was proposed, in which the control law was based on a model of the inverse process dynamics, which was identified directly without the need of the process model of (13). The main advantage of directly identifying the inverse dynamics of the process is that the control law can be implemented without on-line inversion of the neural plant model.

Finally in [6] a novel nonlinear IMC scheme was proposed, which utilized adaptive linear transfer functions for both plant model and controller, with the parameters of each determined at each sample time by linearization of a previously trained neural model of the plant.

Nonlinear IMC implementation based on an approximate NARMA model (NARMA-L2)

The neural network that is obtained from the identification structure of Fig. 1 is of dual use: It can be used as a feed-forward model of the plant as in (15), likewise if $\hat{g}[.]$ in (15) is sign definite in the operating region, the network can be used as the inverse of the plant model as in (16), where $y_m(k+1)$ is replaced by the desired model output $y_{des}(k+1)$

$$\begin{aligned} y_m(k+1) = {} & \hat{f}[y_p(k),\dots,y_p(k-n+1),u(k-1),\dots,u(k-n+1)] \\ & + \hat{g}[y_p(k),\dots,y_p(k-n+1),u(k-1),\dots,u(k-n+1)]\cdot u(k) \end{aligned} \tag{15}$$

$$u(k) = \frac{y_{des}(k+1) - \hat{f}[y_p(k),\dots,y_p(k-n+1),u(k),\dots,u(k-n+1)]}{\hat{g}[y_p(k),\dots,y_p(k-n+1),u(k),\dots,u(k-n+1)]} \tag{16}$$

The sign definite of $\hat{g}[.]$ in the operating region (the region of interest) ensures the uniqueness of the plant inverse at that operating region [10]. Now, by using (15) as the model of the plant and (16) as the inverse of the model (the controller), and after converting (15) and (16) to their parallel form, the IMC structure of Fig. 5 can be implemented directly. In [4] two neural networks were used to implement the same structure of Fig. 5. Moreover, if the model as in (15) is used, (16) will be the *exact* inverse of the model, which is very important to achieve perfect control (see Property 2). This is, of course, different from the approaches in [4] and [5] in which numerical methods were used on-line to enhance the accuracy of the *approximated* inverse of the model at each sample time.

For the sake of completeness, a time delay, *td*, which is between the input and the output of the model may be assumed, then the parallel form of (15) with this time delay can be written as in (17).

$$\begin{aligned} y_m(k+1) = {} & \hat{f}[y_m(k),\dots,y_m(k-n+1),u(k-td-1),\dots,u(k-td-n+1)] \\ & + \hat{g}[y_m(k),\dots,y_m(k-n+1),u(k-td-1),\dots,u(k-td-n+1)]\cdot u(k-td) \end{aligned} \tag{17}$$

The parallel model in (17) can be written in the equivalent prediction form:

$$\begin{aligned} y_m(k+td+1) = {} & \hat{f}[y_m(k+td),\dots,y_m(k+td-n+1),u(k-1),\dots,u(k-n+1)] \\ & + \hat{g}[y_m(k+td),\dots,y_m(k+td-n+1),u(k-1),\dots,u(k-n+1)]\cdot u(k) \end{aligned} \tag{18}$$

If $\hat{g}[.]$ in (18) is sign definite in the operating region, the exact inverse of the model will be readily computed for the current input $u(k)$ as in (19)

$$u(k) = \frac{y_m(k+td+1) - \hat{f}[y_m(k+td),\dots,y_m(k+td-n+1),u(k-1),\dots,u(k-n+1)]}{\hat{g}[y_m(k+td),\dots,y_m(k+td-n+1),u(k-1),\dots,u(k-n+1)]} \tag{19}$$

It is noticed that the future model output $y_m(k+td+1)$ is a function of $u(k)$ and thus cannot be used in the control law. However, an implementable controller can be obtained by replacing this output with the filter output υ (k) as follows:

$$y_m(k+td+1)=\upsilon(k) \tag{20}$$

Using (20), (19) can be written as:

$$u(k)=\frac{\upsilon(k)-\hat{f}[y_m(k+td),\dots,y_m(k+td-n+1),u(k-1),\dots,u(k-n+1)]}{\hat{g}[y_m(k+td),\dots,y_m(k+td-n+1),u(k-1),\dots,u(k-n+1)]} \tag{21}$$

The most important features that the presence of the filter imparts to the system are the following:

- Introducing robustness in the IMC structure in the face of modeling errors.
- Projecting the signal $\varepsilon(k)$ (see Fig. 5) into the appropriate input space for the controller.
- Smoothing out the noisy and/or rapidly changing signals in order to reduce the transient response of the IMC controller, in other words, to avoid controller ringing [15].

It has been shown in [14] that, a simple first order filter as in (22) gives satisfactory results in terms of robustness and performance.

$$F(z)=\frac{1-\alpha}{1-\alpha z^{-1}} \qquad \textit{where} \qquad (0\le\alpha<1) \tag{22}$$

Then from Fig. 5 we have:

$$\begin{gathered}\upsilon(z)=\frac{1-\alpha}{1-\alpha z^{-1}}(y_{sp}(z)-d(z))\\ \textit{where}\\ d(z)\equiv y_p(z)-y_m(z)\end{gathered} \tag{23}$$

For a perfect model $d(z)$=0, then by using (20) with $y_m(k+td+1)=y_p(k+td+1)$, the following closed-loop transfer function between the plant output and the set-point is obtained:

$$\frac{y_p(z)}{y_{sp}(z)} = z^{-(td+1)} \frac{1-\alpha}{1-\alpha z^{-1}} \tag{24}$$

The filter tuning parameter α can be tuned to provide a compromise between performance and robustness. For a perfect model, the effect of the tuning parameter α on the closed-loop response is particularly simple. Small values of α result in vigorous response, while large values cause sluggish response. The robustness of the proposed IMC strategy will be investigated in more detail in Section 4.

The filter equation in (23) can be written as:

$$\upsilon(k) = \upsilon(k-1) + (1-\alpha)\left\{[y_{sp}(k) - d(k)] - \upsilon(k-1)\right\} \tag{25}$$

Using equation (20), (25) can be rewritten as:

$$\upsilon(k) = \upsilon(k-1) + (1-\alpha)\left\{[y_{sp}(k) - d(k)] - y_m(k+td)\right\} \tag{26}$$

The signal $y_m(k+td)$ represents the prediction of the plant output *td* sampling periods ahead which can be obtained from the prediction model in (18) which *predicts* the delayed effect that the manipulated variable $u(k)$ will have on the plant output $y_p(k)$. This prediction is used to provide a dead-time compensator in the form of *Smith predictor* [1] and it is possible only if we have a good model of the plant. In what follows *td* is assumed to be perfectly known. Now, by using (20) and (23), (26) can be written as:

$$\upsilon(k) = \alpha \cdot \upsilon(k-1) + (1-\alpha)\cdot \upsilon(k-td-1) + (1-\alpha)\{y_{sp}(k) - y_p(k) \tag{27}$$

And by taking the z-transform of (27) we get:

$$\upsilon(z) = \frac{1-\alpha}{1-\alpha z^{-1} - (1-\alpha)z^{-td-1}}\{y_{sp}(z) - y_p(z)\} \tag{28}$$

Using (28) the nonlinear IMC can be reconstructed in a classical feedback structure (see Fig. 6). In fact any conventional feedback controller can be restructured to yield IMC. Furthermore, any IMC can be put into the conventional feedback form [13].

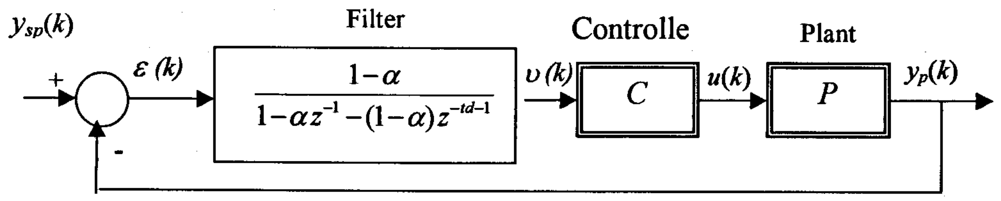


Fig. 6. Nonlinear IMC in its conventional feedback form

SIMULATION EXAMPLES

Example 1

The plant to be controlled is covered by the difference equation:

$$y_p(k+1)=\frac{-0.9y_p(k)+u(k)}{1+y_p^{\,2}(k)} \tag{29}$$

This plant has been adopted from [16]. Fig. 7 shows the open-loop step response with $u(k)$=0.8. The step response of the plant is very oscillatory for the low amplitude input and shows limit cycle oscillation for u(k) > 0.6.

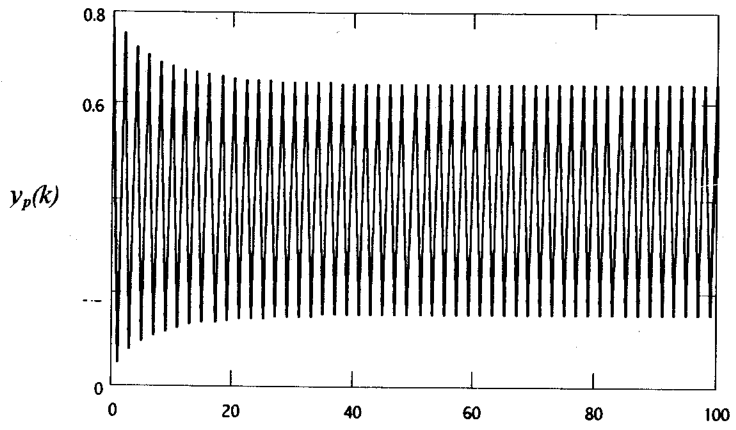


Fig. 7. Plant response with step input by amplitude 0.8

To identify the plant dynamics, a series-parallel identification structure as that in Fig. 1 has been used. The model is described by:

$$y_m(k+1) = N1[y_p(k)] \cdot u(k) + N2[y_p(k)] \qquad (30)$$

where $N1[.]$ and $N2[.]$ are multi-layered neural networks which approximate $g[.]$ and $f\ [.]$ of equation (4) respectively. The initial guess of the number of the hidden nodes was 3 for each network.

Using a random input sequence $\{u\ (k)\}$ with $|\,u(k)\,| \leq 1$ a training set of 300 patterns has been developed. During the training phase, the training set has been presented to the network many times. A training event corresponding to a single pass over the entire training set is called a training epoch or training cycle. However, for this example after 1600 epochs the Average System Error (*ASE*) computed for the latest epoch, which is described by (31), was 0.1×10^{-5}.

$$ASE = \frac{1}{2Q}\sum_{i=1}^{Q}(y_p{}^{(i)}(k+1) - y_m{}^{(i)}(k+1))^2 \qquad (31)$$

where Q = total number of patterns. After that, the training has been continued up to 5000 epochs, but failed to decrease the *ASE* any further. Depending on this perfect model, the chosen architecture has been adopted (3 hidden nodes for each of *N*1 and *N*2) and a parallel model of the form:

$$y_m(k+1) = N1[y_m(k)] \cdot u(k) + N2[y_m(k)] \qquad (32)$$

has been developed from the series-parallel model of (30) simply by replacing the plant output $y_p(k)$ by the model output $y_m\ (k)$. Fig. 8 compares the time response of the parallel model of equation (32) with the actual plant output for the input $u(k)$ generated as the sum of two sinusoids, as in (33).

$$u(k) = 0.5\sin(\frac{2\pi k}{25}) + 0.5\sin(\frac{2\pi k}{10}) \qquad (33)$$

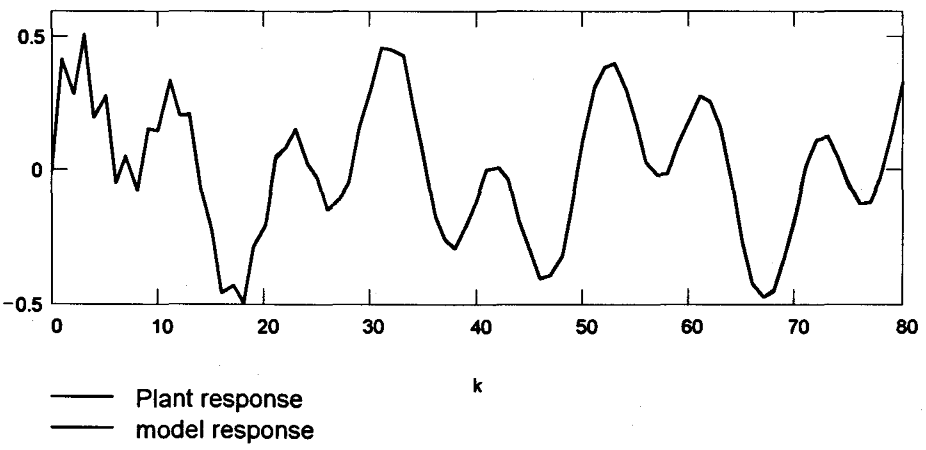


Fig.8. The response of the plant (y_p) and of the parallel NARMA-L2 model (y_m)

Although the input signal used differs significantly from that used to generate the training data, the parallel NARMA-L2 model of equation (32) performs excellently in such a way that the output of the plant as well as the output of the model plotted in Fig. 8 are seen to be indistinguishable. Fig. 9 shows a plot of the coefficient of $u(k)$ ($\hat{g}[.]$) for the parallel NARMA-L2 model as a function of time with values computed using the corresponding network $N1[y_m(k)]$ when a random input sequence $\{u(k)\}$ with $|u(k)| \leq 1$ has been applied to the model. As shown in Fig. 9, $\hat{g}[.]$ is sign definite in the region of interest. This means that the plant model is invertable, or in other words, the model output $y_m(k+1)$ is monotonic with respect to $u(k)$. Since the plant is invertable, the IMC strategy using the approximate NARMA-L2 model can be applied directly.

To make some study on the role of the tuning parameter α, Fig. 10 (a and b) shows the plant responses $y_p(k)$ superimposed on the reference signal (set-point) $y_{sp}(k)$ for two different values of α. For $\alpha = 0.8$ the response $y_p(k)$ was sluggish, see Fig. 10a, it can be seen that the rate of change of the response is slow. Faster response can be obtained by decreasing the value of α as shown in Fig. 10.b, where $\alpha = 0.34$. This will ensure excellent set-point tracking without overshoot and with zero offset.

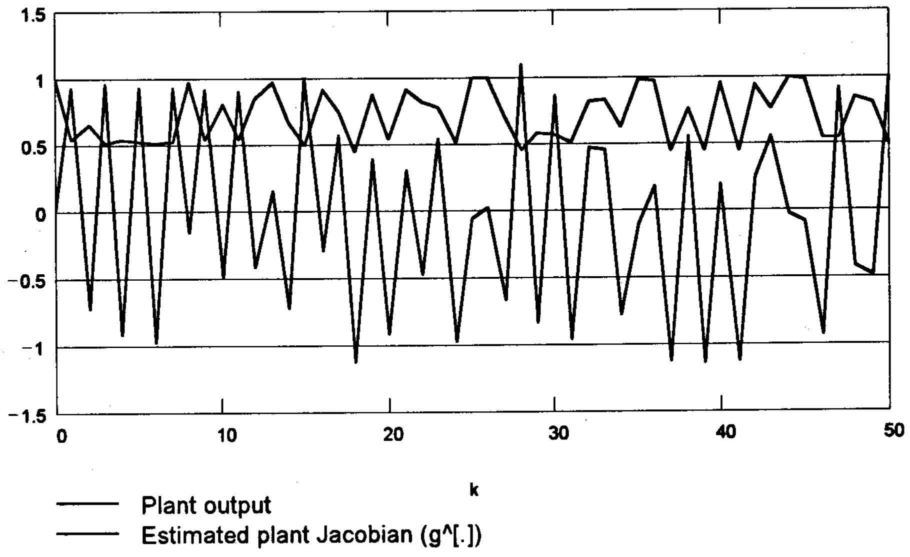


Fig. 9. Plots of the plant output and the estimated plant Jacobian

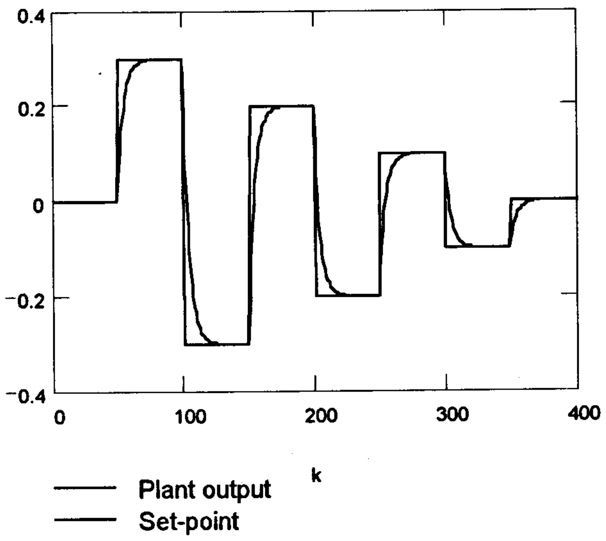


Fig. 10a

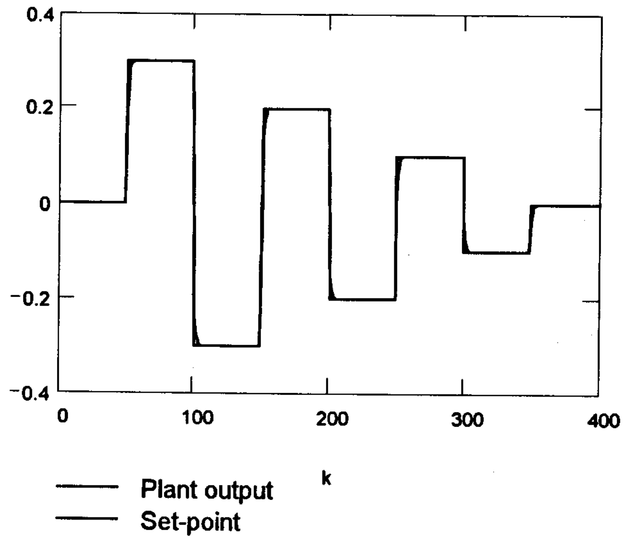


Fig. 10b

Fig. 10. Example 1: Set-point tracking for different values of α: (a) α=0.8, and (b) α=0.34.αα

Example 2

This example shows the performance of the proposed IMC on the same plant of example 1 but with time delay of $2Ts$ ($td = 2$), where Ts is the sampling time. The plant is represented by:

$$y_p(k+1) = \frac{-0.9y_p(k) + u(k-td)}{1 + y_p^2(k)} \tag{34}$$

The network used to model the plant in Example.1 can be used here again simply by replacing *u(k)*, of Example.1, by *u(k-td)*, this is because the nonlinear mapping between the inputs and the output of the network is still the same.

The closed-loop control structure will be identical to that of Example.1 except the robustness filter. Here the filter will have the following pulse transfer function of the form:

$$\upsilon(z) = \frac{1-\alpha}{1-\alpha z^{-1} - (1-\alpha)z^{-td-1}}\left[y_{sp}(z) - y_p(z)\right] \tag{35}$$

where $td = 2$.

Fig. 11 shows the time response of the plant under the proposed IMC strategy superimposed on the reference set-point signal for $\alpha = 0.34$.

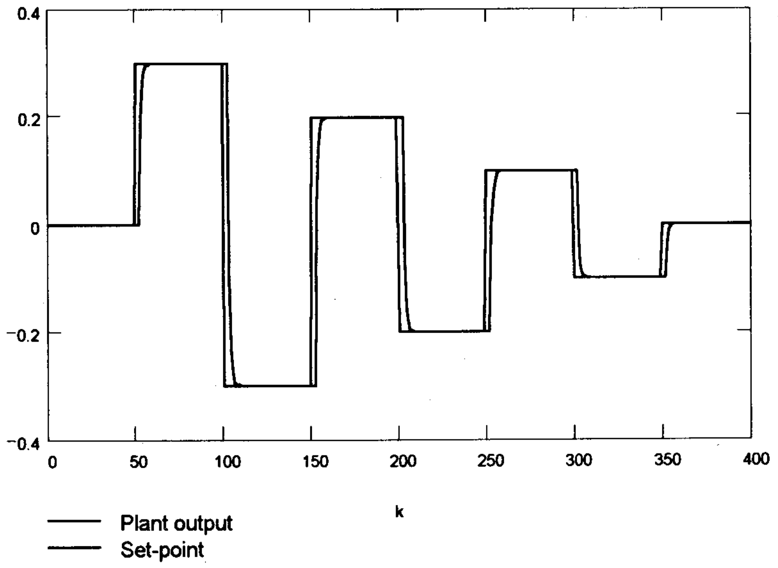


Fig. 11. Set-point tracking with α=0.34

Example 3

Consider a Continuous Stirred Tank Reactor (CSTR) in which an irreversible exothermic reaction A $\rightarrow$ B takes place. The heat of reaction is removed by a coolant medium that flows through a jacket around the reactor (Fig. 12). As is known from the analysis of the CSTR system, the CSTR is at steady state when the heat produced by the reaction equals to the heat removed by the coolant. This requirement yields three steady states, two stable and one unstable.

The process model consists of two nonlinear ordinary differential equations [5]:

$$C'_a(t) = \frac{q}{Vol}\left(C_{af} - C_a(t)\right) - k_0 \cdot C_a(t) \cdot e^{\left(-\frac{E}{R \cdot T(t)}\right)}$$

$$T'(t)=\frac{q}{Vol}\left(T_f-T(t)\right)+\frac{(-\Delta H)\cdot k_0\cdot C_a(t)}{\rho\cdot C_p}e^{\left(-\frac{E}{R\cdot T(t)}\right)}+$$
$$\frac{\rho_c\cdot C_{pc}}{\rho\cdot C_p\cdot Vol}\cdot q_c(t)\left[1-e^{\left(-\frac{h_a}{q_c(t)\cdot\rho_c\cdot C_{pc}}\right)}\right]\cdot\left(T_{cf}-T(t)\right) \quad (36)$$

where $C_a(t)$ is the product (effluent) concentration of component A, $T(t)$ is the reactor temperature, q is the feed flow-rate (assumed to be constant),and $q_c(t)$ is the coolant flow-rate. The remaining model parameters are defined in nominal operating conditions as shown in the Appendix . The operating point in the Appendix corresponds to the lower stable state.

The objective is to control $C_a(t)$, this can be done by introducing a coolant flow-rate $q_c(t)$ (the manipulated variable), the temperature can be varied and hence the product concentration controlled [5,15]. To illustrate the problems involved in controlling the concentration $C_a(t)$, a traditional PID controller was first used. The digital approximation for the PID controller is given in (37).

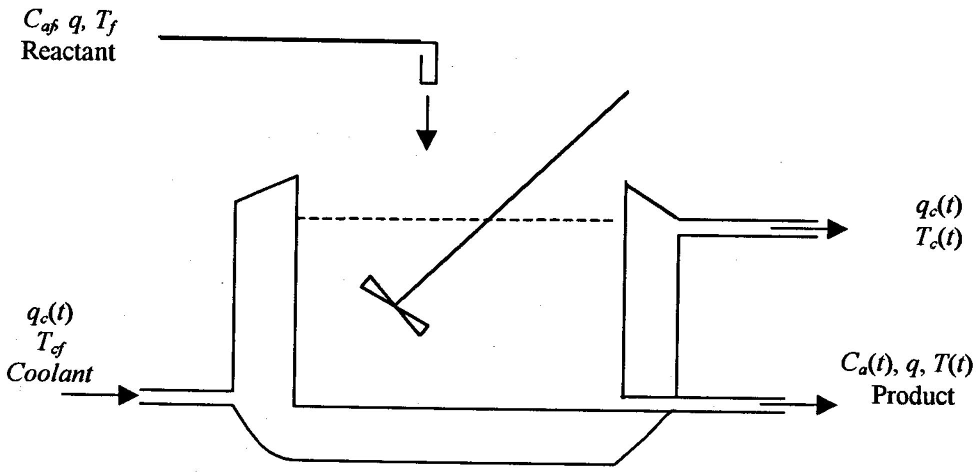


Fig. 12. CSTR with cooling jacket

$$q_c(k) = K_c\left\{\varepsilon(k) + \frac{Ts}{\tau i}\left(\sum_{l=0}^{k-1}\varepsilon(l) + \varepsilon(k)\right) + \frac{\tau d}{Ts}\left(\varepsilon(k) - \varepsilon(k-1)\right)\right\} + q_{ss} \tag{37}$$

where K_c, τi, τd, and Ts are the gain, integral time, derivative time, and sampling time respectively. $\varepsilon(k)$ is the performance error (i.e., $\varepsilon(k) = y_{sp}(k) - C_a(k)$) and finally q_{ss} is the controller's bias signal (i.e., its actuating signal when $\varepsilon = 0$), and for the CSTR it is the coolant flow-rate at steady state $q_{ss} = 103.41$ l/min.

The tuning method of *Cohen and Coon* [15] was used to tune the PID controller. The plant response under the PID controller for set-point tracking was as shown in Fig. 13. As expected, the PID controller performs well in a limited range, this range is approximately between 0.06 mol/l and 0.08 mol/l where the controller was initially tuned. But the performance deteriorates outside this range. This reflects the nature of nonlinearity present within the plant, with the degree of damping varying considerably over the set-point range. Likewise, it is a good indication that the conventional linear controllers cannot cope with complexities in the CSTR process.

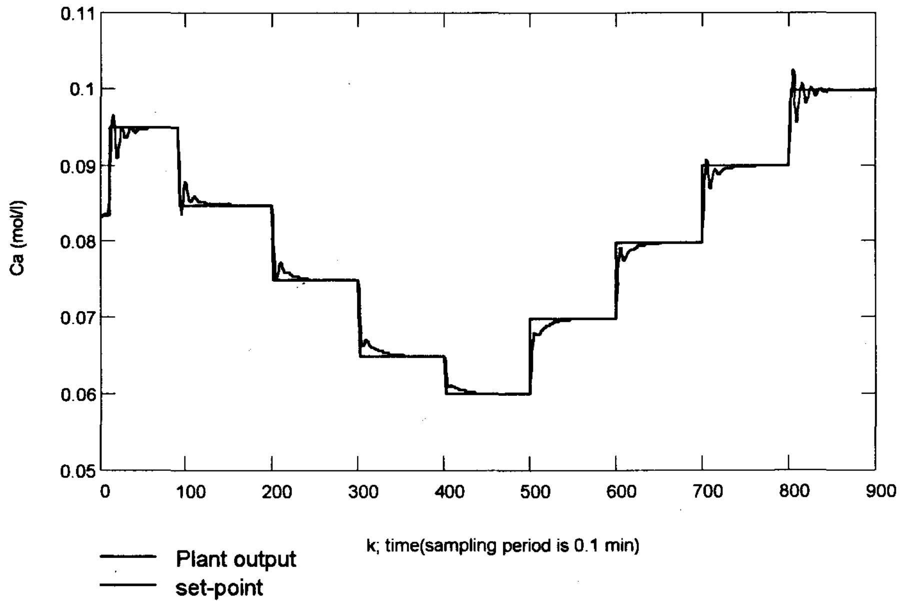


Fig. 13. The set-point tracking performance for the PID controller with the following tuning parameters: K_c =600 l^2/(min mol), τi =0.946 min, and τd=0.0375 min.

As the neural networks are trained off-line, a selection of input-output training patterns 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 cover the specified operating region. The training data has been generated by forcing the dynamic equations with a series of arbitrary step input changes with sampling period of 0.1 min. The signals entering to or emitting from the network (all the elements of the training data) have been normalized to lie within -1 and +1 in order to overcome numerical problems that could possibly arise otherwise.

Using the identification structure of Fig. 1 with $n = 3$ (plant order assumed to be known), a series-parallel NARMA-L2 model of the plant that has the form of (38):

$$\begin{aligned} C_a(k+1) = {} & N1[C_a(k), C_a(k-1), C_a(k-2), q_c(k-1), q_c(k-2)] \cdot q_c(k) \\ & + N2[C_a(k), C_a(k-1), C_a(k-2), q_c(k-1), q_c(k-2)] \end{aligned} \tag{38}$$

has to be identified. Many experiments have been made in order to obtain the optimum structure of the neural network that can fairly approximate the plant dynamics. It has been found that the neural network with 6 hidden nodes (for each of *N1* and *N2*) gives fairly good generalization capabilities (see Fig. 14).

The first step in the design of the IMC is to check the existence of the inverse of the plant model. This can be done easily by checking the sign of the plant Jacobian in the region of interest. Fig. 15 shows the coefficient of $q_c(k)$ ($N1[.]$ = plant Jacobian) as a function of time for the parallel NARMA-L2 model of (39) where $q_c(k)$ is the series of step changes used in generating the training data.

$$\begin{aligned} \hat{C}_a(k+1) = {} & N1\left[\hat{C}_a(k), \hat{C}_a(k-1), \hat{C}_a(k-2), q_c(k-1), q_c(k-2)\right] \cdot q_c(k) \\ & + N2\left[\hat{C}_a(k), \hat{C}_a(k-1), \hat{C}_a(k-2), q_c(k-1), q_c(k-2)\right] \end{aligned} \tag{39}$$

Since the plant Jacobian ($N1[.]$) is sign definite in the region of interest, the plant model is invertable and a controller of the form of (40) can be implemented.

$$q_c(k) = \frac{\upsilon(k) - N2[\hat{C}_a(k), \hat{C}_a(k-1), \hat{C}_a(k-2), q_c(k-1), q_c(k-2)]}{N1[\hat{C}_a(k), \hat{C}_a(k-1), \hat{C}_a(k-2), q_c(k-1), q_c(k-2)]} \tag{40}$$

wherev $\upsilon(k)$ is the output of the robustness filter.

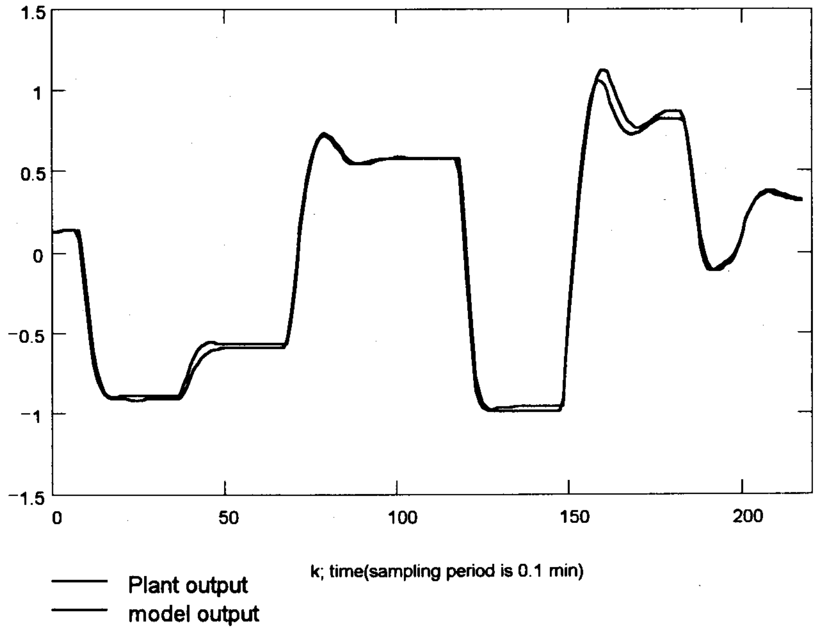


Fig. 14.The performance of the parallel model with the testing set

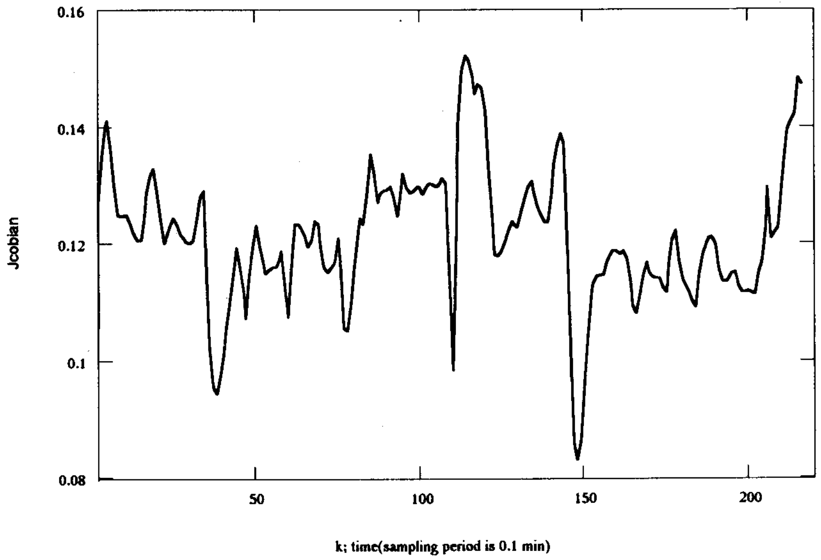
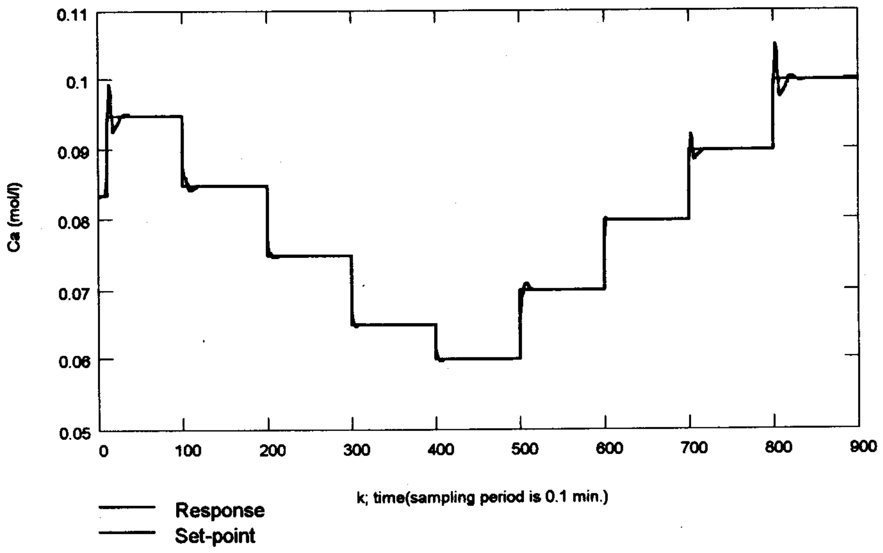


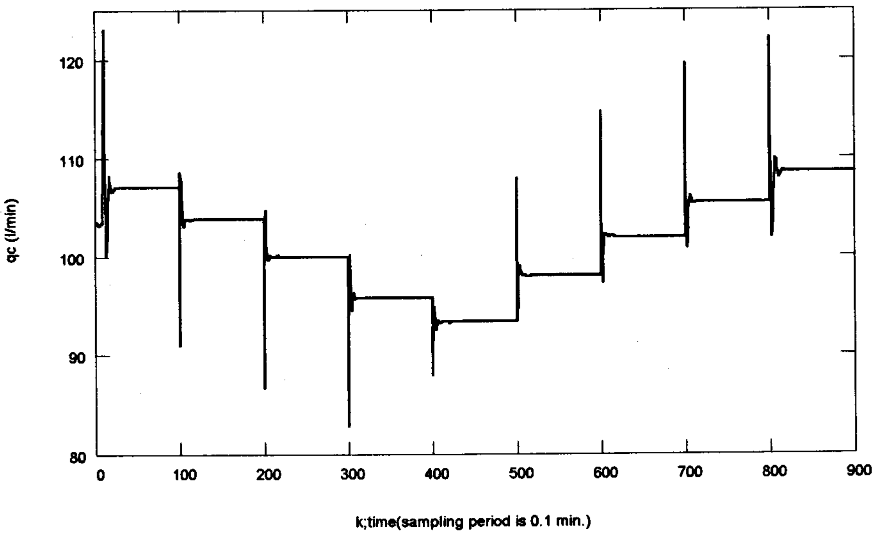
Fig. 15. The estimated plant Jacobian

Each experiment to be made has been conducted over 900 samples using a sampling time of 0.1 min., which resulted in 90 min. duration. The filter was initially tuned with $\alpha = 0.5$. Fig. 16.a shows the plant response and Fig. 16.b shows the corresponding control signal. As can be seen from Fig. 16.a, large overshoots have occurred for set-points greater than 0.09 mol/l. however, the response for set-points less than 0.09 mol/l is very good. It can also be observed that for each step change in the set-point, sharp fluctuations have occurred in the controller output (see Fig. 16.b), which results in very fast plant response. It can be observed, also, that the control signals settle generally within a few samples. It is important to note that prolonged adverse fluctuations of the control signals could damage the actuator. The large overshoots observed in Fig. 16.a were caused by the modeling errors. Fortunately the effect of mismodeling can be decreased by choosing relatively large α for the robustness filter, or/and by adding a filter at the set-point $y_{sp}(k)$. To illustrate the effect of a first order set-point filter (with a tuning parameter γ) on the plant response and more importantly on the controller response, compare Fig. 16 with Fig. 17a. Fig. 17b shows the plant response and the corresponding control signal for $\gamma = 0.6$ and $\alpha = 0.5$.

It can be observed that the controller can track almost precisely over the nonlinear operating region, moreover, lower control moves (as compared with the pervious case) are required. From these Figures (10 and 11), it is obvious that decreasing the pole of the set-point filter (γ) causes large control moves and consequently faster set-point tracking. Generally speaking, excessive control valve (the actuator for the CSTR) movement is unacceptable in industrial practice, and smooth controller outputs prolong actuator life. Thus tuning γ depends mainly on practical limitations.

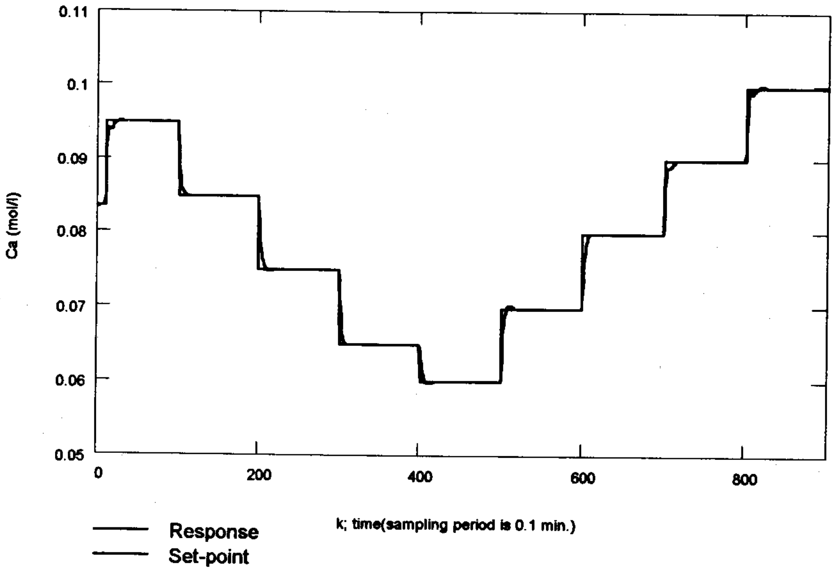


(16a)

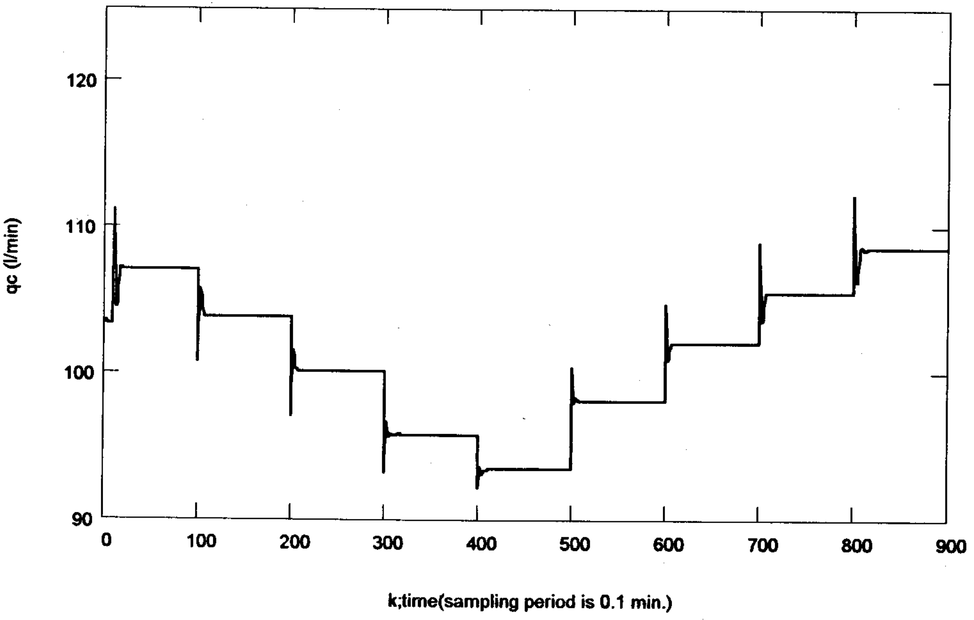


(16b)

Fig. 16. Example 3: (a) the set-point tracking performance, (b) the corresponding control signal for $\alpha = 0.5$



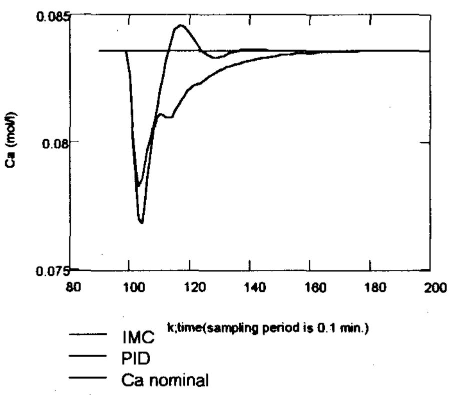
(17a)



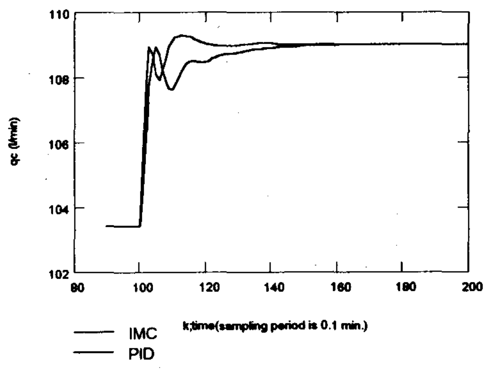
(17b)

Fig. 17 (a) the set-point tracking performance, (b) the corresponding control signal for $\alpha = 0.5$ and $\gamma = 0.6$

The second set of experiments has been carried out with the purpose of studying the ability of the proposed controller in rejecting unmeasured disturbances (for CSTR the most common disturbances are in C_{af}, q, T_f). In order to evaluate the ability of the proposed controller in rejecting unmeasured disturbances, it is necessary to compare it with others used in existing approaches, thus, the PID controller was chosen for this purpose and to ensure identical comparison between the two controllers, the same design and controller parameters were used for the IMC and the PID controllers as in the first set of experiments. The effects of the disturbance on the two controllers are shown as in Fig. 18, when an artificial step change in the feed temperature T_f of value 5K was added to the process at the 100 sampling instant. It can be observed that the proposed neural controller performs well with the ability to recover quickly from the effects of the positive step change in T_f, also it can be observed that the control moves for the two controllers are similar. Improved disturbance rejection can be obtained by decreasing the tuning parameter α of the robustness filter. From the above discussion, it is clear that there is a trade off between robustness and performance, if we want good performance, the robustness must be sacrificed, and vice versa.



(18a)



(18b)

Fig. 18. Disturbance rejection performance under IMC controller with $\alpha = 0.5$ and under PID controller with K_c =600 l^2/(min mol), τi =0.946 min, and τd=0.0375 min.

CONCLUSIONS

An IMC strategy based on neural NARMA-L2 model has been designed and applied successfully to a number of nonlinear systems with and without time delay. The proposed strategy has shown the ability to out-perform conventional PID controller, for set-point tracking and disturbance rejection, as it was clear when applied to the CSTR plant.

Under IMC controller, it is easy for an operator to determine the bandwidth suitable for a particular application by choosing an appropriate value for the robustness filter tuning parameter (α). A low bandwidth gives a slow response, small control signals, and low sensitivity to modeling errors and disturbances. A high bandwidth gives a fast response. However, the control action will be large and the system will be sensitive to modeling errors and disturbances.

Since the proposed controller is fixed, a specialized neural controller chip may be trained as a stand-alone controller.

If an adaptive controller is desired, with simultaneous on-line training of the feed-forward model *M*, this control procedure may be made fully adaptive, without the problems associated with training a controller embedded within a control loop.

Using a feed-forward model of the plant in the form of NARMA-L2, an exact inverse of the model can be computed directly without the need to numerical methods that are used on-line in each sample time to compute an approximate model inverse. It should be noted that the exact inverse is very important to achieve offset-free performance.

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.

REFERENCES

1. **Kanjilal, P. P., 1995.** Adaptive prediction and predictive control. England: Peter Peregrinus Ltd.

2. **Omatu, S., Khalid, M., and Yusof, R., 1995.** Neuro-Control and its Applications. London: Springer-Velag.

3. **Bhat, N. and McAvoy, T. J., 1990.** Use of neural nets for dynamic modeling and control of chemical process systems. Comput. Chem. Eng. , Vol. 14, No.(4-5), pp. 573-583.

4. **Hunt, K. J. and Sbarbaro, D., 1991.** Neural networks for nonlinear internal model control. IEE Proceedings-D, Vol. 138, No. 5, pp. 431-438.

5. **Nahas, E. P., Henson, M. A., and Seborg, D. E., 1992.** Nonlinear internal model control strategy for neural networks. Computers Chem. Eng., Vol. 16, No. 12, pp. 1039-1057.

6. **Lightbody, G. and Irwin, G. W., 1997.** Nonlinear control structures based on embedded neural systems models. IEEE Trans. Neural Networks, Vol. 8, No. 3, pp. 533-567.

7. **Hunt, K. J., Sbarbaro, D., Zbikowski, R., and Gawthrop, P. J., 1992.** Neural networks for control systems-A survey. Automatica, Vol. 28, No. 6, pp. 1083-1112.

8. **Levin, V. and Narendra, K. S., 1996.** Control of nonlinear dynamical systems using neural networks__Part II: Observability, Identification, and Control. IEEE Trans. Neural Networks, Vol. 7, No. 1, pp. 30-42.

9. **Chen, S. and Billings, S. A., 1989**. Representations of nonlinear systems: the NARMAX model. Int. J. Contr. Vol.49, No. 3, pp. 1013-1032.

10. **Narendra, K. S. and Mukhopadhyay, S., 1997**. Adaptive control using neural networks and approximate models. IEEE Trans. Neural Networks, Vol. 8, No. 3, pp. 475-485.

11. **Chen, F. C., 1990**. Back-propagation neural networks for nonlinear self-tuning adaptive control. IEEE Control Systems Magazine, Vol. 10, No. 3, pp. 44-48.

12. **Knight, K., 1990**. Connectionist ideas and algorithms. Communications of the ACM., Vol. 33, No. 11, pp. 59-74.

13. **Gracia, E. and Morari, M., 1982**. Internal model control. 1: A unifying review and some new results. Ind. Eng. Chem. Process Des. Dev., Vol. 21, No. 2, pp. 308-323.

14. **Economou, G., Morari, M., and Palsson, B. O., 1986**. Internal model control 5: Extension to nonlinear systems. Ind. Eng. Chem. Process Des. Dev. Vol. 25, No. 2, pp. 403-411.

15. **Stephanopoulos, G., 1984**. Chemical Process Control: An Introduction to Theory and Practice. Englewood Cliffs, N.J.: Prentice-Hall.

16. **Ahmed, M. S. and Tasadduq, I. A., 1994**. Neural-net controller for nonlinear plants: design approach through linearisation. IEE Proc.-Control Theory Appl., Vol. 141, No. 5, pp. 315-322.

APPENDIX

Nominal CSTR Operating Conditions

Parameter	***Description***	**Nominal Value**
q	Process flow-rate	$100\ l\ min^{-1}$
C_{af}	Inlet feed concentration	$1\ mol\ l^{-1}$
T_f	Feed temperature	350 K
T_{cf}	Inlet coolant temperature	350 K
Vol	Reactor volume	100 l
h_a	Heat transfer coefficient	$7\times10^5\ cal\ min^{-1}.\ K^{-1}$
k_o	Reaction rate constant	$7.2\times10^{10}\ min^{-1}$
E/R	Activation energy	9.95×10^3 K
ΔH	Heat of reaction	$-2\times10^5\ cal\ mol^{-1}$
ρ, ρ_c	Liquid densities	$1000\ g\ l^{-1}$
C_p, C_{pc}	Specific heats	$1\ cal\ g^{-1}.\ K^{-1}$
q_c	Coolant flow-rate	$103.41\ l.\ min^{-1}$
T	Reactor temperature	440.2 K
C_a	Product concentration	$8.36\times10^{-2}\ mol\ l^{-1}$

Note:

$$\text{Process time constant} = \frac{vol}{q} = \frac{100l}{100l \cdot \min^{-1}} = 1\,\text{min}$$