akistan Journal of Information and Technology 1(2): 173-179, 2002

Artificial Neural Network Based Nonlinear Model Predictive Control Strategy

Ebrahim Abdulla Al-Gallaf
Department of Electrical and Electronics Engineering, College of Engineering
University of Bahrain, P. O. Box: 13184, Kingdom of Bahrain

Abstract: This paper presents the application of neural-network based Model Predictive Control (MPC) scheme to control a nonlinear liquid-level system with interaction. We have employed a feed forward neural network as the model in MPC-algorithm and compared it with a model obtained by the conventional statistical identification method (Least Square Method which identifies the linearized model). A model predictive controller synthesis using a neural network based model were found to have better performance in terms of convergence of the system to the desired settling set points than using the statistical model.

Key Words : Model Predictive Control, Artificial Neural Networks, Modeling

Introduction :

Model Predictive Control (MPC) concept has been widely accepted in industrial applications and extensively studied by academia, Zhan and Ishida (1997). The main reasons for such popularity of MPC strategies are the intuitiveness, multivariable interaction handling, process delays and the explicit constraint handling, J. Zhan and M. Ishida(1997). Several versions of MPC techniques are Dynamic Matrix Control (DMC) (Cutler and Ramaker, 1980), Model Algorithmic Control (MAC) (Richalet *et al.* (1978) and Internal Model Control (IMC) Garcia and Morari (1982). Although the above techniques differ from each other in some details, they are fundamentally the same. An important advantage of these control schemes is the ability to handle constraints of actuated variables and internal variables. In most applications of model predictive techniques, a model is used to predict the process behavior over the horizon of interest, A. Draeger *et al.* (1995).

As most real processes show nonlinear behavior, this is the most expensive part of the realization of a nonlinear predictive control scheme is the derivation of the mathematical model. In many cases it is even impossible to obtain a suitable physically founded process model due to the complexity of the underlying processes or the lack of knowledge of critical parameters of the models. Draeger *et al.* (1995). Even if the nonlinear model is available, the computational requirements are expected to be very high Garcia *et al.* (1982), especially for nonlinear MIMO processes. In the past decade there has been a strong resurgence in the field of artificial neural networks involving researchers from many diverse disciplines. This renewed interest in neural networks is fueled by new network topologies, improved learning algorithms, and many actual industrial applications, WU and Harris,(4) Obviously, this is a merge science which is related to the system identification and signal processing, therefore, it is not surprising that there have been many applications of neural networks to the problem of non-linear system identification and prediction. Neural Network Model Predictive Control (NNMPC) is typically a straightforward application of neural networks to nonlinear control. When a neural network is combined with MPC approach, it is used as a forward process model for the prediction of process output. Such neural network models may be derived from measured input/output data of the plant. In many practical cases, however, conventional controlle-s (P,PI,PID-controllers) are already in use at the plant which stabilize the plant and provide basic and sometimes sluggish control. These measurements of input/output variables of the plant operated with the linear controller may provide very good training data for the neural network. As a relatively well-known example, we consider a liquid-level system with interaction, this plant exhibits a nonlinear behavior (Ogata), so that nonlinear prediction using ANN model has to be employed.

Liquid-Level System:

In analyzing systems involving fluid flow, we find it necessary to divide flow regimes into laminar flow and turbulent flow, according to the magnitude of the Reynolds number. If the Reynolds number is greater than about 3000 to 4000, then the flow is turbulent (Ogata). The flow is laminar if the Reynolds number is less than about 2000. In the laminar case, fluid flow occurs in streamlines with no turbulence. Systems involving turbulent flow often have to be represented by nonlinear differential equation, while systems involving laminar flow may be represented by linear differential equations. Industrial processes often involve flow of liquids through connecting pipes and tanks. The flow in such processes is often turbulent and not laminar. The liquid-level system that we want to control consists of two tanks with two inlets and two outlets. In the system $\tilde{Q}_1$ and $\tilde{Q}_2$ are steady-state inflow rates and $\tilde{H}_1$ and $\tilde{H}_2$ are steady-state heads. The quantities $q_{i1}, q_{i2}, h_1, h_2, q_1, q_0$ are considered small. Since the flow through the restriction

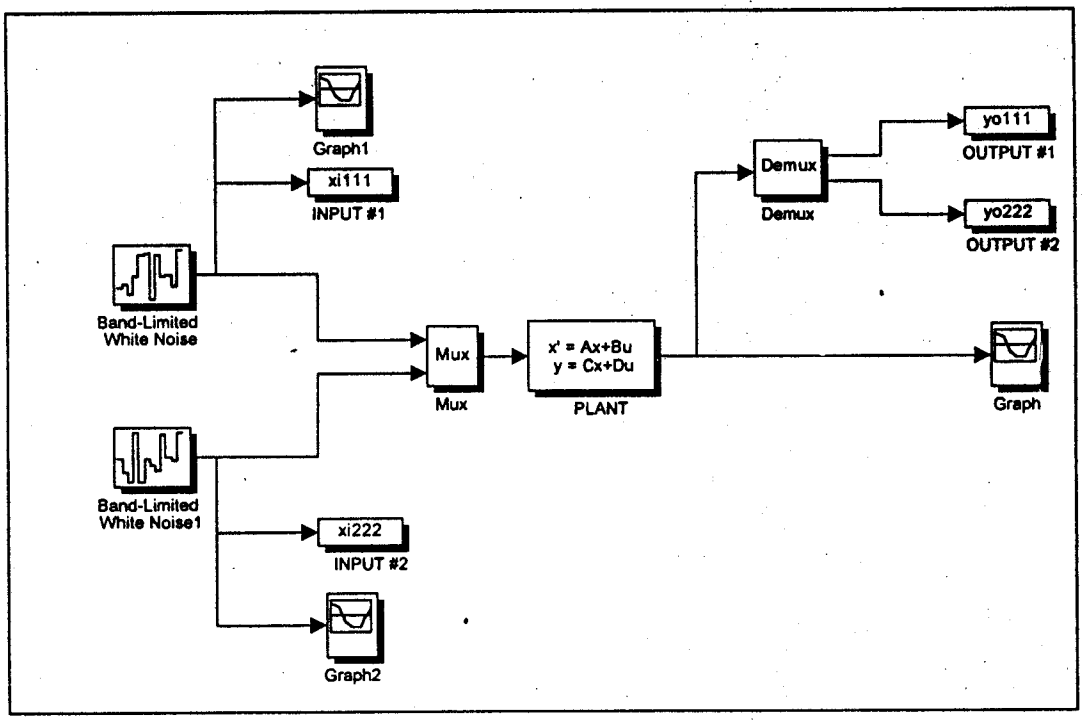

Fig. 1a: Process of Input-Output Data Set Generation

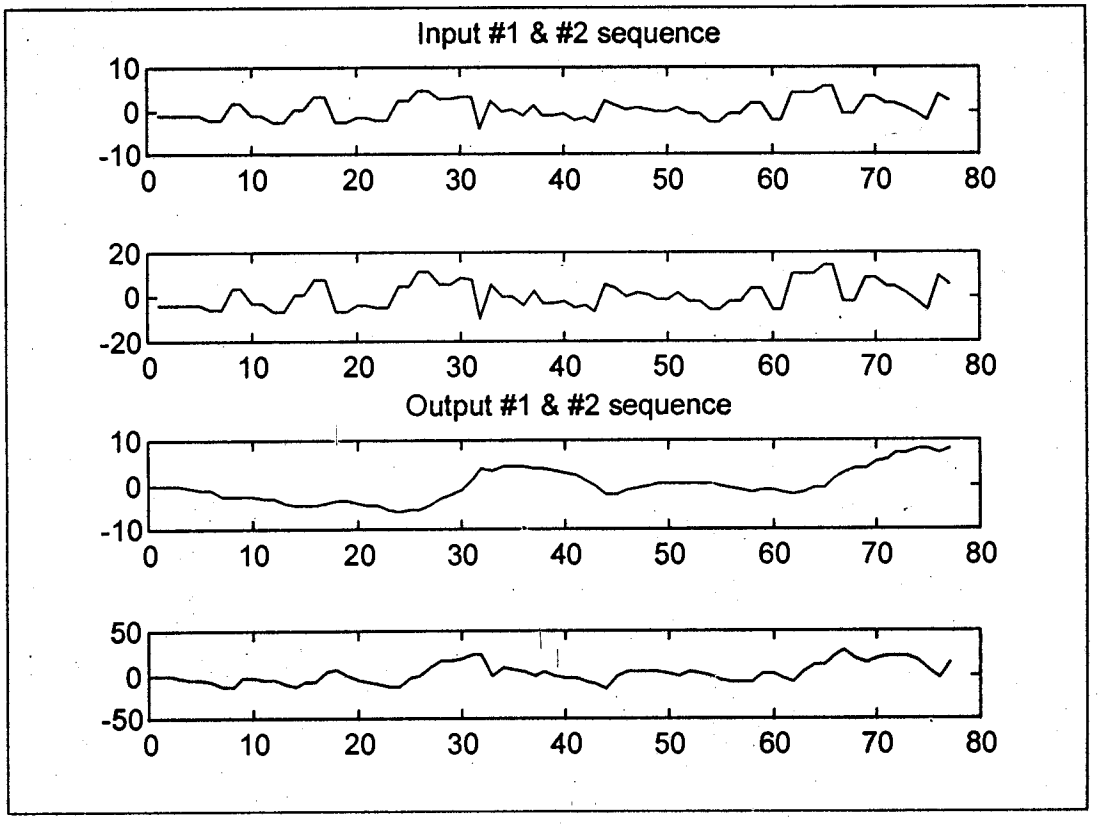

Fig. 1b: Multivariable Input-Output Data Set

is turbulent, the steady-state flow rate and the resistance R_t for turbulent flow are given by(Ogata):

$$Q = K\sqrt{H}$$
$$R_t = \frac{dH}{dQ} \qquad (1)$$

where

Q = steady-state liquid flow rate, m^3/sec

K = coefficient, $m^{2.5}/\text{sec}$

H = steady-state head,

R_t = resistance for liquid flow.

The value of the turbulent-flow resistance R_t depends on the difference of the liquid levels of the two tanks and the change in flow rate. The process flow rate versus the head. The capacitance C of a tank is defined to be the change in quantity of stored liquid necessary to cause a unit change in the potential (head). The potential is the quantity that indicates the energy level of the system. An input-output data set is obtained by randomly changing the manipulated variables as shown in Fig. 1-a.

The data set representing the input-output of the plant is shown in Fig. 1-b. Such input-output set of data are used to train a three layer neural network. For some classes of nonlinear systems, if a linear model at a fixed operating point is used for a local system representation, a global system representation can be realized via a composite set of linear models which correspond to the different operating points. For the case of nonlinear system we are dealing with in this research, such discrete time representation is given by:

$$y(t) = a_1(O_t)y(t-1)+\cdots+a_n(O_t)y(t-n)$$
$$+b_1(O_t)u(t-d-1)+\cdots+b_m(O_t)u(t-d-m)+w(t) \qquad (2)$$

where $a_i(O_t)$ and $b_j(O_t)$ $(i = 1,2,...,n, j = 0,1,...,m)$ are the unknown parameters which are functions of the measurable p-dimensional operating points O_t. The operating points may be some changing environmental operating condition that causes the system's parameters to vary, or may depend upon the past values of the system input and output. Due to their parallel processing capability, non-linear nature and their ability.

The Neural Network and Training Algorithms:
to generalize with a priori knowledge, neural network can be used successfully to capture dynamic, non-linear models of complex, multivariable systems. They therefore offer potential benefits in MPC strategies.
A major obstacle to the widespread use of advanced modeling and control technology is the cost of the model development and validation but the application of Artificial Neural Networks (ANNs) to provide cost efficient and reliable process models appears to be highly promising as demonstrated by Bhat *et al*, (1989),(Karla and Baker,1995).

Neural Network Topology:
The most widely used ANN architecture is the multi-layer, feed-forward network as shown in Fig. 3-a. where there are usually three layers of neurons; input, hidden and output layers are used. The connections between neurons carry signals multiplied by weights. Each neuron sums the incoming signal-weight products and a bias and passes the result through a non-linear activation function as represented by Eq. (3), (Fukuda and Shibata, 1992; Ludstrom *et al.*, 1995 and Montague, 1992):

$$y(t) = f\left(\sum_{i=1}^{n} w_i \cdot x_i(t) - \theta\right) \qquad (3)$$

where $y(t)$ is output, $x_i(t)$ is multiple inputs, w_i is a weight of connection, θ is a bias of neuron unit, t is a time and n is number of inputs. The neuron output function $f(x)$ sometimes uses the two valued function of 1 and 0 using threshold fend-limiter or the sigmoid function that is a continuous and nonlinear function. A conventional sigmoid function is represented by the following expressions :

$$f(x) = \frac{1}{1+e^{\alpha x}} \quad (0 < f(x) < 1) \qquad (4)$$

A tanh function is often used instead of the conventional sigmoid function since the output value of the sigmoid function is positive, whereas that of the tanh function being both positive and negative. The tanh function is as follows:

$$f(x) = \frac{1-e^{-\alpha x}}{1+e^{-\alpha x}} \quad (-1 < f(x) < 1) \qquad (5)$$

For the prediction of the behavior of the liquid-level system, we chose a feed-forward network with linear activation functions as shown in Fig. 2. Feed-forward nets with at least one hidden layer have the capability to approximate any desired nonlinear mapping to an arbitrary degree of accuracy. The neural net considered here consists of three layers: input layer, hidden layer, and output layer.

Inputs of the Neural Netware are: one step before outputs $(h_1(t-1), h_2(t-1))$, one step before inputs $(q_1(t-1), q_2(t-1))$ and one step before states values are fed into the network. The hidden layer consists of three neurons, while the output layer consists of two neurons. The network topology were optimized based on experience with different structures.

Training Algorithm : In order to make the neural network perform the desired mapping from the input layer to the output layer, one usually searches for the optimal connection weights w_{ij} between the neurons to approximate the desired mapping by so-called

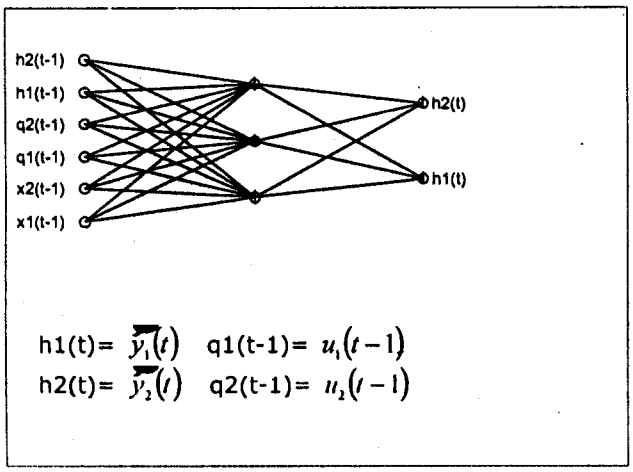

Fig. 2: Neural Network Employed Topology

training algorithms. The most popular training algorithm for feed-forward networks is the generalized delta-rule or back-propagation. For our case, let :

$$E_p = \frac{1}{2}\sum_j \left(t_{pj} - o_{pj}\right)^2 \tag{6}$$

be the measure of error on input/output pattern p, t_{pj} is the target input for the j^{th} component of the output pattern and o_{pj} is the element of the actual output pattern produced by the presentation of input pattern. Also let $E = \sum E_p$ be the overall measure of the error. Here, we define the weighted sum of the output of the previous layer, as the state of the unit. The output,

$$S_{pj} = \sum_i w_{ji} o_{pi} \tag{7}$$

$$o_{pj} = f_j\left(S_{pj}\right) \tag{8}$$

uses the sigmoid function, which is non decreasing and the differentiable function. To get the correct generalization of the delta rule, w_{ji} is set as :

$$\Delta_p w_{ji} = \frac{\partial E_p}{\partial w_{ji}} \tag{9}$$

It is useful to see this derivative as resulting from the product of two parts: one part reflecting the change in error as a function of the change in the net input to the unit and one part representing the effect of changing a particular weight on the net input. Thus we can write

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial S_{pj}} \frac{\partial S_{pj}}{\partial w_{ji}} \tag{10}$$

By Eq. (7) we see that the second factor is :

$$\frac{\partial S_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \sum_k w_{jk} o_{pk} = o_{pi} \tag{11}$$

Now let us define

$$\delta_{pj} = -\frac{\partial E_p}{\partial S_{pj}} \tag{12}$$

Eq. (10) thus has the equivalent form :

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} o_{pi} \tag{13}$$

This says that to implement a gradient descent in E, we should make the weight changes according to :

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \tag{14}$$

The interesting result is that there is a simple recursive computation of these δ's that can be implemented by propagating an error signal backward through the network.

To compute Eq. (12), the chain rule is applied to write this partial derivative as the product of two factors, one factor reflecting the change in error as a function of the output of the unit, and one reflecting the change in the output as a function of changes in the input,

$$\delta_{pj} = -\frac{\partial E_p}{\partial S_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial S_{pj}} \tag{15}$$

By Eq. (8) we see that :

$$\frac{\partial o_{pj}}{\partial S_{pj}} = f_j'\left(S_{pj}\right) \tag{16}$$

which is simply the derivative of the compressing function f_j for the j^{th} unit, evaluated at the net input S_{pj} to that unit. To compute the first factor, there are two cases. First, assume that unit u_j is an output unit of the network. In this case, it follows from the definition of E_p that :

$$\frac{\partial E_p}{\partial o_{pj}} = -\left(t_{pj} - o_{pj}\right) \tag{17}$$

Substituting for the two factors in eq. (15), we can get :

$$\delta_{pj} = \left(t_{pj} - o_{pj}\right) f_j'\left(S_{pj}\right) \tag{18}$$

for any output unit u_j. If u_j is not an output unit, the chain rule is used to write

$$\sum_k \frac{\partial E_p}{\partial S_{pk}} \frac{\partial S_{pk}}{\partial o_{pj}} = \sum_k -\frac{\partial E_p}{\partial S_{pk}} \frac{\partial}{\partial o_{pj}} \sum_i w_{ki} o_{pi} = \sum_k \frac{\partial E_p}{\partial S_{pk}} w_{kj} = -\sum_k \delta_{pk} w_{kj} \tag{19}$$

In this case, substituting for the two factors in equ. (15) yields :

$$\delta_{pj} = f_j'\left(S_{pj}\right) \sum_k \delta_{pk} w_{kj} \tag{20}$$

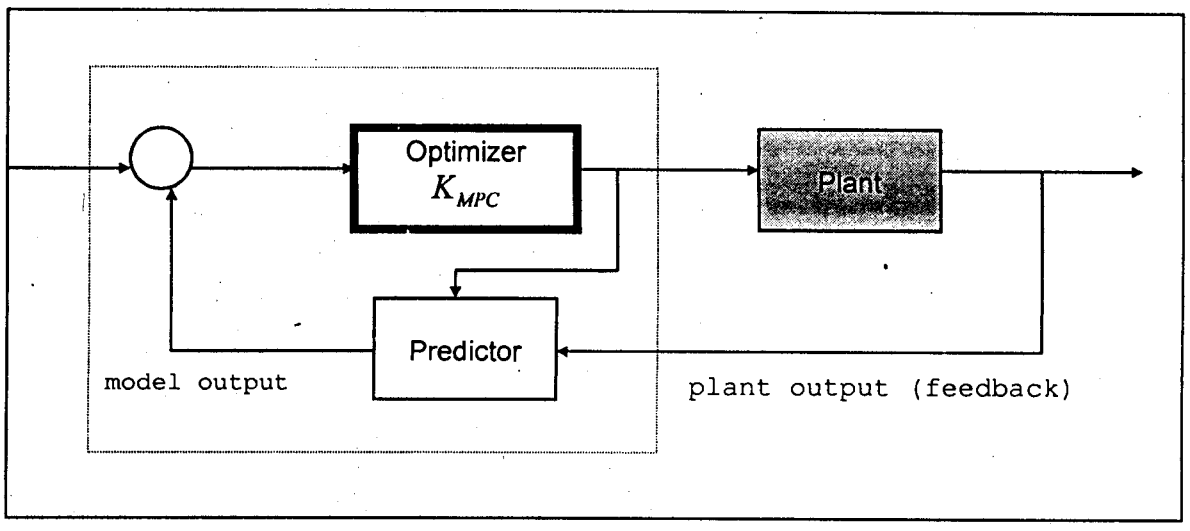

Fig. 3: MPC Controller Separated into a Predictor and an Optimizer

whenever u_j is not an output unit. Eq. (18) and Eq. (20) give a recursive procedure for computing the δ's for all units in the network, which are then used to compute the weight changes in the network according to Eq. (14). This procedure constitutes the generalized delta rule for a feed-forward network. The aim of our work is to obtain a nonlinear black-box model of a process by training a neural net from the generated training data. The simulated model output is compared with the actual output and prediction error. The identification model has the form of, Narendra (1996)

$$\hat{y}(k)=N_F\{y(k-1),y(k-2),\ldots y(k-n+1),u(k-1),u(k-2),\ldots u(k-n+1)\} \quad (21)$$

where $\hat{y}(k)$ is the output of the neural network at time k and N_F is known from the neural network topology. The model represented by Eq. (21) is generally referred to as Nnon-linear Auto Regressive Moving Average (NARMA) model of the system. When investigated in discrete-time can be modeled by the following Eq. (9);

$$z(k+1)=\sum_{i=0}^{n-1} a_i y(k-i)+\sum_{i=0}^{n-1} b_i u(k-i) \quad (22)$$

where $u(k)$ is assumed to be the plant input sequence, $y(k)$ the plant output sequence and $z(k)$ a pseudo-signal, taken that the actual plant output is obtained from:

$$y(k)=\phi\{z(k)\} \quad (23)$$

in which $\phi\{\cdot\}$ is a nonlinear function, realizing the straightforward linear case for (22) when $\phi=1$. eq 22 and eq 23 together define the NARMA plant model, (Warwick) and this has been used as the basis for ANN system identification studies, the key aim being for the ANN to model the underlying plant dynamics in a plant whose performance is characterized by eq. (22) and eq. (23). After further manipulation of variables, the neuro-model of the liquid level system is:

$$\begin{bmatrix} Y_1(z) \\ Y_2(z) \end{bmatrix} = \begin{bmatrix} \frac{0.1998-0.125z^{-1}}{1-1.5z^{-1}+0.521z^{-2}} & \frac{0.1+0.03z^{-1}}{1-1.5z^{-1}+0.521z^{-2}} \\ \frac{0.0198+0.0187z^{-1}}{1-1.5z^{-1}+0.521z^{-2}} & \frac{1.489-1.411z^{-1}}{1-1.5z^{-1}+0.521z^{-2}} \end{bmatrix} \begin{bmatrix} u_1(z) \\ u_2(z) \end{bmatrix} \quad (24)$$

The Model Predictive Control (Mpc) :

In general, the MPC strategy consists of an optimizer and a plant model as shown in Fig. 3. The optimizer computes a horizon of future control actions in such a way as to make a balance between excessive control action and output deviation. Only the first control action will be implemented with the computation process being repeat at each time period. This allows computation of an optimum based on future actions while minimizing the effects of modeling errors.

Plant Modeling :

In the original Dynamic Matrix Control (DMC) formulation (Cutler and Ramaker, 1980),a step response model of the plant is used to predict the future behavior of the controlled variables. Let the step response of a SISO system be represented by the sequence Eq. (11):

$$[S_1 \quad S_2 \ldots\ldots S_{n-1} \quad S_n \quad S_{n+1} \ldots\ldots] \quad (25)$$

where the k[th] element is the output at time k caused by a unit step input at the initial time. For a stable

plant this sequence will asymptotically reach a constant value, i.e. $S_n \approx S_{n+1}$. For a MIMO system with n_u inputs and n_y outputs we get:

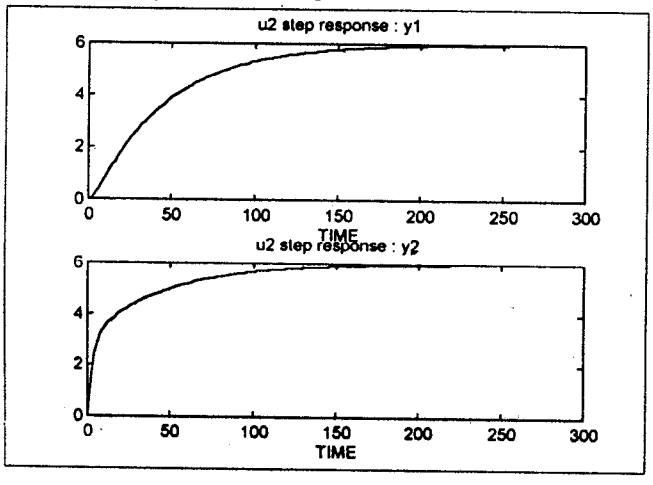

Fig. 4. Model Obtained by Neural Network System

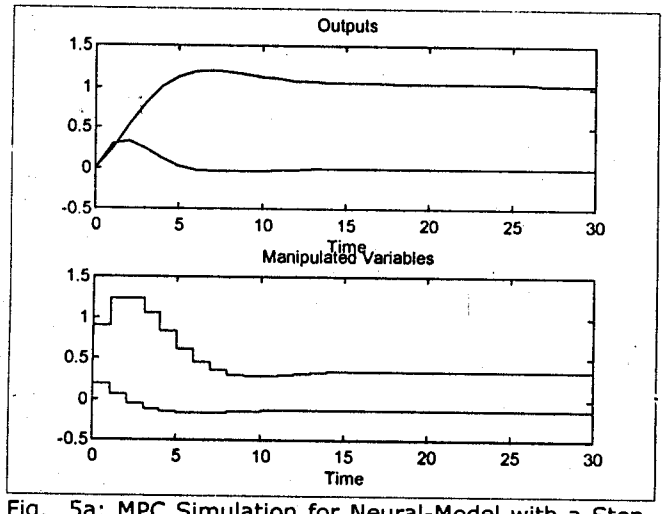

Fig. 5a: MPC Simulation for Neural-Model with a Step in $y_1(h_1)$

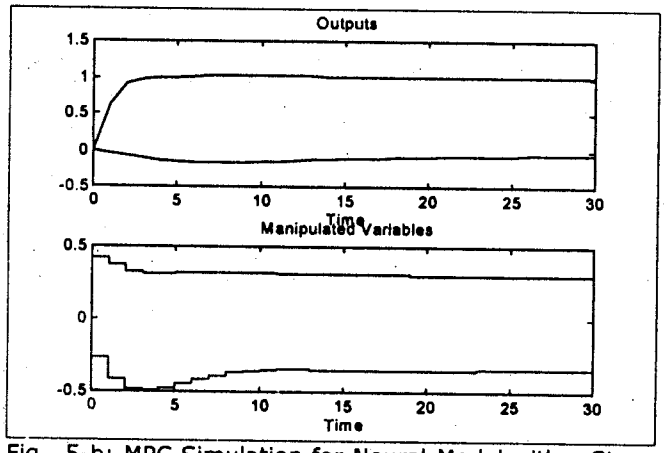

Fig. 5-b: MPC Simulation for Neural-Model with a Step in $y_2(h_2)$

$$S_i = \begin{bmatrix} S_{1,1,i} & S_{1,2,i} & \cdots & S_{1,m-1,i} \\ S_{2,1,i} & S_{2,2,i} & \cdots & S_{2,m-1,i} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ S_{m,1,i} & S_{m,2,i} & \cdots & S_{m,m-1,i} \end{bmatrix} \quad i = 1,2,\ldots,n \tag{26}$$

The step response model can be represented in the following state space form:

$$Y(k+1) = MY(k) + S\Delta u(k-1) \tag{27}$$

$$y(k) = NY(k) \tag{28}$$

where,

$$\Delta u(k) = u(k) - u(k-1) \tag{29}$$

$$Y(k) = \left[y(k)^T \quad y(k+1)^T \ldots\ldots y(k+n-1)^T\right]^T \tag{30}$$

$$M = \begin{bmatrix} 0 & I_m & 0 & \cdots & 0 & 0 \\ 0 & 0 & I_m & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 0 & I_m \\ 0 & 0 & 0 & \cdots & 0 & I_m \end{bmatrix}_{n \times n_y} \tag{31}$$

$$S = \begin{bmatrix} S_1 \\ S_2 \\ \cdot \\ \cdot \\ S_{n-1} \\ S_n \end{bmatrix} \tag{32}$$

and $$N = [I_{ny} \quad 0 \quad 0 \ldots\ldots 0 \; 0] \tag{33}$$

$\Delta u(k)$ is a vector of changes in the manipulated inputs at time k and $y(k)$ is the output vector at time k. The vector $Y(k+1)$ represents the dynamic states of the system. Each state, $y(k+l)$, has a special interpretation: it is the future output vector at time $k+l$ assuming constant inputs, i.e. ($\Delta u(k+j) = 0$ for $j \geq 0$). The new state vector $Y(k+l)$ is the old vector $Y(k)$ shifted up n_y elements plus the contribution made by the latest input change $\Delta u(k)$.

The Predictor :

The objective of the predictor is to generate a vector $\bar{y}(k+1|k)$ of predicted open loop outputs over a horizon of p future time steps. This prediction vector is then used as an input to the optimizer. The predictor is described by the following equations:

$$\bar{Y}(k) = M\bar{Y}(k-1) + S\Delta u(k-1) \tag{34}$$

$$\bar{y}(k) = N\bar{Y}(k) \tag{35}$$

$$\bar{y}(k+1|k) = M_p\bar{Y}(k) + \partial[\hat{y}(k) - \bar{y}(k)] \tag{36}$$

where M_p is the first $p \times n_y$ rows of M and :

$$\partial = [I_{ny} \quad I_{ny} \quad \ldots\ldots \quad I_{ny} \quad I_{ny}]^T \tag{37}$$

We use "-" to denote that the output is from the model and not from the true plant, $\hat{y}(k)$ is a vector of measured outputs at time k. $\hat{y}(k)$ and $\bar{y}(k)$ are

discontinuous at k_- while $u(k)$ is discontinuous at k_+, i.e. $\vec{y}$ is measured slightly before time k and u is adjusted slightly after time k.

The Optimizer :

We use the Quadratic Dynamic Matrix Control (QDMC) objective function from Garcia and Morshedi (1986) :

$$J(k) = \min_{\Delta U(k|k)} \left\{ \left\| \vec{y}_m(k+1 \mid k) - R(k+1 \mid k) \right\|^2 + \left\| \Delta U(k \mid k) \right\|^2 \right\} \quad (38)$$

$$K_{MPC} = \sum_{j=1}^{n_2} \sum_{i=1}^{n_y} (ywt_i(j)[r_i(k+j) - \bar{y}_i(k+j)])^2 + \sum_{j=1}^{n_u} \sum_{i=1}^{n_u} (uwt_i(j)\Delta\hat{u}_i(j))^2 \quad (39)$$

where $\Delta\hat{u}_i(j)$ (is the optimal control sequence computed at k for n_b future input moves, n_b is the number of blocks or moves of the manipulated variables.), where $\bar{y}_i(k+j)$ is a prediction of output i at time j sampling periods into the future (relative to the current time, k), which is a function of $\Delta\hat{u}_i(j), r_i(k+j)$ is the corresponding future set point.

Neuro Model Predictive Control Simulation Results:

After exciting the multivariable double tank system with some *pseudo random inputs* and recording the corresponding *two system outputs* h_1 and h_2, a Neural Network model was obtained by training the Neuro model already shown in Fig. 2. The next step in the simulation process is to validate the model as were discussed in the modeling section. In terms of model validation, it was found that the Neural Network model has a good degree of accuracy once compared to the behavior of the original plant output.

For the purpose of analyzing the results of using Neural Networks in MPC systems, three sets of models were developed corresponding to a perfect model, conventional statistical model and neural network model, and utilized as the process model for MPC controller. A model predictive controller synthesis using the perfect model, conventional statistical based model and neural network model were examined in response to unit step in $y_1(h_1)$, the output $y_2(h_2)$ set point is zero and the response of closed-loop system to unit step in $y_2(h_2)$, the output $y_1(h_1)$ set point is zero as shown in Fig. 4.

In reference to Figs. 5-a-b. , model predictive controller synthesis and simulation using a neural network model was found to have better performance than that using a conventional statistical model as compared with the model predictive controller with the perfect model. The future work in this area is to employ a better way of modeling that could take in consideration the way in which the process is operating. This undoubted will lead us to fuzzy neuro modeling technique that allows a sort of adaptation in the controller parameter in reference to the model being detected by the fuzzy neuro algorithms.

Conclusion :

We have presented an approach to model predictive control of a nonlinear plant without a priori information about the plant dynamics. The process dynamics were identified from an input-output data set which is obtained by randomly changing the manipulated variables (u_1, u_2). We modified a nonlinear extension to the standard MPC control scheme for operation with the trained neural network. A model predictive controller using a neural network model was found to have better performance than that using a conventional statistical model as compared with the model predictive controller with the perfect model. Further work will be to extend this neural network based MPC by using a technique for the modeling of nonlinear system using a fuzzy neural network topology. The fuzzy neural network approach to nonlinear process modeling provides a way of opening up the purely 'black box' approach normally seen in neural network applications.

References :

Cutler C. and B. Ramaker, 1980 Dynamic Matrix Control: A Computer Control Algorithm , Joint Automatic Control Conference.

Draeger A.,S. Engell and H. Ranke, 1995. Model Predictive Control Using Neural Network, IEEE Control Systems.

Fukuda T. , T . Shibata, 1992. Theory and Applications of Neural Networks for Industrial Control Systems , IEEE, 39.

Garcia C. E. , M. Morari , Internal Model Control: A Unifying Review and Some New Results , Ind. Eng. Chem. Process 21 : 308-323.

Garcia C. E. and A.M. Morshedi, Quadratic Programming Solution of Dynamic Matrix Control (QDMC) , Chem. Eng. Commun. 46. 73-87.

Hunt K. J.,D. Sbarbaro , R. Zbikowski R. and P. Gawthrop, 1992. Neural Networks for Control Systems-A Survey ", Automatica 28, 1083-1112.

Karla V. R. and Baker H. C., 1995. "Neural-Network-Based Model Predictive Control: A Case Study ", IEEE Control Systems.

Lundstrom P., H. Lee, M. Morari and S. Skogestad, 1995. Limitations of Dynamic Matrix Control Computer Chem. Eng 9: 409-421.

Montague G. A., A.J. Morris and M.J.Willis, 1992 Neural Networks: Methodologies for Process Modeling And Control " , IFAC Artificial Intelligence in Real-Time Control. Delft The Netherlands.

Narendra K. S., 1996. Neural Networks for Control: Theory and Practice ", IEEE.

Warwick K., An Introduction to Control System ", World Scientific.

Ogata K., "Modern Control Engineering ", Prentice Hall.

Wu Q. , and Harris J., Modeling and Adaptive Filtering of Nonlinear Systems Using Neural Network ", Uni. of Southampton, Technical Report.

Zhan J. and M. Ishida, 1997. The Multi-Step Predictive Control of Nonlinear SISO Processes With A Neural MPC " Computers Chem. Eng 21: 2, 201-210.