ISTANBUL UNIVERSITY –
JOURNAL OF ELECTRICAL & ELECTRONICS ENGINEERING

YEAR : 2007 (277 - 285)
VOLUME : 7
NUMBER : 1

PREDICTIVE CONTROL OF CHAOTIC DISCRETE PLANTS

S. SHAMS ABBAD FARAHANI[1] **M.A. NEKOUI[2]**

[1]Department of electrical engineering, Azad Islamic University, Tehran South, Iran
[2]Department of electrical engineering, K.N Toosi University, Tehran, Iran

[2]E-mail: manekoui@eetd.kntu.ac.ir

ABSTRACT

This paper deals with identification and predictive control of a nonlinear chaotic discrete plant. The main difficulties for identification and control of this plant arise from the strongly nonlinear center and its chaotic behavior. First, an internal feedback is applied to suppress the chaotic behavior. Then, a neural network based predictive controller using Multi Layer Perceptron (MLP) is designed to govern the dynamics of this plant. The effectiveness of the purposed methodology is shown through simulation results.

Keywords: *Neural network control, Multi Layer Perceptron, Local Linear Model Tree, predictive control, Chaos.*

1. INTRODUCTION

The analysis and control of chaotic behavior in dynamical systems has been widely investigated in recent years [29-32] and the predictive control is now widely used in industry and a large number of implementation algorithms. Most of the control algorithms use an explicit process model to predict the future behavior of a plant and because of this, the term model predictive control (MPC) is often utilized [1-2]. The inclusion of the constraints is the feature that most clearly distinguishes MPC from other process control techniques, leading to a tighter control and a more reliable controller. Another important characteristic, which contributes to the success of the MPC technology, is that the MPC algorithms consider plant behavior over a future horizon in time. Thus, the effects of both feedforward and feedback disturbances can be anticipated and eliminated, which permits the controller to drive the process output more closely to the reference trajectory.

Although more practical plants usually contain complex nonlinearities or chaotic behavior, most of the MPC algorithms are based on a linear model of the process. Linear models such as step response and impulse response models derived from the convolution integral are preferred, because they can be identified in a straightforward manner from process test data. In addition, the goal for most of the applications is to maintain the system at a desired steady state, rather than moving rapidly between different operating points, so a precisely identified linear model is sufficiently accurate in the neighborhood of a single operating point. As linear models are reliable from this point of view, they will provide most of the benefits with MPC technology. Even so, if the process is highly nonlinear and subject to chaotic behavior, a nonlinear model will be necessary to describe the behavior of the process. Also, in servo control problems where the operating point is frequently changing, a nonlinear model of the plant is indispensable [3-5].

Received Date : 22.06.2006
Accepted Date: 01.02.2007

The key issue in this work is to propose an internal feedback loop based on time delayed state feedback to suppress the chaotic behavior in the system, then the appropriate MPC controller is applied as shown in figure 1. In situations like the ones mentioned above, the task of obtaining a high-fidelity model is more difficult to build for nonlinear processes. Recently, neural networks have become an attractive tool in the construction of models for complex nonlinear systems [6-7]. A large number of control and identifications structures based on neural networks have been proposed [8-15].

Most of the nonlinear predictive control algorithms imply the minimization of a cost function, by using computational methods for obtaining the optimal command to be applied to the process. The implementation of the nonlinear predictive control algorithms becomes very difficult for real-time control because the minimization algorithm must converge at least to a sub-optimal solution and the operations involved must be completed in a very short time (corresponding to the sampling period). This paper analyzes an artificial neural network based nonlinear predictive controller for a nonlinear chaotic discrete plant. The procedure is based on construction of a neural network model for the process and the proper use of that in the optimization process. The method eliminates the most significant obstacles for nonlinear MPC implementation by developing a nonlinear model, designing a neural predictor and providing a rapid, reliable solution for the control algorithm and inserts an internal feedback based on delayed state feedback to suppress the chaotic behavior. Also, the performance of the proposed neural network based predictive controller is compared with that of LOLIMOT, which the latter leads to better performance.

The organization of this paper is as follows: Sections II and III present the predictive control methodology based on MLP and the simulation results using MLP. Section IV, V, VI and VII present the predictive control methodology based on LOLIMOT and the simulation results. Finally, the paper is concluded in Section VIII.

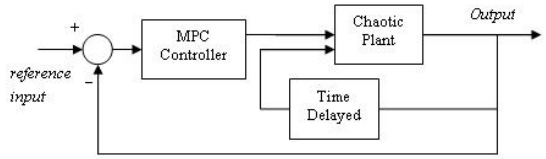

Figure (1): *MPC Controller for a Chaotic Plant*

2. PREDICTIVE CONTROL METHODOLOGY BASED ON MULTI LAYER PERCEPTRON

This section presents the role and architecture of the neural predictors resulting from the following nonlinear modeling techniques based on neural network principles [26-28].

A network with $k+1$ layers and $n_0,n_1,\dots n_k$ points in each layer is recognized . In zero and first layers, we mention x as input layer vector, w_1 as weight vector, z_1 as state vector and y_k as output vector. Thus we obtain:

$$X=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n0} \end{bmatrix}, W_1=(w_{ij}^1)_{n0\times n_k}, Z_1=\begin{bmatrix} z_1^1 \\ z_2^1 \\ \vdots \\ z_{n_k}^1 \end{bmatrix}=W_1^T X, Y_1=\begin{bmatrix} y_1^1 \\ y_2^1 \\ \vdots \\ y_{n_k}^1 \end{bmatrix}=f(Z_1) \quad (1)$$

f is a function which is considered to be:

$$f(s)=\frac{1}{1+\exp(-s)} \quad (2)$$

to implement BP algorithm we have to minimize the following cost function:

$$F(w)=\frac{1}{2}\sum_{i=1}^{Q} e_i^T e_i=\frac{1}{2}\sum_{i=1}^{Q}(R_i-Y_i')^T(R_i-Y') \quad (3)$$

Where, R is the desired output vector and w is a vector including bias and weights. Using steepest descent algorithm to minimize that cost function, we have:

$$w_{k+1}=w_k+\mu(-\frac{\partial e}{\partial w})_{w=w_k} \quad (4)$$

Where, μ is the learning rate.

In a Multi Layer Perceptron (MLP) with Back propagation as training method with just one hidden layer, h neurons in hidden layer and p neurons in input layer, the output of MLP network becomes:

$$y(i)=\sum_{j=1}^{h} w_j f_j(z_j(i))+b \quad (5)$$

$$z(j)=\sum_{k=1}^{p} w_{j,k}x(k)+bj \quad (6)$$

and *y(i)* is the output of the i^{th} neuron, f_j output function of j^{th} neuron in hidden layer, *z(j)* activation level of the output function of j^{th} neuron, *h* the neuron number in hidden layer, *p* the number of input neurons, w_j the connecting weight of j^{th} neuron of hidden layer to output neuron, $w_{j,k}$ connecting weight of k^{th} input neuron to j^{th} neuron of hidden layer, b_j the bias of j^{th} neuron in hidden layer and *b* as the bias in output neuron.

A quadratic cost function is utilized to compute the prediction error and to derive the optimal predictive control strategy.

$$Min_{u(t)}\ J = Min_{u(t)} \frac{1}{2}\sum_{i=N1}^{N2}\left\|y(t+i)-r(t+i)\right\|^2 + \frac{1}{2}\sum_{i=0}^{Nu}\lambda_i\left\|u(t+i)\right\|^2 + \lambda_i'\left\|\Delta u(t+i)\right\|^2 \quad (7)$$

$$s.t.\ \begin{cases} u_{\min} \le u(t+i) \le u_{\max} & i = 0,...,N_u \\ y_{\min} \le y(t+i) \le y_{\max} & i = N_1,...,N_2 \\ \left\|\Delta u(t+i)\right\| \le \Delta u_{\max} & i = 0,...,N_u \\ \Delta u(t+i) = 0 & i > N_u - 1 \end{cases} \quad (8)$$

Where λ and λ' are weighting matrixes and N_1, N_2, N_u are the minimum, maximum of prediction horizon and control horizon, respectively.

Using steepest descent strategy we have:

$$u^{i+1}(t) = u^i(t) - \alpha\frac{\partial J}{\partial u^i(t)} \quad (9)$$

Where $\alpha \in R^+$ is the optimization step.

The derivation of the cost function (J) in time of $t+h, (h = 1,2,...,N_u)$ is as follows:

$$\frac{\partial J}{\partial u(t+h)} = -\sum_{i=N_1}^{N_2}[y(t+i)-r(t+i)]\frac{\partial y(t+i)}{\partial u(t+h)} + \sum_{i=0}^{N_u}\lambda_i u(t+i)\frac{\partial u(t+i)}{\partial u(t+h)} + \sum_{i=0}^{N_u}\lambda_i \Delta u(t+i)\frac{\partial \Delta u(t+i)}{\partial u(t+h)} \quad (10)$$

Possibly we write $\frac{\partial \Delta u(t+j)}{\partial u(t+h)}$ in the form of Kronecker delta function and we have:

$$\frac{\partial u(t+j)}{\partial u(t+h)} - \frac{\partial u(t+j-1)}{\partial u(t+h)} = \delta(h,j) - \delta(h,j-1) \quad (11)$$

While Kronecker delta function is

$$\delta(h,j) = \begin{cases} 1 & if \quad h = j \\ 0 & if \quad h \ne j \end{cases} \quad (12)$$

So, we have:

$$\frac{\partial u(t+j)}{\partial u(t+h)} = \delta(h,j) \quad (13)$$

In accordance to (5) and (6), we have:

$$y(t+i) = \sum_{j=1}^{h} w_j f_j(z_j(t+i)) + b \quad (14)$$

And $\frac{\partial y(t+j)}{\partial u(t+h)}$ can be written as:

$$\frac{\partial y(t+j)}{\partial u(t+h)} = \sum_{j=1}^{h} w_j \frac{\partial f_j(z_j(t+i))}{\partial u(t+h)} \quad (15)$$

Using Chain rule we obtain:

$$\frac{\partial f_j(z_j(t+i))}{\partial u(t+h)} = \frac{\partial f_j(z_j(t+i))}{\partial z_j(t+i)}\frac{\partial z_j(t+i)}{\partial u(t+h)} \quad (16)$$

$\frac{\partial f_j(z_j(t+i))}{\partial z_j(t+i)}$ Can be calculated using output function deviation. $z_j(t+i)$ is depended on input $u(t+i)$,delayed inputs $u(t+i-1)$, $u(t+i-2),\ldots$, $u(t+i-n)$ and output $y(t+i-1)$, delayed output $y(t+i-2)$, $y(t+i-3),\ldots, y(t+i-m)$.

Suppose having *k* neurons as input while the first neurons from1 to *q* introduce $u(t+i-n);\ n = 0,1,...,\ q-1$ Neurons from $q+1$ to K show $y(t+i-1-m);\ m = 0,1,...,\ K-q-1$.

So we have:

$$z_j(t+i) = + \sum_{m=0}^{K-q-1} w_{j,m+q+1} y(t+i-1-m) + \sum_{n=0}^{q-1} w_{j,n+1} u(t+i-n) \quad (17)$$

and $\frac{\partial z_j(t+j)}{\partial u(t+h)}$ can be calculated as follows:

$$\frac{\partial z_j(t+j)}{\partial u(t+h)} = \sum_{m=0}^{K-q-1} w_{j,m+q+1}\frac{\partial y(t+i-1-m)}{\partial u(t+h)} + \sum_{n=0}^{q-1} w_{j,n+1}\frac{\partial u(t+i-n)}{\partial u(t+h)} \quad (18)$$

Then,

$$\frac{\partial u(t+i-n)}{\partial u(t+h)} = \begin{cases} \delta(i-n,h) & if \quad i-n < N_u \\ \delta(N_u,h) & if \quad i-n > N_u \end{cases} \quad (19)$$

$\frac{\partial y(t+i-1-m)}{\partial u(t+h)}$ can be calculated through a repetitive calculation but when $i-1-m<1$ the result will be zero.

3. SIMULATION RESULTS OF PREDICTIVE CONTROL IN THE CHAOTIC PLANT WITH THE USE OF MLP

Consider the following discrete chaotic plant which behaves chaotically as shown in figure 2.

$$y_{k+1}=\frac{1.5y_k y_{k-1}}{1+y_k^2+y_{k-1}^2}-2\cos(y_k+y_{k-1})+1.2u_k \quad (20)$$

Where, y_k is the scalar state variables, u_k is the predictive control effort, and $k=0,1,...$ is the number of the sampling instants.

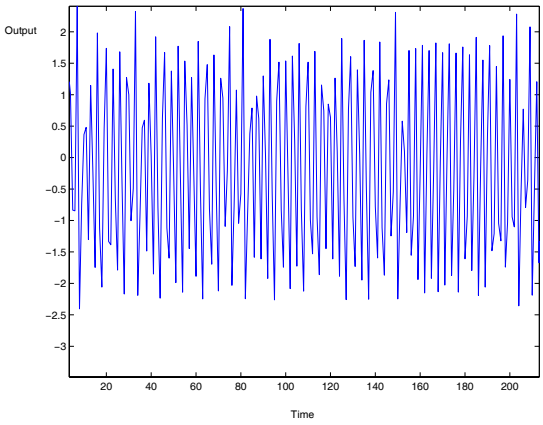

Figure (2): *The chaotic behavior*

To design the MPC controller, the chaotic behavior must be eliminated by an internal time-delayed state feedback as follows:

$$y_{k+1}=\frac{1.5y_k y_{k-1}}{1+y_k^2+y_{k-1}^2}-2\cos(y_k+y_{k-1})+1.2u_k+K(y_{k-1}-y_k) \quad (21)$$

Where, K is the delayed state feedback gain and must be chosen appropriately. The set point tracking results of the simulation on the plant is depicted in Figure 3. Clearly the system could track the set points with satisfactory performance using a numerical optimization the prediction and control horizons are both 2 also λ_i' is 0.1

Next, the cost function J is constructed

$$Min_{u(t)}\ J=+\sum_{i=0}^{N_u}\lambda_i'\|\Delta u(t+i)\|^2+$$

$$Min_{u(t)}\left\{\frac{1}{2}\sum_{i=N1}^{N2}\|y(t+i)-r(t+i)\|^2\right\}$$

The minimization algorithm gives the control input vector $U=[u(t),u(t-1),u(t-2),\ y(t),y(t-1),y(t-2)]^T$ to be applied to the plant.

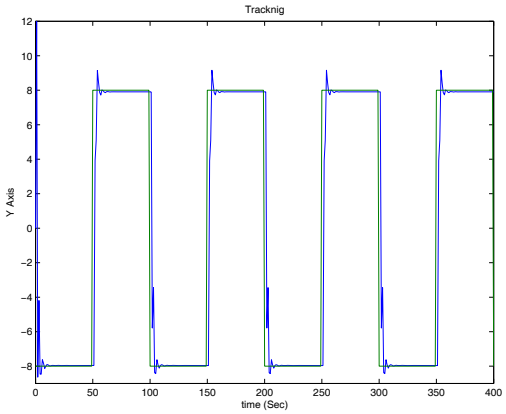

Figure (3): *Tracking*

Clearly the system could track the set points with satisfactory performance.

4. AN INTRODUCTION TO LOCAL LINEAR MODEL TREE (LOLIMOT)

The network structure of a local linear neurofuzzy model [6] is depicted in Figure 4. Each neuron realizes a local linear model (LLM) and an associated validity function that determines the region of validity of the LLM. The validity functions form a partition of unity, i.e., they are normalized such that

$$\sum_{i=1}^{M}\Phi_i(\underline{z})=1 \quad (22)$$

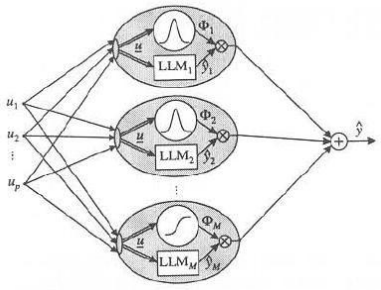

Figure (4): *Network structure of a local linear neuro fuzzy model with M neurons for n_x LLM inputs x and n_z validity function inputs z.*

for any model input z. The output of a local linear neuro-fuzzy model is calculated as

$$\tilde{y} = \sum_{i=1}^{M} (w_{i,0} + w_{i,1} x_1 + ... + w_{i,nx} x_{nx}) \Phi_i(z) \quad (23)$$

where the local linear models depend on $\underline{x} = [x_1, x_2, ..., x_{nz}]^T$ and the validity functions depend on $\underline{z} = [z_1, z_2, ..., z_{nz}]^T$. Thus, the network output is calculated as a weighted sum of the outputs of the local linear models where the $\Phi i(.)$ are interpreted as the operating point dependent weighting factors. The network interpolates between different LLMs with the validity functions. The weights $w_{i,j}$ are linear network parameters.

The validity functions are typically chosen as normalized Gaussians. If these Gaussians are furthermore axis-orthogonal the validity functions are

$$\Phi_i(\underline{z}) = \frac{\mu_i(\underline{z})}{\sum_{j=1}^{M} \mu_j(\underline{z})}$$

$$\mu_i(\underline{z}) = \exp\left(-\frac{1}{2} \left(\frac{(z_1 - c_{i1})^2}{\sigma_{i1}^2} + \Lambda + \frac{(z_{nz} - c_{inz})^2}{\sigma_{inz}^2} \right) \right)$$

$$= \exp\left(-\frac{1}{2} \frac{(z_1 - c_{i1})^2}{\sigma_{i1}^2} \right) \Lambda \ \exp\left(-\frac{1}{2} \frac{(z_{nz} - c_{inz})^2}{\sigma_{inz}^2} \right) \quad (24)$$

The centers and standard deviations are *nonlinear* network parameters.
In the fuzzy system interpretation each neuron represents one rule. The validity functions represent the rule premise and the LLMs represent the rule consequents. One-dimensional Gaussian membership functions:

$$\mu_{ij}(z_j) = \exp\left(-\frac{1}{2} \frac{(z_j - c_{ij})^2}{\sigma_{ij}^2} \right) \quad (25)$$

can be combined by a t-norm (conjunction) realized with the product operator to form the multidimensional membership functions in (22). One of the major strengths of local linear neuro-fuzzy models is that premises and consequents do not have to depend on identical variables, i.e. *z* and *x* can be chosen independently.

5.PREDICTIVE CONTROL METHODOLOGY BASED ON LOIMOT ALGORITHM

The prediction output can be written as [25]:

$$\hat{y}_{t+d+i} = \overline{F}_i^T \overline{\Phi}_i + \sum_{j=0}^{i} \overline{G}_{ij}^T \overline{\Phi}_{ij} u_{t+j} \quad (26)$$

$\overline{\Phi}_i$ and $\overline{\Phi}_{ij}$ are basis functions, we can possibly define the weight and basis function vectors as:

$$\overline{F}_i = [f_{i,1} \ f_{i,2} \ \Lambda \ f_{i,N_i}]^T$$

$$\overline{G}_{ij} = [g_{ij,1} \ g_{ij,2} \ \Lambda \ g_{ij,N_{ij}}]^T$$

$$\overline{\Phi}_i = [\varphi_{i,1}(x_t) \ \varphi_{i,2}(x_t) \ \Lambda \ \varphi_{i,N_i}(x_t)]^T \quad (27)$$

$$\overline{\Phi}_{ij} = [\varphi_{ij,1}(x_t) \ \varphi_{ij,2}(x_t) \ \Lambda \ \varphi_{ij,N_{ij}}(x_t)]^T$$

to define how well the predicted process output tracks the reference trajectory, a number of cost functions are employed for predictive control , here we use a cost function which is of the following quadratic form:

$$J_{np} = \frac{1}{2} \| R_{t+d+L} - Y_{t+d+L} \|_2^2 + \frac{1}{2} \alpha \| \Delta U_{t+L} \|_2^2 \quad (28)$$

where:

$$R_{t+d+L} = [r_{t+d} \ \ r_{t+d+1} \ \Lambda \ \ r_{t+d+L}]^T$$

$$Y_{t+d+L} = [\hat{y}_{t+d} \ \ \hat{y}_{t+d+1} \ \Lambda \ \ \hat{y}_{t+d+L}]^T \quad (29)$$

$$U_{t+L} = [u_t \ u_{t+1} \ \Lambda \ \ u_{t+L}]^T$$

R_{t+d+L}, Y_{t+d+L} and U_{t+L} are the future reference input, predicted output and control input vectors, respectively, L is the control horizon, $L+D$ is the prediction horizon, and $\alpha > 0$ is the weight.

The optimal controller output sequence over the prediction horizon is obtained by minimizing the performance index J_{np} with respect to U_{t+L}. This can be carried out by taking the derivative of the performance function J_{np} with respect to the control input vector U_{t+L} and results in:

$$\frac{\partial J_{np}}{\partial U_{t+L}} = 0 \ \Rightarrow \ \frac{\partial Y_{t+d+L}^T}{\partial U_{t+d+L}} (Y_{t+d+L} - R_{t+d+L})$$

$$+ \alpha \frac{\partial \Delta U_{t+L}^T}{\partial U_{t+L}} \Delta U_{t+L} = 0 \quad (30)$$

Using the mentioned predictor, the derivatives of Y_{t+d+L} with respect to the control input vector U_{t+L} are given by:

$$\frac{\partial Y_{t+d+L}^T}{\partial U_{t+d}} = Q_L^T =$$

$$\begin{bmatrix} \overline{G}_{00}^T \overline{\Phi}_{00} & 0 & 0 & 0 \\ \overline{G}_{10}^T \overline{\Phi}_{10} & \overline{G}_{11}^T \overline{\Phi}_{11} & 0 & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & 0 \\ \overline{G}_{L0}^T \overline{\Phi}_{L0} & \overline{G}_{L1}^T \overline{\Phi}_{L1} & \mathrm{K} & \overline{G}_{LL}^T \overline{\Phi}_{LL} \end{bmatrix}^T \quad (31)$$

Let

$$\mathrm{H}_{\mathrm{L}} = [\overline{\mathrm{F}}_0^{\mathrm{T}} \overline{\Phi}_0 \;\; \overline{\mathrm{F}}_1^{\mathrm{T}} \overline{\Phi}_1 \;\; \Lambda \;\; \overline{\mathrm{F}}_{\mathrm{L}}^{\mathrm{T}} \overline{\Phi}_{\mathrm{L}}]^{\mathrm{T}}$$

Suppose $I_1 = [1,0,\Lambda\ ,0]$ is an identity vector and the matrix D_l is of the form

$$D_L = \begin{bmatrix} 1 & & & 0 \\ -1 & 1 & & \\ & \mathrm{O} & \mathrm{O} & \\ 0 & & -1 & 1 \end{bmatrix}$$

It is clear that the controller input vector U_{t+L} can be calculated by

$$U_{t+L} = \left(Q_L^T Q_L + \alpha D_L^T D_L\right)^{-1} \left(Q_L^T R_{t+d+L} - Q_L^T H_L + \alpha I_1^T u_{t-1}\right) \quad (32)$$

Thus, the control input u_t *minimizing the performance function* J_{np} *is given by:*

$$u_t = I_1 \left(Q_L^T Q_L + \alpha D_L^T D_L\right)^{-1} \left(Q_L^T R_{t+d+L} - Q_L^T H_L + \alpha I_1^T u_{t-1}\right) \quad (33)$$

6. ON-LINE LEARNING OF NEURAL PREDICTORS

Here we consider the online adjustment of the weights of the i^{th} predictor [25]. The weight estimation of the other predictors are the same. It will be assumed that the basis functions which are used in the predictors are given and the required prediction accuracy can be achieved by adjusting the corresponding weights to those functions.

Using the available output data and the input data, the output of the i^{th} predictor at time t can be written as

$$y_t = (\overline{F}_i^*)\Phi(x_{t-d-i}) + \sum_{j=0}^{i} (\overline{G}_{ij}^*)\overline{\Phi}_{ij}(x_{t-d-i}) u_{t-d-i+j} + \varepsilon_t \quad (34)$$

Where, $(\overline{F}*_i)$ and $(\overline{G}_{ij}*)$ are the optimal estimates of the weight vectors $(\overline{F}_i)$ and $(\overline{G}_{ij})$ for $j = 0,1,2,...,$ respectively ε_t is the approximation error of the predictor using the neural network and is assumed to be bounded by a positive number of δ for all time, that is:

$$\max_{t \in N^+} |\varepsilon_t| \le \delta \quad (35)$$

Where, the number δ represents the prediction accuracy, which is known by assumption. The i^{th} estimated predictor can also be compactly expressed by:

$$\hat{y}_t = W_t^T \Phi_{t-1}$$

Where the weight vector W_t and the basis function vector Φ_t are

$$\Phi_t = \begin{bmatrix} \overline{\Phi}_i(x_{t-d-i}) \\ \overline{\Phi}_{i0}(x_{t-d-i}) u_{t-d-i} \\ \overline{\Phi}_{i1}(x_{t-d-i}) u_{t-d-i+1} \\ \mathrm{M} \\ \overline{\Phi}_{ii}(x_{t-d-i}) u_{t-d} \end{bmatrix}$$

$$W_t = \left[\overline{F}_i^T \quad \overline{G}_{i0}^T \quad \overline{G}_{i1}^T \quad \mathrm{K} \quad \overline{G}_{ii}^T\right]^T \quad (36)$$

Based on the recursive least squares algorithm, an on-line weight learning algorithm is developed for affine nonlinear predictors. The algorithm is given by the following theorem.

THEOREM: Consider the i^{th} predictor and the learning algorithm [1]:

$$W_t = W_{t-1} + \alpha_t \beta_t P_{t-1} \Phi_t e_t$$

$$P_t = P_{t-1} - \beta_t \gamma_t P_{t-1} \Phi_{t-1} \Phi_{t-1}^T P_{t-1}$$

$$\alpha_t = \left(1 - \delta |e_t|^{-1}\right)\left(1 + \Phi_{t-1}^T P_{t-1} \Phi_{t-1}\right)^{-1}$$

$$\gamma_t = \left(|e_t| - \delta\right)\left(|e_t| + \left(2|e_t| - \delta\right)\Phi_{t-1}^T P_{t-1} \Phi_{t-1}\right)^{-1}$$

$$e_t = y_t - W_{t-1}^T \Phi_{t-1} \quad (37)$$

$$\beta_t = \begin{cases} 1, & |e_t| > \delta \\ 0, & |e_t| \le \delta \end{cases} \quad (38)$$

Then (i) $\lim_{t\to\infty} \dfrac{\beta_t (|e_t| - \delta)^2}{1 + \Phi_{t-1}^T P_{t-1} \Phi_{t-1}} = 0$

(ii) $\lim_{t\to\infty} |W_t - W_{t-1}| = 0$

(iii) $\left\|\tilde{W}_t\right\|_2 \le \sqrt{\dfrac{\lambda_{\max}(P_0^{-1})}{\lambda_{\min}(P_0^{-1})}} \left\|\tilde{W}_0\right\|_2$

Where, $\tilde{W}_t = W^* - W_t$

$\lambda_{\max}(.)$ and $\lambda_{\min}(.)$ denote the maximum and the minimum eigenvalues of the matrix (.),

respectively, and W^* is the optimal estimate of the weight vector W_t .

Property (i) of the theorem above shows that if $1+\Phi_{t-1}^T P_{t-1}\Phi_{t-1}$ is finite for all time, which is true if the closed-loop system is stable, the estimation error e_t converges to δ . Also, it can be seen from property (ii) that the weights converge as time *t* approaches infinity. In addition, Property (iii) implies that the weights will never drift to infinity over time.

7. SIMULATION RESULTS OF PREDICTIVE CONTROL IN THE HEAT-EXCHANGER WITH USE OF LOLIMOT

The set point tracking results of the simulation on the plant is depicted in Figure 5 .One and two step ahead predictors are depicted in figure 6 as well. Clearly the system could track the set points with satisfactory performance. To implement the algorithm, 3 neurons have been used and the prediction and control horizon are both set at 2 with the control weight which is equal to 2 and

$$x(t) = [y(t), y(t-1), y(t-2), u(t-1)]$$

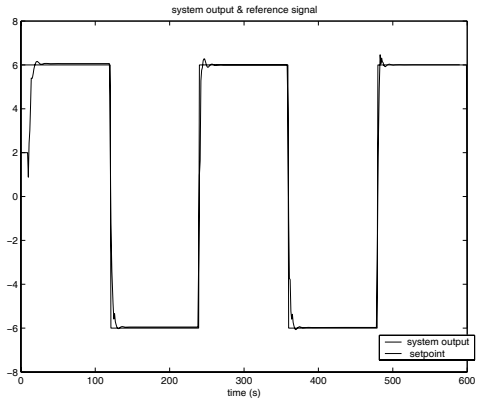

Figure (5): *System output and reference signal*

Based upon the above simulations, the following table is presented and we can conclude that LOLIMOT algorithm provides a better performance.

It can be concluded from the table that the mean square error and overshoot percentage has been significantly decreased in LOLIMOT comparing with MLP resulting from the fact that in LOLIMOT an analytical optimization is used however a numerical optimization is used in MLP.

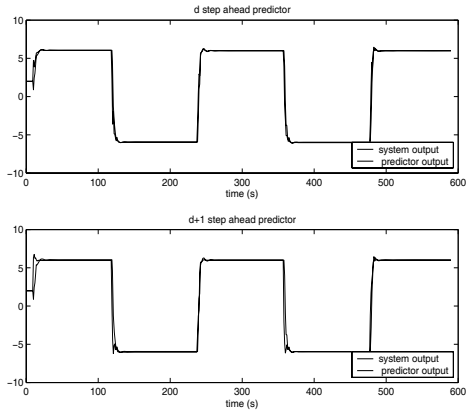

Figure (6): *1 and 2 step ahead predictors*

Table (1): *Comparison between MLP and LOLIMOT*

Method	Over shoot percentage	Settling time
MLP	%7.18	9
LOLIMOT	%2.33	14

8. CONCLUSIONS

A neural network based predictive control strategy was applied to a nonlinear chaotic discrete system; therefore, a nonlinear prediction method, e.g. neural network based methods, should be a better match in a predictive control strategy. To suppress the chaotic behavior an internal feedback control based on time delayed state feedback is proposed. Using the neuro predictive controller, the output of the plant tracked the desired set points by applying the control signal. A neural network model for the plant was constructed. Once having such a model, *i*-step ahead predictions were obtained and a quadratic form cost function was utilized to compute the prediction error and to derive the optimal predictive control strategy. The performance of the proposed control strategy was compared with that of LOLIMOT strategy when dealing with the tracking problem of output, simulation results showed that the later strategy performs much better than the former one in case of mean square error and the percent overshoot.

Acknowledgments

This work was supported by the complex system research center in K. N. Toosi University of Technology.

REFERENCES

[1] Camacho, E.F. Model predictive control, Springer Verlag, 1998.

[2] Garcia, C.E., Prett, D.M., and Morari, M. Model predictive control: theory and practice- a survey, *Automatica,* 25(3), pp.335-348, 1989.

[3] Badgwell, A.B., Qin, S.J. Review of nonlinear model predictive control applications, *In Nonlinear predictive control theory and practice, Kouvaritakis, B, Cannon, M (Eds.),* IEE Control Series, pp.3-32, 2001.

[4] Parker, R.S., Gatzke E.P., Mahadevan, R., Meadows, E.S., and Doyle, F.J. Nonlinear model predictive control: issues and applications, *In Nonlinear predictive control theory and practice, Kouvaritakis, B, Cannon, M (Eds.),* IEE Control Series, pp.34-57, 2001.

[5] Babuska, R., Botto, M.A., Costa, J.S.D., and Verbruggen, H.B. Neural and fuzzy modeling on nonlinear predictive control, a comparison study, *Computatioinal Engineering in Systems Science*, July, 1996.

[6] Nelles, O. Nonlinear system identification: from classical approach to neuro-fuzzy identification, Springer Verlag, 2001.

[7] Narendra, K. S., and Parthasarathy, K., Identification and control of dynamic systems using neural networks. *IEEE Transactions on Neural Networks*, 1, pp.4–27, 1990.

[8] Arahal, M.R., Berenguel, M., and Camacho, E.F. Neural identification applied to predictive control of a solar plant, *Con. Eng. Prac*. 6(3), pp.333-344, 1998.

[9] Lennox, B., and Montague, G. Neural network control of a gasoline engine with rapid sampling, *In Nonlinear predictive control theory and practice, Kouvaritakis, B, Cannon, M (Eds.)*, IEE Control Series, pp.245-255, 2001.

[10] Petrovic, I., Rac, Z., and Peric, N. Neural network based predictive control of electrical drives with elastic transmission and backlash, *Proc. EPE2001*, Graz, Austria, 2001.

[11] Tan, Y. and Cauwenberghe, A. Non-linear one step ahead control using neural networks: control strategy and stability design, *Automatica* 32(12), pp.1701-1706, 1996.

[12] Temeng, H., Schenelle, P. and McAvoy, T. Model predictive control of an industrial packed bed reactor using neural networks, *J. Proc. Control* 5(1), pp.19-28, 1995.

[13] Zamarrano, J.M., Vega, P. Neural predictive control. Application to a highly nonlinear system, *Engineering Application of Artificial Intelligence*, 12(2), pp.149-158, 1999.

[14] Draeger, A., Engel, S., and Ranke, H., Model predictive control using neural networks, *IEEE Control System Magazine*, 15, pp.61–66, 1995.

[15] Gomm, J. B., Evans, J. T., and Williams, D., Development and performance of a neural network predictive controller. *Control Engineering Practice*, 5(1), pp.49–60, 1997.

[16] Diyaz, G., Sen, M., Yang, K.T., McClain, R.L., Simulation of heat exchanger performance by artificial neural networks, *Int. J. HVAC and R Res.,* 5 (3), pp.195-208, 1999.

[17] Ayoubi, M., Dynamic multi-layer perception networks: application to the nonlinear identification and predictive control of a heat exchanger, *in: Applications of Neural Adaptive Control Technology, World Scientific series in Robotics and Intelligent Systems*, 17, pp.205-230, 1997.

[18] Renotie, C., Wouwer, A.V. and Remy, M. Neural modeling and control of a heat exchanger based on SPSA techniques, *Proc. American Control Conference*, Illinois, pp.3299-3303, 2000.

[19] Bittanti, S. and Piroddi, L. Nonlinear identification and control of a heat exchanger: a neural network approach, *Journal of the Franklin Institute*, 1996.

[20] Lim, K.W. and Ling, K.V. generalized predictive control o*f a heat exchanger,* IEEE Control System Magazine, pp.9-12, 1989.

[21] Parte, M.P. and Camacho, E.F. Application of a predictive sliding mode controller to a heat exchanger, *Proc. IEEE Conf. Control Application*, Glasgow, Scotland, pp.1219-1224, 2002.

[22] Skrijave, I. and Matko, D. Predictive functional control based on fuzzy model for heat exchanger pilot plant, *IEEE Trans. Fuzzy Systems*, 8(6), pp.705-812, 2000.

[23] Nelles, O. and R. Isermann (1996). Basis function networks for interpolation of local linear models. In: *IEEE Conference on Decision and Control (CDC)*. Kobe, Japan. pp. 470–475.

[25] Liu, G.P., 2001, Nonlinear Identification and Control: A Neural Network Approach, Springer.

[26] Hunt, K.J., Sbarbaro, D., Zbikowski, R., Gawthrop, P.J., (1992). Neural networks for control system—A survey. *Automatica, 28*, pp.1083–1112.

[27] Takahashi, Y., (1993). Adaptive predictive control of nonlinear time varying system using neural network, *in Proc. IEEE International Symposium on Neural Networks*, pp.1464–1468.

[28] Montague, G.A., Willis, M.J., Tham, M.T., Morris, A.J., (1991). Artificial neural network based control. *International Conference on Control*, pp.266–271.

[29] Chen, G. and Moiola, J. L. An overview of bifurcation, chaos, and nonlinear dynamics in nonlinear system, J. Franklin Inst., vol. 331B, pp. 819-858, 1994.

[30] Ogorzalek, M. J. Taming chaos, Part two: control, IEEE Trans. Circuits Syst. I, vol. 40, pp. 700-706, 1993.

[31] Chen, G. and Done, X. From chaos to order: methodologies, perspectives and applications, World Scientific, Singapore, 1998.

[32] Fradkov, A. L. and Pogromsky, A. Y. Introduction to control of oscillations and chaos, World Scientific, Singapore, 1998.