A NEW DEVELOPMENT OF ADAPTIVE MODEL PREDICTIVE CONTROL

D. L. Yu*, D. W. Yu, J. B. Gomm* and G.F. Page***

** Control Systems Research Group, School of Engineering,*
Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, U.K.
*** Department of Automation, Northwestern University at Qinhuangdao,China*
Email: d.yu@livjm.ac.uk

Abstract: An adaptive radial basis function (RBF) neural network model is developed in this paper for nonlinear systems using the recursive orthogonal least squares (ROLS) algorithm. The model is used in a nonlinear model predictive control (NMPC). The developed adaptive NMPC is applied to a chemical reactor rig. On-line control performance is presented and it demonstrates superiority over the fixed parameter PID control. *Copyright © 2005 IFAC*

Keywords: Adaptive RBF network, NMPC, adaptive control, ROLS algorithm.

1. INTRODUCTION

It is realized that a fixed parameter neural network model trained off-line often generate an intolerable modeling error when it is used to model an industrial process, especially a process with some time-varying parameters or with considerable uncertainties. Therefore, different adaptive neural networks have been developed in recent years. For example, Yingwei *et al*. [1] and Luo and Billings [2] proposed different adaptation algorithms for RBF network structures to recursively train the network model. However, these models were mainly developed for system identification and were not control oriented. Although the model is finally trained to meet the requirement, the prediction performance of these models during the training are not considered, especially when the new centers are added. The reason is that the new centers have not been trained by the previous measurement data and the training of these centers is recursively with the new data starting from the initial condition. Liu *et al*. [3] developed an adaptive RBF network with Lyapunov method. Pereira *et al*. [4] applied an adaptive RBF network model in the internal model control strategy to control an experimental process, and compared the performance with that achieved using a linear pole-placement controller.

A novel method for RBF model adaptation is proposed in this paper. The objective of the method is to develop an adaptive model to be used in model-based control scheme. Hence, the model must have a smooth accurate prediction no matter which region the operating point is moved to. The proposed model is adapted in both structure and parameters. The advantage of the adaptation method developed in this paper is that the active centers used in prediction are chosen on-line from a center bank and the weights associated to the centers in the bank are on-line updated. In this way, the degradation of the model accuracy caused by adding or pruning centers is reduced to a minimum. A backward center selection method proposed by the authors in the previous research [5] is used for on-line selection of the active centers. The number of currently active centers is determined by a measure of center pruning error that gives a best compromise between the prediction accuracy and the model complexity. The developed adaptive model is evaluated by modeling a reactor rig and is also applied in a model predictive control scheme for on-line evaluation.

2. ADAPTIVE RBF MODEL

2.1 Model Structure

The nonlinear system to be modeled is represented by the multivariable NARX model of the following form,

$$y(k) = f[y(k-1),\cdots, y(k-n_y), u(k-1-d), \cdots, u(k-n_u-d)] + e(k) \quad (1)$$

where $u \in \Re^m$, $y, e \in \Re^p$ are the process input, output and noise vectors respectively with *m* and *p* being the number of inputs and outputs, n_y and n_u are the maximum lags in the outputs and inputs respectively, *d* is a dead-time vector representing delayed time to different control variables, $f(*)$ is a vector-valued non-linear function. The Gaussian function output φ in each hidden layer node is given by

$$\varphi_i = \exp(-\frac{\|x - c_i\|^2}{\sigma_i^2}), \quad i = 1,\cdots, n_h \quad (2)$$

where c_i is the i^{th} center vector of the same dimension as the input vector $x \in \Re^{n_i}$, σ_i is the i^{th} width which is a scalar, n_h is the number of hidden layer nodes. Thus, the network output vector $\hat{y}$ *is* given by

$$\hat{y} = \phi^T W \quad (3)$$

where $W \in \Re^{n_h \times p}$ is a weight matrix. When a RBF neural network is used to model a process of the NARX form given in (1), it is actually to approximate the static non-linear mapping of *f*(*) in (1) while the system dynamics are realized by the external feedback of delayed outputs and the delayed inputs. Therefore, it is natural to choose neural network inputs the same as the variables involved in the NARX model (1) and the neural network model represents the following equation.

$$\hat{y}(k) = \hat{f}[x(k)] \quad (4)$$

where

$$x(k) = [y(k-1),\cdots, y(k-n_y), u(k-1-d), \cdots, u(k-n_u-d)] \quad (5)$$

is the network input vector.

2.2 Training Algorithm

Training of the adaptive RBF network has two steps, the first step is the off-line training of the initial model and the second step is the on-line training. In the off-line training, a set of network centers are chosen to form the center bank using the *K*-means clustering algorithm from a set of real data that covering the frequency band and the amplitude range for the entire operating space. The width for each center is determined using the *p*-nearest-center rule but chose a bigger *p* for uniform sampling, say *p* = *3~5*. Then, the backward center selection method [5] that is based on the orthogonal least squares training algorithm is used to select significant centers from the center bank.

In the on-line training, the R matrix associated to the centers n the bank is updated with the new measurement using the ROLS algorithm. Then, if two conditions are satisfied a smaller number of significant centers will be chosen using the pruning method. These centers are used to do model prediction in this sample time. The least squares problem is formulated as follows. Considering (1) at sample interval *k* for a set of *N* samples of input-output training data from $k-N+1$ to *k*, in other words a window going back in time N samples, we have

$$Y(k) = \hat{Y}(k) + E(k) = \Phi(k)W(k) + E(k) \quad (6)$$

where $Y \in \Re^{N \times p}$ is the desired output matrix, $\hat{Y} \in \Re^{N \times p}$ is the neural network output matrix, $\Phi \in \Re^{N \times n_h}$ is the hidden layer output matrix, $E \in \Re^{N \times p}$ is the error matrix and equation (6) can be solved for $W(k)$ using the recursive MIMO Least Squares algorithm to minimize the following time-varying cost function,

$$J(k) = \left\| \begin{bmatrix} \sqrt{\lambda} Y(k-1) \\ ---- \\ y^T(k) \end{bmatrix} - \begin{bmatrix} \sqrt{\lambda}\Phi(k-1) \\ ---- \\ \phi^T(k) \end{bmatrix} W(k) \right\|_F \quad (7)$$

where the *F*-norm of a matrix is defined as $\|A\|_F^2 = trace(A^T A)$ and $\lambda < 1$ is used to introduce exponential forgetting to the past data. It has been shown [5] that minimizing (7) is equivalent to minimizing the following cost function,

$$J(k) = \left\| \begin{bmatrix} \sqrt{\lambda} \hat{Y}(k-1) \\ ---- \\ y^T(k) \end{bmatrix} - \begin{bmatrix} \sqrt{\lambda} R(k-1) \\ ---- \\ \phi^T(k) \end{bmatrix} W(k) \right\|_F \quad (8)$$

where *R* is an $n_h \times n_h$ upper triangular matrix, and $\hat{Y}$ is computed by an orthogonal decomposition as follows,

$$\begin{bmatrix} \sqrt{\lambda} R(k-1) \\ ---- \\ \phi^T(k) \end{bmatrix} = Q(k) \begin{bmatrix} R(k) \\ --- \\ 0 \end{bmatrix}, \quad \begin{bmatrix} \hat{Y}(k) \\ --- \\ \eta^T(k) \end{bmatrix} = Q^T(k) \begin{bmatrix} \sqrt{\lambda}\hat{Y}(k-1) \\ ---- \\ y^T(k) \end{bmatrix} \quad (9)$$

where *Q* is an orthogonal matrix. Combining (8) and (9) and considering that the *F*-norm is preserved by orthogonal transformation, the following equivalent cost function is obtained,

$$J(k) = \left\| \begin{bmatrix} \hat{Y}(k) - R(k)W(k) \\ --------- \\ \eta^T(k) \end{bmatrix} \right\|_F \quad (10)$$

which allows the optimal solution of $W(k)$ to be solved straightforwardly from

$$R(k)W(k) = \hat{Y}(k) \quad (11)$$

and leaves the residual at sample interval *k* as $\|\eta^T(k)\|_F$. Since $R(k)$ is an upper triangular matrix,

$W(k)$ can be easily solved from (11) by backward substitution.

The decomposition in (9) can be achieved efficiently by applying Givens rotations to an augmented matrix to obtain the following transformation [4],

$$\begin{bmatrix} \sqrt{\lambda}R(k-1) & \sqrt{\lambda}\hat{Y}(k-1) \\ \phi^T(k) & y^T(k) \end{bmatrix} \rightarrow \begin{bmatrix} R(k) & \hat{Y}(k) \\ 0 & \eta^T(k) \end{bmatrix} \quad (12)$$

The procedure of the ROLS algorithm is therefore the following: for on-line training, calculate $\phi(k)$ at each sampling period to update the augmented matrix and compute the Givens rotations to realize the transformation in (12). Then solve $W(k)$ in (11) with $R(k)$ and $\hat{Y}(k)$ obtained in (12). In this case, $W(k)$ is needed at each sample instant for prediction. Also, $\lambda < 1$ is needed to follow time-varying dynamics at the current time. For use in off-line mode, the Givens rotations can be computed to realize the transformation in (12) continuously to the end of training, then W is solved finally from (11). In this case, λ is set to 1. Initial values for $R(k)$ and $\hat{Y}(k)$ in both cases can be assigned as $R(0) = \mu I$ and $\hat{Y}(0) = 0$, where μ is a small positive number and I is a unity matrix with appropriate dimension.

An important extension of the algorithm is that it can be used to evaluate the significance of each center in contributing to the model output, so as to determine which centers in the pre-defined center bank should be used in the current working condition to model the current system dynamics. The method can be briefly outlined as follows [5]. When a column of $R(k)$ in the augmented matrix $[R(k)\ \hat{Y}(k)]$ in (12) is removed, the remaining matrix is re-triangularized using Givens rotation as shown in Fig.2. The resulting $\|\eta^T(k)\|$ is the loss of F-norm caused by pruning the corresponding center in the network and therefore it is the contribution of the center to the network output.
When each center is removed in turn from the original R matrix and $\|\eta_j^T(k)\|$ for $j = 1, \cdots, n_h$ is calculated, the center with minimum $\|\eta_j^T(k)\|$ is the least significant one and its corresponding column is moved to the last column of R. This procedure is repeated until all the centers are rearranged in order of significance. Then, one can choose the most significant centers of number n_h which can be chosen for a best trade off between modeling accuracy and network complexity using the Akaike's FPE rule. Finally, the optimal weights corresponding to the pruned network can be calculated from the augmented R matrix and they are equivalent to that which would be produced if these centers were not used at the beginning.

2.3 Network Adaptation

The RBF models are adapted on-line to accommodate these changes. In addition to adaptation of the neural model weights, the structure is also adapted by changing the RBF centers. This is particularly useful when a plant is subject to a large disturbance or an actuator fault, as in these cases the distribution of the model input data will significantly change and adaptation of the weights only will not be enough.
The backward center selection method described in [Gomm and Yu, 2000] was extended to on-line model adaptation. As the network centers may not all be active from time to time in producing an output, a smaller set of active centers will be selected on-line and used for prediction. This is supported by our neural network modeling experience that the less neurons used in the network model, the more accurate the model generalization will be. These centers are selected by evaluating their contribution to the network output using the backward selection method.

The pre-selected center set, chosen off-line, is trained on-line with current input-output data to track the time varying dynamics of the process. When the process under controlled is not injected with an additional excitation signal, e.g. superimposed to the control signal, the PE condition can only be satisfied when set points change or the system is subject to disturbance or excessive system noise. To avoid that the model is on-line trained by input/output data not containing enough dynamic information, and also to reduce the computing load, the following two criteria are introduced to determine if the on-line adaptation is actually to be done. Condition (13) tests the modeling error to indicate if the network needs to be updated,

$$\frac{1}{M}\sum_{i=k-M+1}^{k}[y(i)-\hat{y}(i)]^2 > e_{Tol1} \quad (13)$$

Condition (17) tests the input data changes to indicate if the data is in a steady state with constant output.

$$[x(k)-x(k-1)]^T[x(k)-x(k-1)] > e_{Tol2} \quad (14)$$

where x is the input vector of the neural model. Condition (13) implies that changes in the plant dynamics cause an intolerable modeling error for a few consecutive steps, while condition (14) means there exists a change in at least one of the model input variables. The model will not be on-line trained unless the two conditions are satisfied simultaneously. It has been observed in control simulations and on-line control experiments that the neural model is not updated when the system is in steady state and without disturbance or changes in dynamics.

After the center set is updated in the sample period, next is to choose the active centers to be used in actual model prediction. This is realized using the backward center pruning method reviewed in the last section. By removing one center in turn to evaluate the resulting residual, the center contributing the least to the model output is identified. Removing this center and repeating the procedure, the center with second least contribution is identified. In this way, all

centers are re-ordered from the most contribution to the least. Then, a number of centers are chosen as active centers. To automatically determine the number of active centers, a trade-off between the modeling accuracy and the model complexity is needed, in the light that a network with fewer centers will give better generalization. Akaike's *FPE* is then used,

$$FPE(i) = \frac{N + n_p}{N - n_p}\varepsilon(i) \qquad (15)$$

where n_p is the number of adjusting parameters (weights) and i is the number of centers removed. N is the number of data samples that is known for off-line training, here it is approximated by

$$N = \frac{1}{1-\lambda_0} \qquad (16)$$

$\varepsilon(i)$ in (15) is an index of modeling error, which is updated from a zero initial value each time of pruning a center by the resulting residual as follows,

$$\varepsilon(i) = \lambda_o \varepsilon(i-1) + \|\eta(i)\|_F^2 / N \qquad (17)$$

While $\varepsilon(i)$ increases as i increases, $FPE(i)$ will decrease and reach a minimum. The pruning stops at this point and the remaining centers are the active centers to be used for model prediction. The on-line implementation procedure is given as be low.

Step 1. Collect new data $y(k)$ and $u(k)$ to form $x(k)$. Then check if both (13) and (14) are satisfied. If yes, go on to the next step. Otherwise, do not adapt the model.

Step 2. Apply Given's rotation to (12).

Step 3. Re-order the centers in the set from greatest to least contributions.

Step 4. Calculate FPE to determine the number of active centers using (15)-(17).

Step 5. Update the weights of the active center set using (11).

Step 6. The active centers are used in model prediction.

3. MODELING A REACTOR

The reactor used in this research shown in Fig.1 is a pilot process representing the dynamic behavior of real chemical processes in industry. The variables to be controlled in the reactor are temperature, *pH* and dissolved oxygen.

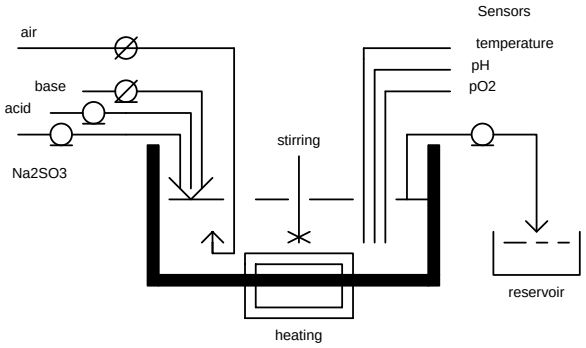


Fig.1 The schematic of the chemical reactor process

Process inputs and outputs are chosen as $u = [Q \quad f_b \quad f_a]^T$, $y = [T \quad pH \quad pO_2]^T$ where Q, f_b and f_a denote the heating power, the flow rate of the base and the flow rate of air respectively. The working space for the process output is

$$35\,^{\circ}C < T(t) < 55\,^{\circ}C, \quad 5.5 < pH(t) < 8.5, \quad 20\% < pO_2(t) < 80\% \qquad (18)$$

The constraints to the manipulated variables are

$$0 < Q(t) < 600\,W, \quad 0 < f_b(t) < 100\,ml/\min, \quad 0 < f_a(t) < 15\,l \qquad (19)$$

The three network models chosen in this research are

$$\hat{T}(k) = \hat{f}_T[\hat{T}(k-1), \hat{T}(k-2), Q(k-22), f_a(k-1)] \qquad (20)$$

$$\hat{pH}(k) = \hat{f}_{pH}[\hat{pH}(k-1), \hat{pH}(k-2), \hat{T}(k-1), f_b(k-1)] \qquad (21)$$

$$\hat{pO_2}(k) = \hat{f}_{pO_2}[\hat{pO_2}(k-1), \hat{pO_2}(k-2), \hat{T}(k-1), \hat{pH}(k-1), f_a(k-1)] \qquad (22)$$

Two data sets are collected from the reactor. Firstly, the three RBF models (20)-(22) are used. Three center banks with 25, 25 and 30 centers for temperature, *pH* and dissolved oxygen respectively are selected off-line using the K-means clustering method and the backward selection method using data set 1. Then, on-line adaptation of the developed model is evaluated using data set 2 for multi-step-ahead prediction. The predictions are compared with that produced by the same models with centers and weights fixed. When using adaptive models, the active centers are chosen from the three center banks after the R matrix is updated, and used for 1, 8 and 20-step-ahead predictions. In the experiment, parameters used in adaptation are chosen as $\lambda = 0.99$, $N = 1/(1-\lambda) = 100$, $M = 5$, the scaled value of error tolerance in (13) is set to the equivalent non-scaled value $e_{Tol1} = [0.0018 \quad 0.000017 \quad 0.05]^T$ and the scaled tolerance in

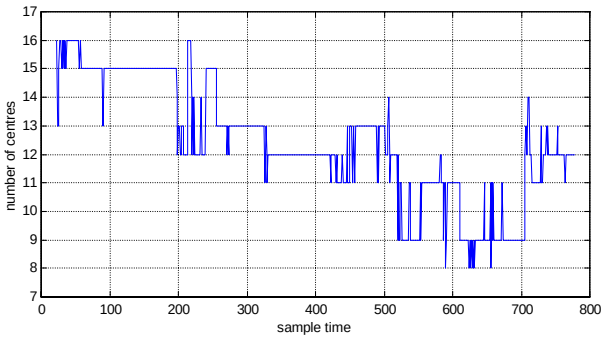


Fig.2 Center variation in adaptive model

(14) as $e_{Tol2} = [0.0025 \quad 0.0025 \quad 0.0025]^T$. The fixed model for the dissolved oxygen has 12 centers, being the average of the adaptive centers. The predictions of the adaptive networks are found similar to that of the fixed networks for temperature and *pH* because these two variables are not significantly time varying. Therefore, only the performance for dissolved oxygen is presented here. The center variation out of 25 candidates for dissolved oxygen is displayed in Fig.2.

8-step-ahead predictions by the adaptive and fixed models are displayed with the real process output in Fig.3 for comparison.

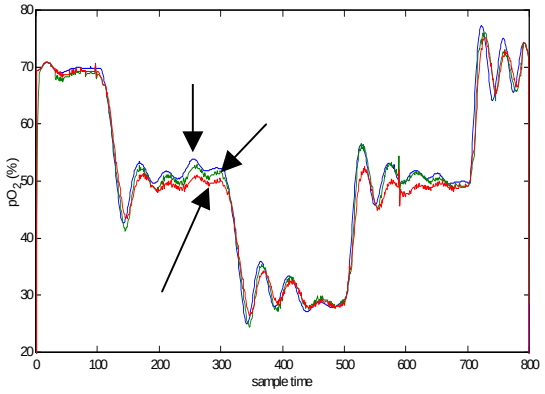


Fig.3 8-step-ahead predictions

From Fig.3 it can be clearly seen that the multi-step prediction of the adaptive model is much closer to the real output than that of the fixed-parameter model. The performance of both models are also assessed by the MSE defined in (23) and the MSEs of the dissolved oxygen of the adaptive and non-adaptive models are listed in Table I.

$$e_{MSE} = \frac{1}{N}\sum_{k=1}^{N}[y(k) - \hat{y}(k)]^2 \qquad (23)$$

Table I. Prediction errors

MSE	1-step	8-step	20-step
Adaptive model	0.033	2.22	22.9
Fixed model	0.041	5.87	39.1

4. ADAPTIVE NMPC

The neural network model based NMPC scheme developed in this research is shown in Fig.4.

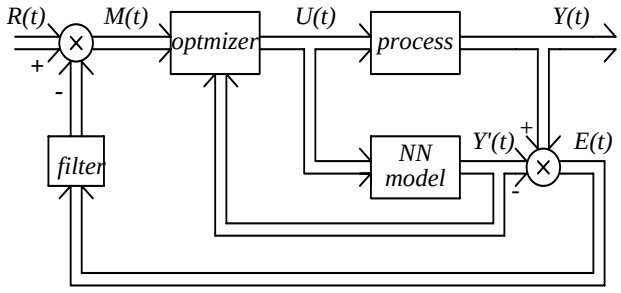


Fig.4 Multivariable MPC control structure

The cost function in (24) has a normal quadratic form but with the predictive horizon not from the current sample time.

$$J(k) = \sum_{i=1}^{p}(r_{M(i)} - \hat{y}_i)^T W_i^y (r_{Mi(i)} - \hat{y}_i) + \sum_{j=1}^{m}\xi(j)\Delta u_j^T W_j^u \Delta u_j \qquad (24)$$

with $r_{M(i)} = r_i - e_i$ being the modified set point vector and e_i being the filtered model prediction error vector, and

$$r_i^T = [r_i(k+N_1(i)),\cdots,r_i(k+N_2(i))], \quad i=1,\cdots p \qquad (25)$$

$$\hat{y}_i^T = [\hat{y}_i(k+N_1(i)),\cdots,\hat{y}_i(k+N_2(i))], \quad i=1,\cdots,p \qquad (26)$$

$$\Delta u_j^T = [u_j(k)-u_j(k-1),\cdots;u_j(k+N_u(j))-u_j(k+N_u(j)-1)], \quad j=1,\cdots,m \qquad (27)$$

where $\hat{y}$ is the neural model output vector and u is the control vector, subscripts i or j denotes the i^{th} or j^{th} entry of the vectors, $N_1, N_2 \in \Re^p$ are two vectors with their i^{th} elements specifying the starting and ending sample instants for the i^{th} tracking error to be evaluated, while $N_u \in \Re^m$ is a vector with its j^{th} element specifying the control horizon for the j^{th} control variable.

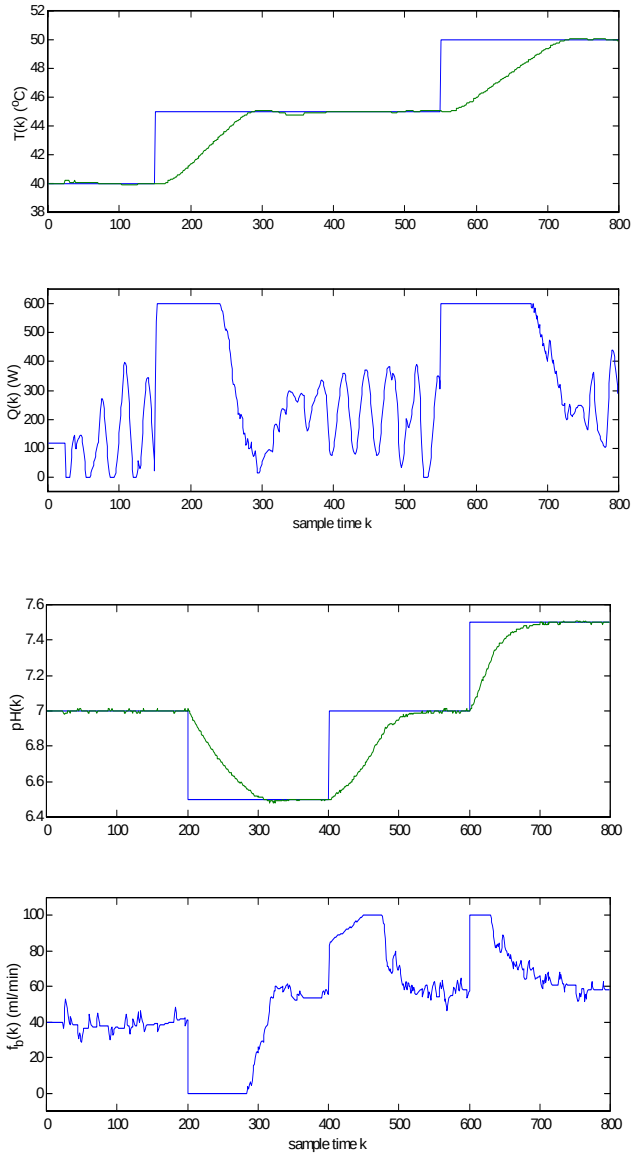
The control parameters used in on-line control are set initially to that tuned in the simulation and are further tuned on-line to the following values:
$N_1 = [d_\Omega \quad 2 \quad 3]^T$, $d_\Omega = 20$, $N_2 = [d+20 \quad 18 \quad 12]^T$, $N_u = [1 \quad 1 \quad 1]^T$, $\xi = [0.1 \quad 0.1 \quad 0.1]^T$, $W_y = W_u = I_3$.
To test effectiveness of the NMPC scheme for rejecting disturbance, two step-disturbances as in (28)-(29) are used as the two flow rates of the inlet of Na_2SO_3 and CH_3COOH respectively.

$$f_c(k) = \begin{cases} f_c & k<300 \\ f_c*(1+50\%) & k\geq 300 \end{cases} \quad f_n(k) = \begin{cases} f_n & k<400 \\ f_n*(1+50\%) & k\geq 400 \end{cases}$$

The on-line performance of the ANMPC and the corresponding control variables are displayed in Fig.5. It can be seen that the performance in both reference tracking and disturbance rejection are very good.

5. CONCLUSIONS

A model adaptation method for the RBF network is developed with both structure and weight on-line updating. The weight updating for the center set can keep the model refreshed by the current system measurements. The active centers are selected on-line from the center set rather than recruited with zero initial condition, which greatly improves the consistency of the model accuracy. The adaptive RBF model is used in MPC and on-line evaluation shows an improved performance.



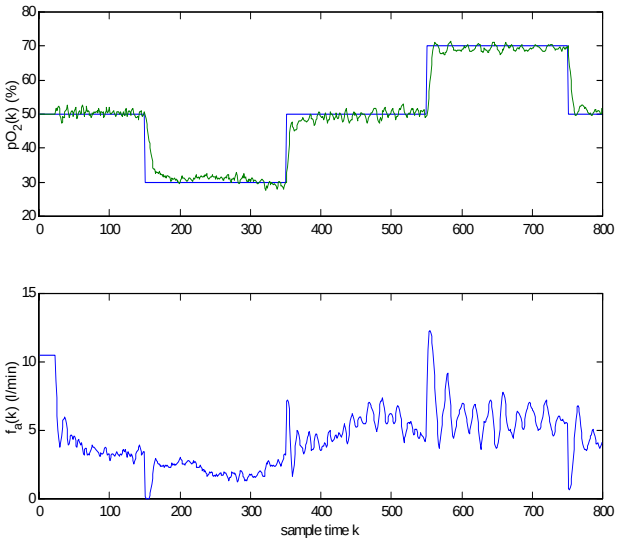


Fig.5 On-line performance of the ANMPC

REFERENCES

L. Yingwei, N. Sundararajan and P. Saratchandran, "Identification of time-varying nonlinear systems using minimal radial basis function neural networks," *IEE Proc. Part. D*, Vol.144, No.2, pp. 202-208, 1997.

W. Luo and S.A. Billings, "Structure selective updating for nonlinear models and radial basis function networks," *Int. J. Adap. Cont. Sign. Proces.*, Vol.12, pp.325-345, 1998.

G.P. Liu, V. Kadirkamanathan and S.A. Billings, "Variable neural networks for adaptive control of nonlinear systems," *IEEE Trans. Sys. Man. Cyber*. Pert. C, Vol.29, No.1, pp. 34-43, 1999.

C. Pereira, J. Henriques and A. Dourado, "Adaptive RBFNN versus conventional self-tuning: comparison of two parametric model approaches for non-linear control," *Control Engn. Prac.*, Vol.8, No.1, pp. 3-12, 2000.

J.B. Gomm and D.L. Yu, "Selecting redial basis function neural network centers with recursive orthogonal least squares training," *IEEE Trans. Neural Networks*, Vol.11, No.2, pp. 306-314, 2000.