Adaptive Neural Network Predictive Control Based on PSO Algorithm

Chengli Su, Yun Wu
School of Information and Control Engineering, Liaoning Shihua University, Fushun China 113001

E-mail: sclwind@sina.com

Abstract: A neural network-based model predictive control scheme is proposed for nonlinear systems. In this scheme an adaptive diagonal recurrent neural network (DRNN) is used for modeling of nonlinear processes. A recursive estimation algorithm using the extended Kalman filter (EKF) is proposed to calculate Jacobian matrix in the model adaptation so that the algorithm is simple and converges fast. Particle swarm optimization (PSO) is adopted to obtain optimal future control inputs over a prediction horizon, which overcomes effectively the shortcoming of descent-based nonlinear programming method on the initial condition sensitivity. A case study of biochemical fermentation process shows that the performance of the proposed control scheme is better than that of PI controller.

Key Words: Model predictive control (MPC), Diagonal recurrent neural network (DRNN), Particle swarm optimization (PSO)

1 INTRODUCTION

Model predictive control (MPC) is known to be a very powerful control strategy for a variety of industrial processes [1]. The attraction for MPC is due to its ability to handle various constraints directly in the formulation through on-line optimization. Linear model predictive control (LMPC), a widely recognized method, has been successfully applied to many industrial processes. A comprehensive survey on MPC industrial applications can be found in [2]. However, many practical processes are inherently strong nonlinearities. Linear model is not adequate to describe the process dynamics. Under such condition, predictive control based nonlinear model (NMPC) should be considered. NMPC retains the advantages of MPC such as the use of an explicit model to predict the behavior of a process over a future horizon and implementation of control action that guides the process to follow the predetermined objective in an optimal manner. Although, NMPC meets the control requirement of many nonlinear processes, most of NMPC are based on physical process models whose systematic development is difficult for complex nonlinear processes. NMPC technique can be applied to a wide range of nonlinear processes if suitable empirical models can be identified from plant input-output data. Neural networks (NN), having an inherent ability to approximate an arbitrary nonlinear function, have become attractive means for modeling and control of nonlinear processes. Several neural networks based predictive control (NNMPC) algorithms for nonlinear processes have been proposed. A new recurrent neural-network feedback predictive control strategy for a class of uncertain nonlinear dynamic time-delay system is developed and analyzed [3]. A radial basis function neural network (RBFNN) model predictive control scheme is developed for multivariable nonlinear systems [4]. A fast convergence algorithm is proposed and employed in multidimensional optimization in control scheme to reduce the computing effort. Although research on NNMPC has obtained much attention, most previous works are based on nonlinear programming method which provides local optimum values only and in addition these values depend on the selection of the starting point. In this paper a PSO-based neural network predictive control scheme is proposed for nonlinear systems. An adaptive DRNN is used to model nonlinear process. A novel recursive algorithm using the extended Kalman filter (EKF) is proposed to calculate Jacobian matrix in the model adaptation so that the algorithm is simple and converges fast. PSO technique is adopted to obtain optimal future control inputs for nonlinear process.

This work is supported by National High Technology Research and Development Program of China (2007AA04Z162)

2 PROCESS MODELING AND ADAPTIVE ALGORITHM

Consider a class of discrete time nonlinear dynamic processes can be described by the following NARX model

$$y(k) = f\,(y(k-1)\,,\,y(k-n_y)\,,\cdots, \\ u(k-1)\,,\,u(k-n_u)) + e(k) \tag{1}$$

where $y(k) \in R^p$ is the process output vector, $u(k) \in R^m$ is the process input vector, n_u and n_y are the input and output orders respectively, $e(k)$ is a zero-mean, white noise vector, and $f(\cdot)$ denotes a vector-valued nonlinear function.

DRNN is simple, learning fast, and doesn't need know the accurate orders of the process. In this paper, a simplest three-layer DRNN is applied for modeling of nonlinear process. The configuration of DRNN is shown in Fig.1.

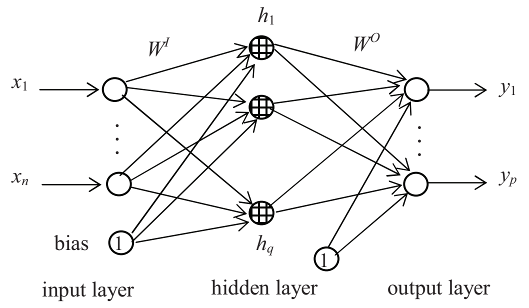


Fig. 1 The structure of DRNN

The neural network model can be expressed as the following form

$$\hat{y}(k) = W^O \begin{bmatrix} h(k) \\ 1 \end{bmatrix} \tag{2}$$

$$h(k) = \frac{1}{1+e^{-s(k)}} \tag{3}$$

$$s(k) = W^I \begin{bmatrix} x(k) \\ 1 \end{bmatrix} + \left[\begin{bmatrix} w_{1,1}^h & & 0 \\ & \ddots & \\ 0 & & w_{1,q}^h \end{bmatrix} \cdots \begin{bmatrix} w_{v,1}^h & & 0 \\ & \ddots & \\ 0 & & w_{v,q}^h \end{bmatrix} \right] \begin{bmatrix} h(k-1) \\ \vdots \\ h(k-v) \end{bmatrix} \tag{4}$$

$$= W^I \begin{bmatrix} x(k) \\ 1 \end{bmatrix} + \begin{bmatrix} \overline{W}_1^h \\ \vdots \\ \overline{W}_v^h \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} h(k-1) \\ \vdots \\ h(k-v) \end{bmatrix}$$

where

$\hat{y}(k) = [\hat{y}_1(k), \cdots, \hat{y}_p(k)]^{\mathrm{T}}$, $x(k) = [x_1(k), \cdots, x_n(k)]^{\mathrm{T}}$, $h(k) = [h_1(k), \cdots, h_q(k)]^{\mathrm{T}}$,

$$W^I = \begin{bmatrix} w_{1,1}^I & \cdots & w_{1,n+1}^I \\ \vdots & \ddots & \vdots \\ w_{q,1}^I & \cdots & w_{q,n+1}^I \end{bmatrix}, W^O = \begin{bmatrix} w_{1,1}^o & \cdots & w_{1,q+1}^o \\ \vdots & \ddots & \vdots \\ w_{p,1}^o & \cdots & w_{p,q+1}^o \end{bmatrix}$$

$$W^h = \begin{bmatrix} w_{1,1}^h & \cdots & w_{1,q}^h \\ \vdots & \ddots & \vdots \\ w_{v,1}^h & \cdots & w_{v,q}^h \end{bmatrix}, \overline{W}_r^h = \begin{bmatrix} w_{r,1}^h & & 0 \\ & \ddots & \\ 0 & & w_{r,q}^h \end{bmatrix}, r = 1, \cdots, v$$

$x(k) \in R^n$ is the network input, $h(k) \in R^q$ is the hidden layer output, $\hat{y}(k) \in R^p$ is the network output, v is the time lags of the feedback vector within the hidden layer, which is used to internally present the dynamics of the process to be modeled. $w_{i,j}^I$ represents the linking weight between the input layer neuron j and the hidden layer neuron i , $w_{l,i}^o$ represents the linking weight between the hidden layer neuron i and the network output layer neuron l , $w_{r,i}^h$ represents the r-th order time delay recurrent linking weight of the hidden layer neuron i , $s(k) \in R^q$ is the input accumulation vector of the hidden layer neuron, $j = 1 \cdots n$, $i = 1 \cdots q+1$, $r = 1 \cdots v$ .Then nonlinear process (1) can be realized with the above DRNN

$$\hat{y}(k) = NN(x(k), W^I, W^h, W^O) \tag{5}$$

where $x(k) = [u(k-1)^{\mathrm{T}} \ y(k-1)^{\mathrm{T}}]^{\mathrm{T}}$, $v = \max\{n_u, n_y\} - 1$, $y(k)$, $u(k)$ are the sampled process output and input at sample instant k respectively, $NN(\cdot)$ is the combination of the relationships presented by (2)~(4).

There are many uncertainties, immeasurable disturbances and so on in practical processes, so that a recursive EKF algorithm [5] is used to on-line update the linking weights of the DRNN model. The weights of the DRNN to be updated and trained are formulated as a parameter vector $\theta \in R^{(q(n+1)+qv+p(q+1))\times 1}$, i.e.

$$\theta = [\theta^I \quad \theta^O]^{\mathrm{T}} \tag{6}$$

$$\theta^I = \begin{bmatrix} [w_1^I & w_{1,1}^h & \cdots & w_{v,1}^h]^{\mathrm{T}} \\ & \vdots & & \\ [w_q^I & w_{1,q}^h & \cdots & w_{v,q}^h]^{\mathrm{T}} \end{bmatrix}, \ \theta^O = \begin{bmatrix} w_1^{o\,\mathrm{T}} \\ \vdots \\ w_p^{o\,\mathrm{T}} \end{bmatrix} \tag{7}$$

where w_i^I is the i-th row vector in W^I , w_i^o is the i-th row vector in W^O . Then the weights updating with the EKF are formulated as the following equations

$$\theta(k+1) = \theta(k) \tag{8}$$

$$y(k) = \hat{y}(k) + e_m(k) \tag{9}$$

where $e_m(k)$ is the modeling error, the parameter is estimated by applying the EKF algorithm

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)(y(k) - \hat{y}(k \mid k-1)) \tag{10}$$

$$E(k) = (I - K(k)C(k))E(k-1) \tag{11}$$

$$K(k) = \frac{E(k-1)C(k)^{\mathrm{T}}}{R(k \mid k-1) + C(k)E(k-1)C(k)^{\mathrm{T}}} \tag{12}$$

with $\hat{y}(k \mid k-1) = NN(\hat{\theta}(k-1), x(k))$, $C(k) = \frac{\partial \hat{y}(k)}{\partial \theta}\Big|_{\theta = \hat{\theta}(k-1)}$

where $R(k \mid k-1) \in R^{p\times p}$ is an unknown priori error covariance matrix. It can be estimated on-line by the following formulas

$$\begin{aligned} R(k \mid k-1) = R(k-1) + [(y(k) - \hat{y}(k \mid k-1)) \\ \times (y(k) - \hat{y}(k \mid k-1))^{\mathrm{T}} - R(k-1)]/k \end{aligned} \tag{13}$$

$$\begin{aligned} R(k) = R(k-1) + [(y(k) - \hat{y}(k)\big|_{\theta=\hat{\theta}(k)}) \\ \times (y(k) - \hat{y}(k)\big|_{\theta=\hat{\theta}(k)})^{\mathrm{T}} - R(k-1)]/k \end{aligned} \tag{14}$$

$C(k) \in R^{p\times(q(n+1)+p(q+1))}$ is the Jacobian matrix of DRNN model output $\hat{y}(k)$ with respect to θ , and is expressed as

$$C(k) = \frac{\partial \hat{y}(k)}{\partial \theta}\Big|_{\theta=\hat{\theta}(k-1)} = \begin{bmatrix} \frac{\partial \hat{y}(k)}{\partial \theta^I} & \frac{\partial \hat{y}(k)}{\partial \theta^O} \end{bmatrix}_{\theta=\hat{\theta}(k-1)} \tag{15}$$

$$\begin{aligned} \frac{\partial \hat{y}(k)}{\partial \theta^I}\Big|_{\theta=\hat{\theta}(k-1)} = \hat{W}^O(k-1) \times \\ \begin{bmatrix} h_1(k)(1-h_1(k) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h_q(k)(1-h_q(k) \end{bmatrix}_{\theta=\hat{\theta}(k-1)} \times \frac{\partial s(k)}{\partial \theta^I}\Big|_{\theta=\hat{\theta}(k-1)} \end{aligned} \tag{16}$$

$$\frac{\partial \hat{y}(k)}{\partial \theta^O}\Big|_{\theta=\hat{\theta}(k-1)} = \begin{bmatrix} [h(k)^{\mathrm{T}} \quad 1] & & 0_{1\times(q+1)} \\ & \ddots & \\ 0_{1\times(q+1)} & & [h(k)^{\mathrm{T}} \quad 1] \end{bmatrix}_{\theta=\hat{\theta}(k-1)} \tag{17}$$

3 PREDICTIVE CONTROL USING PSO ALGORITHM

PSO is a stochastic global optimization method which is based on simulation of social behavior. PSO exploits a population of potential solutions to probe the search space. In contrast to the aforementioned methods in PSO no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead of mutation PSO relies on the exchange of information between individuals, called *particles*, of the population, called *swarm*. In effect, each particle adjusts its trajectory towards its own previous best position, and towards the best previous position attained by any member of its neighborhood. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and particles profit from the discoveries and previous experience of all other companions during the search for promising regions of the landscape. Initially, let us define the notation adopted in this paper: assuming that the search space is D-dimensional, the i-th particle of the swarm is represented by a D-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and the best particle of the swarm, i.e. the particle with the lowest function value, is denoted by index g. The best previous position (i.e. the position corresponding to the best function value) of the i-th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and the position change (velocity) of the i-th particle is $V_i = (v_{i1}, v_{i2}; \ldots, v_{iD})$. All particles start with randomly initialized velocities and positions. Then the d-th component of the new velocity and the new position for the i-th particle are updated by using the following equations:

$$V_{i,d}(t+1) = \lambda * V_{i,d}(t) + c_1 * (P_i(t) - X_{i,d}(t)) + c_2 * (P_g(t) - X_{i,d}(t)) \tag{18}$$

$$X_{i,d}(t+1) = X_{i,d}(t) + V_{i,d}(t+1) \tag{19}$$

where λ is the inertia weight; c_1 and c_2 are random numbers, P_i is the best particle found so far within the neighbors and P_g is the best position discovered so far by the corresponding particle. Velocity magnitudes are often clipped to a predetermined maximum value, $V_{\max}$.

In general, MPC solves online an open-loop optimal control problem subject to system dynamics and constraints at each time instant and implements only the first element of the control profile. At each sample instant k, plant measurements are obtained and a model of the process is used to predict the future outputs of the process. Using these predictions, the future control sequences $u(k+i\,|\,k)\ i \geq 0$ are computed by minimizing a nominal objective function $J(k)$ over a prediction horizon as follows:

$$\min_{u(k+i|k)\ i\geq 0} \quad J(k) \tag{20}$$

s.t.

$$J(k) = \frac{1}{2}\sum_{i=1}^{N_\mathrm{p}} \| y_\mathrm{r}(k+i) - \hat{y}(k+i) \|_Q^2 + \frac{1}{2}\sum_{i=1}^{N_\mathrm{c}} \| u(k+i-1) - u(k+i-2) \|_R^2$$

where Q>0, R>0 are the weighting matrix of the output and input respectively. N_p is the prediction horizon, N_c is the control horizon, and $\hat{y}$ is the model prediction output, y_r is the output reference trajectory.

To solve the optimization problem (20), the prediction output of the process $y(k+i)\ i>0$ is needed. Here the DRNN model of the process is applied to predict the process output $\hat{y}(k+i)$.

According to (5), The DRNN model of nonlinear process (1) can be rewritten to

$$\hat{y}(k) = NN(u(k-1), y(k-1), W^I, W^h, W^O) \tag{21}$$

Then the i-step-ahead prediction output of nonlinear process (1) can be obtained by using (26) in cascade form

$$\hat{y}(k+i) = NN(u(k+i-1), \hat{y}(k+i-1)) \tag{22}$$

where $\hat{y}(k+i-1) = y(k)\ \ i=1,\ \hat{y}(k+i-1), i>1$

For simplification, let prediction horizon is equal to control horizon, i.e. $N_\mathrm{p} = N_\mathrm{c}$. The overall configuration of the DRNN prediction model output (22) is shown in Fig. 2.

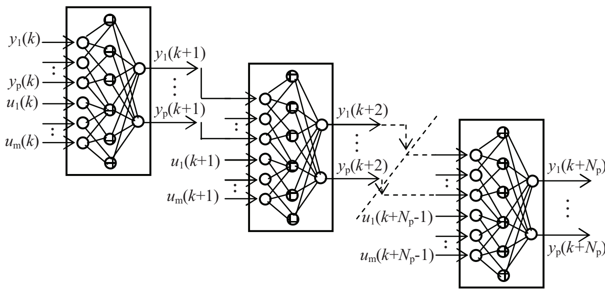


Fig. 2 Structure of prediction model based on DRNN

From (22), it may be seen that $\hat{y}(k+i)$ is a nonlinear function of the future control input sequence $\mathbf{U}$=[$u(k)$ $u(k$+1) $\cdots$ $u(k$+N_p-1)]$^\mathrm{T}$, which makes the minimization problem of with respect to $\mathbf{U}$ still a nonlinear optimization problem. If the conventional descent-based nonlinear optimization method is applied to solve $\mathbf{U}$, it needs much computational efforts. For the purpose of real-time control, avoiding solve nonlinear optimization using conventional recursive iteration method, reducing computational quantity and time, a PSO technique is used to optimize the performance index (20), and directly solve the future control input sequence $\mathbf{U}$. The detailed work principle of the PSO-based predictive control is as follows:

The cost function $J(k)$ is directly regarded as the fitness evaluation function f_{it} of PSO algorithm, the future control input sequence $\mathbf{U}$=[$u(k)$ $u(k$+1) $\cdots$ $u(k$+N_p-1)]$^\mathrm{T}$ is directly regarded as the optimal target of PSO. In the PSO algorithm, the population is a set of solutions and can be represented as $X(t) = \{\mathbf{U}_t^1\ \mathbf{U}_t^2\ \ldots\ \mathbf{U}_t^M\}$ at iteration step t where M is an arbitrary size of population. According to the principle of PSO algorithm, using (18) and (19), $X(t)$ is iteratively updated, and calculates the evaluation function f_{it}. If f_{it} satisfies the end criterion of PSO algorithm, then the particle $\mathbf{U}_t^*$ with the minimized evaluation function $f_{it}{}^*$ in the population $X(t)$ is just the optimal future control input sequence $\mathbf{U}^*$, the first element of $\mathbf{U}^*$, as the control input $u^*(k)$ at the sample instant k is implemented to the plant. At the sample instant k+1, repeat the above procedure.

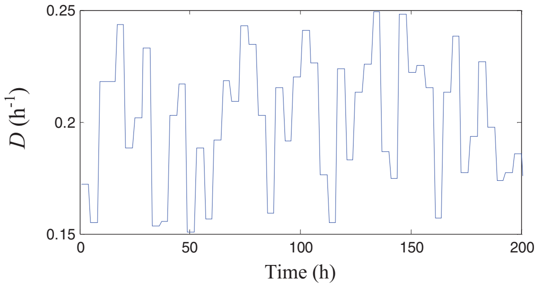
4 Simulation Study

In this study, a continuous fermentor is used to show the effectiveness of the proposed PSO-based predictive control method using adaptive DRNN model. The fermentor is described by the following nonlinear ordinary differential equations [8]:

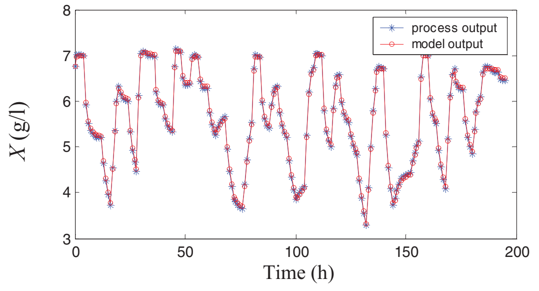
$$\dot{X} = -DX + \mu X\ ,\ \dot{S} = D(S_f - S) - \frac{1}{Y_{x/s}}\mu X$$

$$\dot{P} = -DP + (\alpha\mu + \beta)X\ ,\ \mu = \frac{\mu_m(1 - P/P_m)S}{K_m + S + S^2/K_i}$$

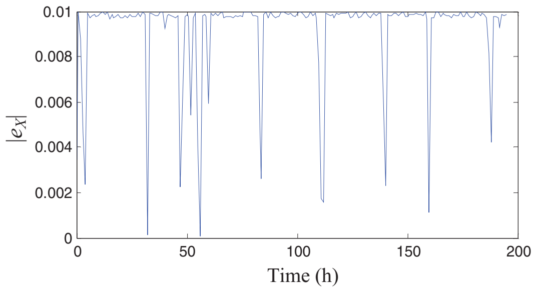
where the dilution rate D and the feed substrate concentration S_f are the process inputs, the effluent cell concentration X, substrate concentration S, and product concentration P are the process state variables, μ is the specific growth rate, $Y_{x/s}$ is the cellmass yield, α and β are kinetic parameters, μ_m is the maximum specific growth rate. The nominal parameters and operating conditions of the fermentor are shown in [8]. The control objective is to regulate the effluent cell concentration X of the fermentor by manipulating the dilution rate D.



(a) Excitation signal



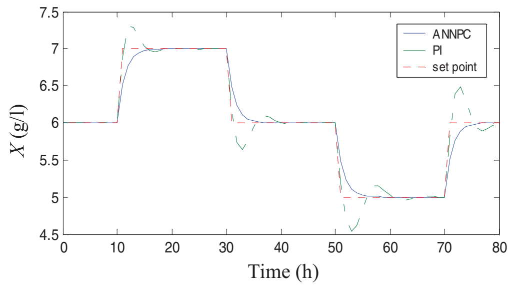
(b) DRNN output and process output



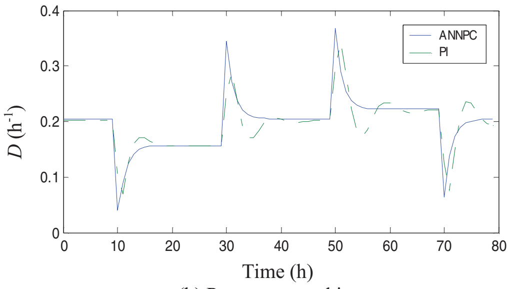
(c) Absolute modeling error

Fig. 3 Validation of DRNN model

To realize the control of the process, we first need build DRNN model of the fermentation process. An input-output data set is generated from the fermentation process when a different amplitude uniform random sequence is used as excitation signal. Then the data set is used to build an adaptive DRNN model. The input of DRNN is chosen as $[D(k\text{-}1)\ X(k\text{-}1)]$, the output of DRNN is $X(k)$, the hidden layer nodes of DRNN are ten. The hidden layer delay of DRNN v=2. In the DRNN recursive training, the initial value of $\theta(0)$ vector is assigned to a small random value in the range of [-0.1, 0.1]. The initial value of the parameters of EKF, $E(0)$ and $R(k|k\text{-}1)$ are assigned to be an identity matrix and a zero matrix respectively. All the initial values of the Jacobian matrix $C(0)$ in the learning calculation are assigned to be zero. The compare results of the adaptive DRNN model output based on EKF algorithm and the process measurement output are shown in Fig. 3. From the figure, it is clearly seen that the absolute modeling error is very small; the adaptive DRNN model can reflect primly the dynamic characteristics of the biochemical reactor.



(a) Process output

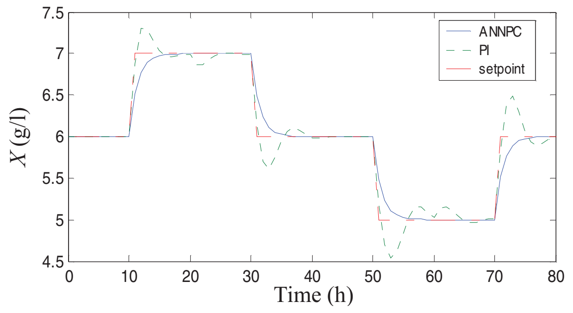


(b) Process control input

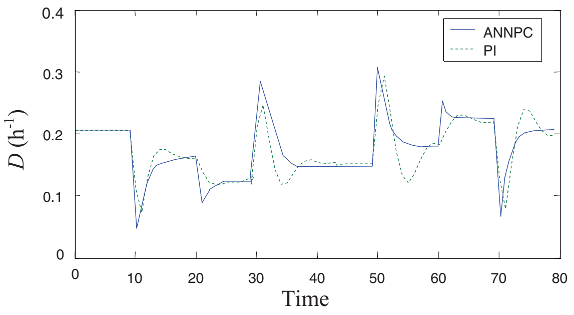
Fig. 4 Cell concentration responses for step changes via ANNPC and PI method with no disturbance

To demonstrate the effectiveness of the proposed method, the results of PSO-based predictive control using adaptive DRNN model are compared to a PI controller. In the simulation, the PI controller parameters K_c=-0.025, T_i=5.5, the prediction horizon $N_p = 5$, the sample time T_s=0.05 h, the parameters of PSO algorithm λ=1.2～0.1, denotes the inertia weight is decreased linearly from 1.2 to 0.1; c_1=0.5, c_2=0.5; the population size M=50, the initial population is chosen as the random number between 0.05 and 0.35. The control objective is to track a series of step changes of 5g/l and 7g/l in the effluent cell concentration X. The compare results are shown in Fig. 4～5. Figure 4 shows the compare result of the process response via PSO-based predictive control using adaptive DRNN model (ANNPC) and PI control with no disturbance. We can see that the tracking performance of ANNPC scheme is better than PI controller. Figure 5 shows the compare results of the process response via ANNPC and PI controller with μ_m disturbances. Due to the fact that the weight parameters of DRNN model are on-line updated using the EKF algorithm, PSO-based ANNPC can effectively overcome the influence of the disturbances, while using PI control the process output greatly fluctuates under the existence of the disturbances.

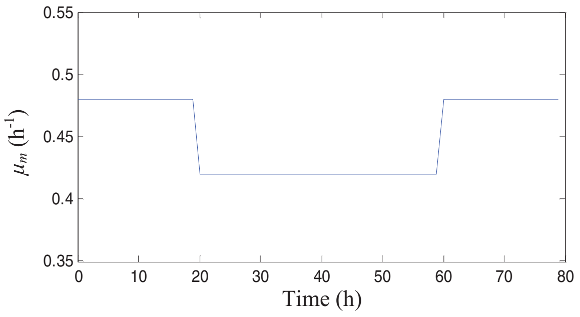
From these compare results, it is clearly seen that the proposed PSO-based ANNPC is a scheme involving the excellent tracking ability and strong robustness. The better control performance can be obtained through PSO-based ANNPC over PI controller.



(a) Process output



(b) Process control input



(c) Step disturbance

Fig. 5 Cell concentration responses for step changes via ANNPC and PI method with μ_m disturbance

5 Conclusion

In this paper, we have presented a PSO-based neural network predictive control method. DRNN is used to model nonlinear process and predict the process output. The extended Kalman filter (EKF) recursive method is used to on-line update and train the linking weights of DRNN. PSO algorithm is used to solve the optimal future control inputs of the process. The case study of a biochemical reactor further demonstrates that the proposed method is an effective control scheme with excellent tracking performance and strong robustness.

REFERENCES

[1] E.F. Camacho, C. Bordons. "Model predictive control in the process industry", London: Springer, 1995

[2] S.J. Qin, T. A. Badgwell. "A survey of industrial model predictive control technology", *Control Engineering Practice*, 2003, 11: 733-764.

[3] J.Q. Huang, F.L. Lewis. "Neural-network predictive control for nonlinear dynamic systems with time-delay", *IEEE Transactions on Neural Networks*, 2003, 14(2): 377-389.

[4] D.W. Yu, D.L Yu. "Neural network control of multivariable processes with a fast optimization algorithm", *Neural Computing & Applications*, 2003, 12: 185-189.

[5] Y. Iiguni, H. Sakai, H. Tokumaru. "A real-time algorithm for multilayered neural network based on the extended Kalman filter", *IEEE Transactions on Signal Processing*, 1992, 40(4): 959-966.

[6] J. Kennedy, R.C. Eberhart, Y. Shi. "Swarm intelligence", San Francisco: Morgan Kaufmann Publishers, 2001.

[7] K.E Parsopoulos, M.N. Vrahatis. "Recent approach to global optimization problems through particle swarm optimization", *Nature Computing*, 2002, 1(2/3): 235-306.

[8] M.A. Henson, D.E. Seborg. "An internal model control strategy for nonlinear systems", *AICHE Journal*, 1991, 37: 1065-1081