Hybrid Modeling Based Adaptive Neural Controller

Anton Andrášik, Alois Mészáros and Sebastião Feyo de Azevedo

Abstract— **In this paper, a new control technique for nonlinear control based on hybrid modeling is proposed. The control system utilizes the well-known gradient descent, but the learning rate is adapted in each iteration step in order to accelerate the speed of convergence. It is shown that the selection of the learning rate results in stable training in the sense of Lyapunov. Advantages of this technique are illustrated by simulations where a continuous flow stirred biochemical reactor is chosen as a case study.**

Keywords— **neural control, PID controller, hybrid modeling**

I. Introduction

The design of most neural control schemes is based on gradient optimisation such as Back-Propagation for the training of weights. Although some nonlinear control problems can be handled by using these neural control schemes, in tasks for nonlinear systems with high nonlinearities and large uncertainties, the existing neural control schemes are severely inadequate. Major drawbacks of using neural networks in controlling real systems are that the ability of the neural network to adapt to system changes is too slow and there remains some degree of error.

Here we will present a new control strategy based on an indirect adaptive control, where the model of the controlled process is represented by hybrid neural network. Just a few parameters of the neural controller are adapted on-line using a special, stable training algorithm in the sense of Lyapunov. To demonstrate the feasibility and the performance of this control scheme, a continuous-flow stirred biochemical reactor model has been chosen as a simulation case study. Simulation results demonstrate the usefulness and the robustness of the control system proposed.

II. Hybrid neural networks

By combining black box techniques with a physical model framework, hybrid models are obtained that combine first principles knowledge with the ability to deal with complex, poorly understood behavior. A partial model is derived from simple physical considerations (such as mass or energy balances), while a black box technique is used to augment the model. Hybrid models are especially suited to describe highly nonlinear behavior over a large operating domain. Examples are models of batch or fed-batch processes, cyclic processes or distributed parameter processes, such as plug flow reactors.

Combining black box techniques with physical equations has received some attention since the early 1990's. There are several approaches to hybrid modeling discussed in literature [1]. In [2], a hybrid model of a fed-batch bioreactor is developed, in which an artificial neural network augments the performance of a parametric model that describes the specific kinetic rates, such as biomass growth and substrate consumption. The combined output of the parametric model and the ANN is processed by an output model which calculates the system state. In comparison with conventional approaches the hybrid modeling is a powerful tool for process modeling, particularly when limited theoretical knowledge of the process is available [3].

However, for the hybrid neural network model, target outputs are not directly available. In this case, the known partial process model can be used to calculate a suitable error signal that can be used to update the network's weights. The observer error between the structured model's predictions and the actual state variable measurements can be "back-propagated" through the known set of equations and translated into an error signal for the neural network component.

Let's consider a process described by differential equation

$$\frac{dy}{dt} = f(y(t), u(t), w) \tag{1}$$

where f is a nonlinear vector function of the system inputs $u(t)$, the system outputs $y(t)$ and some parameters w, which we assume to be represented by means of a neural network. In order to train the neural network (black-box part of the hybrid model), pair of input/output data vectors (u,y) must be available, e.g., as a set of past measurements. The training consists in adaptation of network's weights w in such a way that the sum of the squared deviations between the output data predicted by the network y_i and the corresponding target data y_i^* becomes minimal

$$J = \frac{1}{2}\sum_{i=1}^{N}(y_i - y_i^*)^2 \tag{2}$$

The usual way to minimize J is to use gradient procedures, like the steepest-descent algorithm. Weights in the n-th step of this iterative process are changed in the direction of gradient

$$w_{ij,n+1} = w_{ij,n} - \alpha\frac{\partial J}{\partial w_{ij,n}} \tag{3}$$

In consideration of equation (2) the derivation of J with respect to $w_{ij,n}$ gives

$$\frac{\partial J}{\partial w_{ij,n}} = \sum_{i=1}^{N}\left[(y_i - y_i^*)\frac{\partial y_i}{\partial w_{ij,n}}\right] \tag{4}$$

Now the problem consists in determining the derivatives $\frac{\partial y}{\partial w}$. One possibility is a method called sensitivity approach.

A. Andrášik and A. Mészáros are with the Slovak University of Technology, Faculty of Chemical and Food Technology, Radlinského 9, 812 37 Bratislava, Slovakia. E-mail: andrasik@chtf.stuba.sk, ameszaro@cvt.stuba.sk

S.F. de Azevedo is with the Faculdade de Engenharia de Universidade do Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal. E-mail: sfeyo@fe.up.pt

This approach is based on a concept developed in systems engineering [4]. The main idea of this method is to incorporate the weights as additional variables to the differential equation

$$\frac{dy}{dt} = f\left(y(t), u(t), w\right) \tag{5}$$

During the training phase, this differential equation must be solved for fixed $u(t)$. Differentiation of equation (5) with respect to weights gives

$$\frac{d}{dt}\left(\frac{\partial y}{\partial w}\right) = \frac{\partial f \partial y}{\partial y \partial w} + \frac{\partial f}{\partial w} \tag{6}$$

with initial condition

$$\left.\frac{\partial y}{\partial w}\right|_{t=0} = 0 \tag{7}$$

A. Hybrid model of the bioprocess

The initial phase of an adaptive control is usually the model acquisition of the controlled process. The non-linear model describing the response of *Saccharomyces cerevisiae*, known as baker's yeast, has been used for the process dynamic simulation. This mathematical model, adopted partly from work of [5] and extended to the present dynamical structure [6] is based on limited oxidation capacity of yeast leading to a switchover from oxidative to oxido-reductive metabolism. The model is well behaved for the description of cell growth on glucose as substrate, during the simulation of the control experiments. We found it as an appropriate tool for the data acquisition needed to train the hybrid network. Structure of the hybrid model consisting of neural network and available information of the process is depicted in Fig. 1.

Simplified mass balance of the fermentation process consists of the following differential equations

$$\frac{dX}{dt} = \mu X - \frac{q}{V_l} X \tag{8}$$

$$\frac{dS}{dt} = -k_1 \mu X - \frac{q}{V_l}(S - S_{in}) \tag{9}$$

where X is the biomass concentration, S is the substrate concentration, q is the substrate flow rate, V_l is volume of the liquid phase and S_{in} is the initial substrate concentration. With respect to equation (5), we can formally write the model as

$$f = \mu X - \frac{q}{V_l} X \tag{10}$$

The biomass concentration X is the modelled variable. Gradients $\frac{\partial f}{\partial y}$ and $\frac{\partial f}{\partial w}$ have the form

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial X} = \mu + X\frac{\partial \mu}{\partial X} - \frac{q}{V_l} \tag{11}$$

$$\frac{\partial f}{\partial w} = X\frac{\partial \mu}{\partial w} \tag{12}$$

These two terms are inserted into the sensitivity equation (6), leading to

$$\frac{d}{dt}\left(\frac{\partial X}{\partial w}\right) = \left(\mu + X\frac{\partial \mu}{\partial X} - \frac{q}{V_l}\right)\frac{\partial X}{\partial w} + X\frac{\partial \mu}{\partial w} \tag{13}$$

Now we have one differential equation which computes the gradient $\frac{\partial X}{\partial w}$ necessary to optimize weights during iteration steps: equations (3) and (4). Unknown partial derivations, $\frac{\partial \mu}{\partial X}$ and $\frac{\partial \mu}{\partial w}$ in sensitivity equation, depend on the chosen neural net structure and can be evaluated by using the backpropagation method.

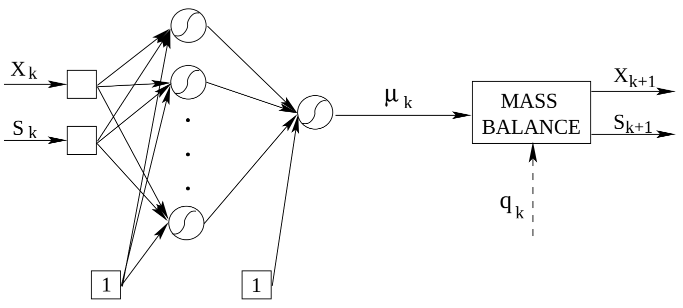

Fig. 1. Structure of the hybrid model of the fermentation process

The mathematical model of the biochemical process was used to generate data sets for training hybrid network and neural controller, respectively. The substrate flow rate q was used as a manipulated variable for generating data. The hybrid network with 7 hidden neurons and structure as in Fig. 1 was fed with training data set and it was trained using sensitivity approach. The comparison of testing data and network predictions is depicted in Fig. 2

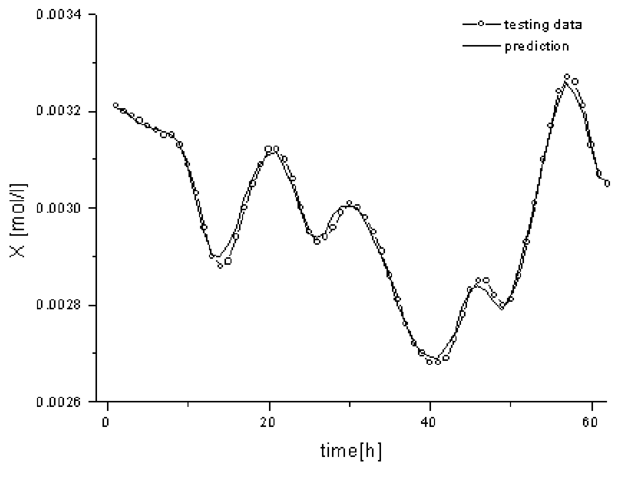

Fig. 2. Modeling of the bioprocess using the hybrid network

III. Control system design

The proposed control system uses two feedforward neural networks, one in a role of the process model and the second network as an inverse dynamic model of the process. Inverse dynamics identification is defined as finding the inverse mapping of a system [7]. It is useful to know the inverse dynamics of a plant in order to control it. Then, in an ideal situation, the dynamics of the controller could simply be made equal to the plant inverse dynamic. For our case we'll consider an approach adopted by [8] called direct or generalised inverse learning. A neural network is fed by outputs from the plant and directly taught to generate the plant input that produced those outputs, as illustrated in Fig. 3. Error between the desired and actual output of the network is used to adjust the network weights.

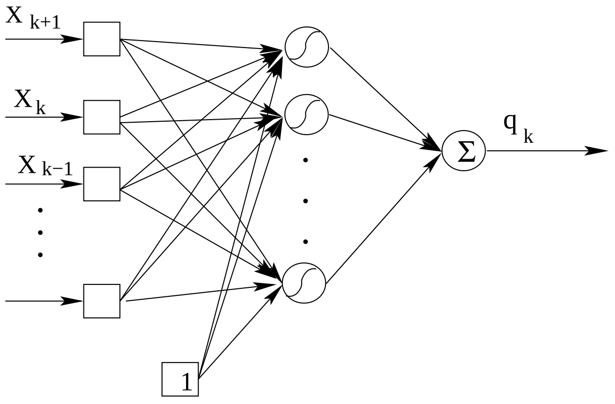

Fig. 3. Architecture of the proposed inverse model of the plant

However, inverse dynamic models are prone to giving incorrect results if the plant has nonlinearities. To prevent the control system to make permanent control error we suggest to incorporate so-called PID neurons into the neural controller as shown in Fig. 4. There are three neurons instead of common bias, where inputs to neurons are control error, sum of control error and deviation of control error. Connections between PID neurons and the output of the neural network are evaluated by weights called w_P, w_I and w_D. These weights are trained on-line using special type of optimizing algorithm described in the next section. The whole control scheme is depicted in Fig. 5.

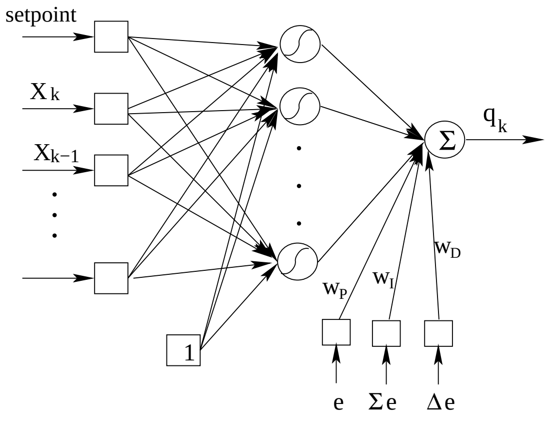

Fig. 4. Neural controller supplemented with PID neurons for on-line learning

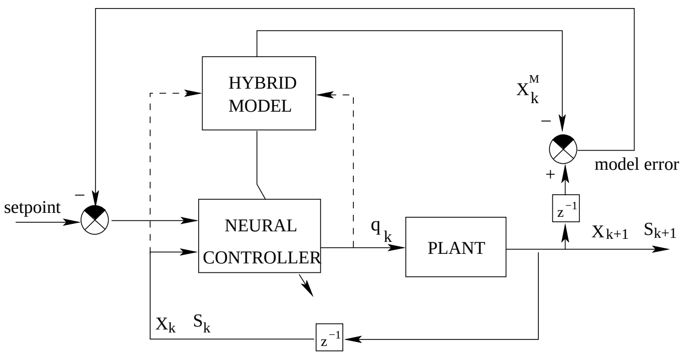

Fig. 5. Block diagram of the proposed control system

A. Convergence of the learning algorithm

The basic idea behind the training consists in on-line tuning of the PID weights of the neural controller in such a way that the control error will converge to zero. The principle of the proof is adopted from [9] where the stability of neural predictor is based on proper selection of the learning rate. Finding of the bounds of the learning rate is accomplished using discrete Lyapunov function.

Let's consider a discrete Lyapunov function given by

$$J_k = \frac{1}{2}(e_k^2 + \lambda \Delta u_k^2) = \frac{1}{2}[(r - y_k^M)^2 + \lambda(u_k - u^{old})^2] \quad (14)$$

where r is the setpoint, y_k^M is predicted value of the controlled variable using the hybrid model, u_k is estimated manipulated variable as the output of the neural controller, u^{old} is the last action applied to the plant and k is an iteration index. The variation of the discrete Lyapunov function is given by

$$\Delta J_k = J_{k+1} - J_k = \frac{1}{2}(e_{k+1}^2 - e_k^2) + \frac{1}{2}\lambda(\Delta u_{k+1}^2 - \Delta u_k^2) \quad (15)$$

where the setpoint error in the next sampling period is estimated as

$$e_{k+1} = e_k + \Delta e_k = e_k + \left(\frac{\partial e_k}{\partial \vec{w}}\right)^T \Delta \vec{w} \quad (16)$$

The same holds for the change of the manipulated variable

$$\Delta u_{k+1} = \Delta u_k + \left(\frac{\partial u_k}{\partial \vec{w}}\right)^T \Delta \vec{w} \quad (17)$$

As we tune just PID weight coefficients and activation function of the output neuron is linear (see Fig. 4), the change of the manipulated variable in the next sampling period is given by

$$\Delta u_{k+1} = \Delta u_k + \vec{B}^T \Delta \vec{w} \quad (18)$$

where $\vec{B}$ is a vector of inputs to PID neurons.

The PID weights are adjusted in proportion to the negative gradient of their contribution to the control error. Unlike the common "steepest descent", learning rate in our case isn't constant. The change of the weight coefficients is given by

$$\Delta\vec{w} = -\alpha\frac{\partial J}{\partial\vec{w}} = -\alpha\left(\frac{\partial J}{\partial e_k}\frac{\partial e_k}{\partial\vec{w}} + \frac{\partial J}{\partial u_k}\frac{\partial u_k}{\partial\vec{w}}\right) = \\ = -\alpha\left(e_k\frac{\partial e_k}{\partial\vec{w}} + \lambda u_k\frac{\partial u_k}{\partial\vec{w}}\right) \quad (19)$$

Let's start with equation (15). Substituting for e_{k+1} from equation (16) and Δu_{k+1} from equation (18) we have

$$\Delta J = \frac{1}{2}\left\{\left[e_k + \left(\frac{\partial e_k}{\partial\vec{w}}\right)^T\Delta\vec{w}\right]^2 - e_k^2 + \right. \\ \left. + \lambda\left[\left(\Delta u_k + \vec{B}^T\Delta\vec{w}\right)^2 - \Delta u_k^2\right]\right\} = \\ = e_k\left(\frac{\partial e_k}{\partial\vec{w}}\right)^T\Delta\vec{w} + \frac{1}{2}\left[\left(\frac{\partial e_k}{\partial\vec{w}}\right)^T\Delta\vec{w}\right]^2 + \\ + \lambda\Delta u_k\vec{B}^T\Delta\vec{w}_k + \frac{1}{2}\lambda\left(\vec{B}^T\Delta\vec{w}_k\right)^2 \quad (20)$$

Next, let us substitute $\Delta\vec{w}_k$ from equation (19) into equation (20), we find

$$\Delta J = -\alpha e_k^2\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 - \alpha\lambda\Delta u_k e_k\left(\frac{\partial e_k}{\partial\vec{w}}\right)^T\vec{B} + \\ + \frac{\alpha^2}{2}\left[e_k\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 + \lambda\Delta u_k\left(\frac{\partial e_k}{\partial\vec{w}}\right)^T\vec{B}\right]^2 - \\ - \alpha\lambda\Delta u_k e_k\vec{B}^T\frac{\partial e_k}{\partial\vec{w}} - \alpha\lambda^2\Delta u_k^2\|\vec{B}\|^2 + \\ + \frac{\alpha^2}{2}\lambda\left[e_k\vec{B}^T\frac{\partial e_k}{\partial\vec{w}} + \lambda\Delta u_k\|\vec{B}\|^2\right]^2 \quad (21)$$

Next modifications results in

$$\Delta J = -\frac{\alpha}{2}e_k^2\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2\left[2 - \alpha\left(\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 + \lambda\|\vec{B}\|^2\right)\right] - \\ - -\frac{\alpha\lambda^2}{2}\Delta u_k^2\|\vec{B}\|^2\left[2 - \alpha\left(\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 + \lambda\|\vec{B}\|^2\right)\right] - \\ - \alpha\lambda\Delta u_k\left(\frac{\partial e_k}{\partial\vec{w}}\right)^T\vec{B}e_k \times \\ \times\left[2 - \alpha\left(\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 + \lambda\|\vec{B}\|^2\right)\right] \quad (22)$$

what leads to the following equation

$$\Delta J = -\frac{\alpha}{2}\left[2 - \alpha\left(\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 + \lambda\|\vec{B}\|^2\right)\right] \times \\ \times\left(e_k\frac{\partial e_k}{\partial\vec{w}} + \lambda\Delta u_k\vec{B}\right)^2 \quad (23)$$

Since the last term is always positive and α is a positive number, we can conclude that the change of the Lyapunov function is negative just in the case when

$$\left[2 - \alpha\left(\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 + \lambda\|\vec{B}\|^2\right)\right] > 0 \quad (24)$$

i.e.

$$\alpha < \frac{2}{\left(\left\|\frac{\partial e_k}{\partial\vec{w}}\right\|^2 + \lambda\|\vec{B}\|^2\right)} \quad (25)$$

B. On-line tuning of PID weights

During the control phase it is necessary to synchronize the following steps:

- data acquisition
- tuning of PID weights
- computation of new control action

The only unknown procedure is how to tune on-line the PID weight coefficients. The key task is to determine the learning rate α in order to speed up the optimization process. First it is necessary to find the gradient $\frac{\partial e_k}{\partial\vec{w}}$. If we suppose that the hybrid model is an exact model of the plant then we can find this gradient using the stochastic approximation approach [10].

For the algorithm is accomplished iteratively, it is convenient to describe it by the following steps:

1. perturbation of weight coefficient in positive direction
2. computation of neural controller response: q_t
3. q_t is the input in mass balance of hybrid network
4. computation of hybrid network response X_{t+1}, what is prediction of the controlled variable at the next sampling period
5. repeating 2-4 for negative perturbation of weight
6. repeating 1-5 for next weight coefficients (w_I, w_D)
7. computation the unknown gradient

$$\frac{\partial e_k}{\partial w} = \frac{X_{t+1}^- - X_{t+1}^+}{\Delta}$$

where Δ is the size of perturbation and +/- is an indication of positive and negative direction

8. computation of the learning rate at the actual iteration

$$\alpha_k = \frac{2c}{\left(\left\|\frac{\partial e_k}{\partial w}\right\|^2 + \lambda\|\vec{B}\|^2\right)}$$

where $c \in (0, 1)$

9. optimization of the PID weight coefficients

$$\vec{w}_k = \vec{w}_{k-1} - \alpha\frac{\partial J}{\partial\vec{w}}$$

10. computation of the objective function at actual iteration

$$J_k = \frac{1}{2}[(r - y_k^M)^2 + \lambda(u_k - u^{old})^2]$$

11. if J_k <(acceptable error) or number of iterations $k > k_{max} \Longrightarrow$ jump to the next step; else $k := k + 1$ and jump to step 1;
12. computation of new control action as a response of the neural PID controller
13. application of new control action to the plant

IV. Results

Two different approaches were carried out, for comparison. First, the inverse model of the process was used as a nonadaptive controller, corresponding control structure is shown in Fig. 6. The training data set generated by the mathematical model [6] was used for inverse mapping of the process. Inputs to the neural controller (Fig. 3) were X_{k+1}, X_k and X_{k-1}, respectively. Two simulation experiments were carried out. First, the controlled process was simulated using the same biomodel parameters as used for training the network. In the second case, the input substrate concentration S_{in} was perturbed at each sampling period in the range up to 20%. Simulation results for deterministic and stochastic system are depicted in Fig. 7 and Fig. 8, respectively. From the experiment it is clear that this type of controller is unable to control the process in a case of imperfect inverse mapping.

The proposed adaptive neural controller was utilized as the second method for the biomass control. The weights of the neural controller were pretrained off-line using the structure in Fig. 3 with 3 inputs and 5 hidden neurons. To training the network, a modified back-propagation algorithm with conjugate gradients was applied. During the on-line control the modified structure shown in Fig. 4 was used, where the first input to the network is setpoint instead of unknown X_{k+1}. Just bias PID weight coefficients have been adapted, rest weight coefficients have remained constant and set on the values pretrained off-line. New control action was optimized every 30 minutes on the basis of the proposed algorithm. The results of control for both the deterministic and stochastic systems are depicted in Fig. 9 and Fig. 10, where setpoints were chosen the same as in the first case, for comparison.

From the evolution of the learning rate α shown in Fig. 11 we can see that the optimization speed of the algorithm was adapted till the end of the control process. Evolutions of the tuned values of P, I and D weight coefficients are depicted in Fig. 12.

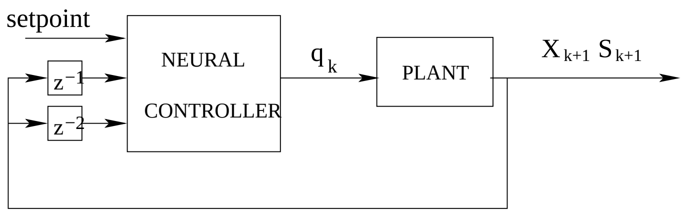

Fig. 6. Block diagram of the direct inverse control

V. Conclusions

A new method for adaptive nonlinear control of biochemical processes has been presented in this report. The new control law incorporates the ability for adaptation through an adjustment of bias neurons and ensures offset-free performance in the presence of unmeasured disturbances. For the special type of control structure has been developed the algorithm for on-line tuning of PID weight coefficients,

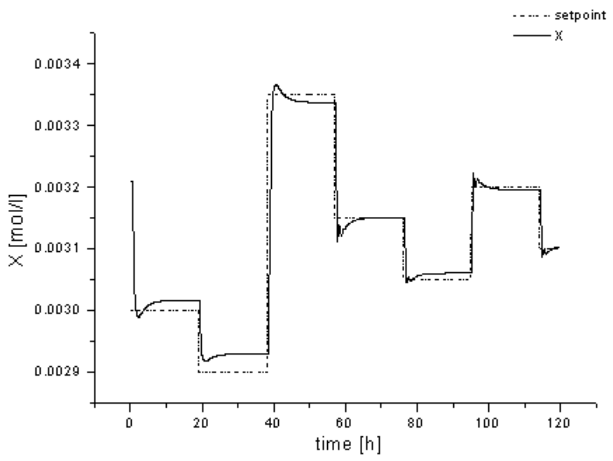

Fig. 7. Control behaviour of direct feedback controller for deterministic system

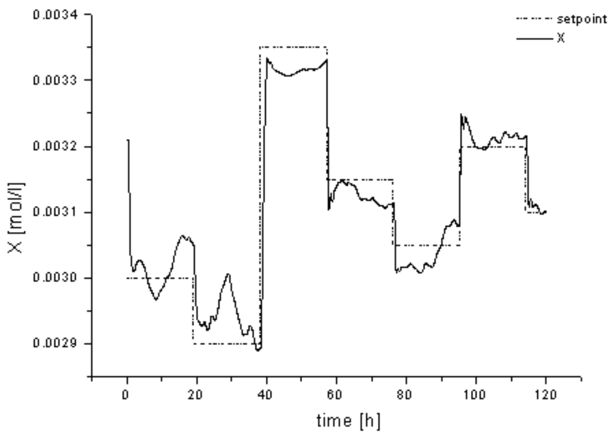

Fig. 8. Direct control of the bioprocess where S_{in} is perturbed in the range of 20%

where the convergence of the algorithm is guaranteed by proper selection of the learning rate.

The proposed neural control system performance has been demonstrated using the nonlinear continuous biochemical process as the case study. The obtained simulation results have demonstrated good regulatory and tracking performance of the augmented adaptive controller introduced. The advantage is obvious especially in the presence of disturbances or parameter uncertainties when it is not possible to obtain a perfect inverse map of the process.

Acknowledgments

The authors highly acknowledge the financial support of the Scientific Grant Agency of the Slovak Republic under grants No. 1/8108/01 and 1/135/03.

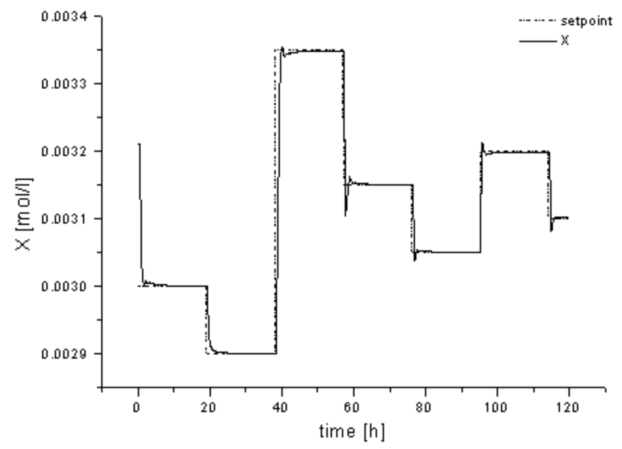

Fig. 9. Control results obtained using on-line tuning of the neural controller - deterministic system

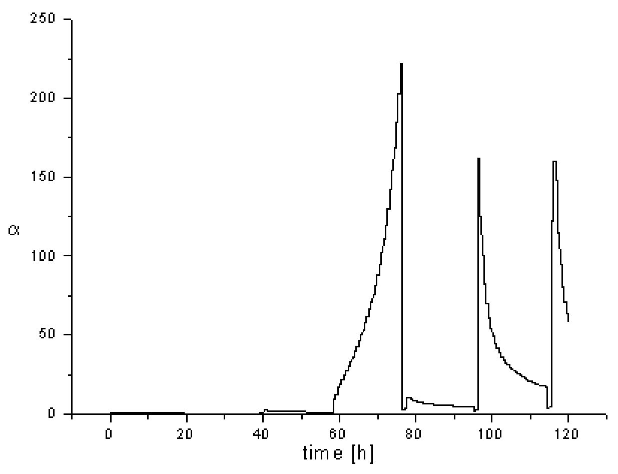

Fig. 11. Evolution of α coefficient

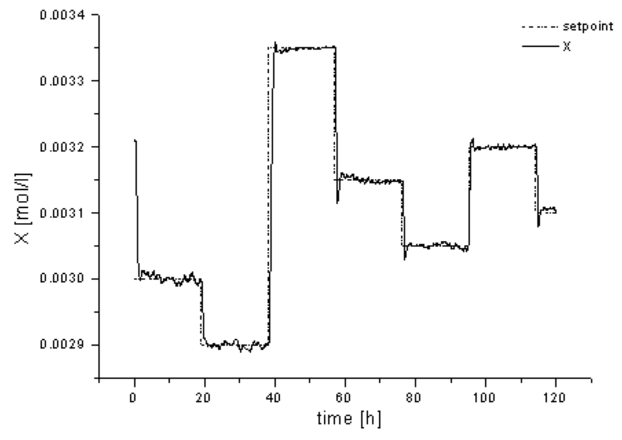

Fig. 10. Adaptive neural PID control of the bioprocess where S_{in} is perturbed in the range of 20%

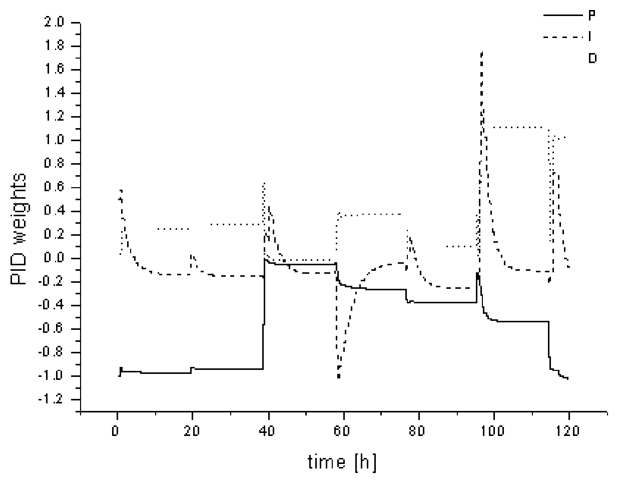

Fig. 12. Evolution of P,I and D weight coefficient during the adaptive control

References

[1] J. Schubert, R. Simutis, M. Dors, I. Havlik, and A. Lübbert, "Bioprocess optimization and control: Application of hybrid modelling," *Journal of Biotechnology*, vol. 35, pp. 51–68, 1994.

[2] M.L. Thompson and A. Kramer, "Modeling chemical processes using prior knowledge and neural networks," *American Institute of Chemical Engineering Journal*, vol. 8, no. 40, pp. 1328–1340, 1994.

[3] S. Feyo de Azevedo, B. Dahm, and F.R. Oliveira, "Hybrid modelling of biochemical processes: A comparison with the conventional approach," *Computers and Chemical Engineering*, vol. 21, pp. S751–S756, 1997.

[4] P.M. Frank, *Introduction to System Sensitivity Theory*, Academic Press. New York, 1978.

[5] A.P.J. Sweere, *Response of baker's yeast to transient environmental conditions relevant to largescale fermentation processes*, Ph.D. thesis, Drukkerij Elinkurijk Utrecht, 1988.

[6] A. Mészáros, M.A. Brdys', P. Tatjewski, and P. Lednický, "Multilayer adaptive control of continuous bioprocesses using optimising control technique. Case study: Baker's yeast culture," *Bioprocess Engineering*, vol. 12, pp. 1–9, 1995.

[7] D.T. Pham and S.J. Oh, "Identification of plant inverse dynamics using neural networks," *Artificial Intelligence in Engineering*, vol. 13, pp. 309–320, 1999.

[8] D. Psaltis, A. Sideris, and A. Yamamura, "A multilayered neural network controller," *IEEE Control Systems Magazine*, pp. 17–21, April 1989.

[9] K.A. Hoo, E.D. Sinzinger, and M.J. Piovoso, "Improvements in the predictive capability of neural networks," *Journal of Process Control*, vol. 12, pp. 193–202, 2002.

[10] H.J. Kushner and E. Sanvicente, "Stochastic approximation methods for constrained systems with observation noise on the systems and constraints," in *IFAC Symposium on Stochastic Control*, 1974.