НАУЧНИ ТРУДОВЕ
ТОМ LX
"ХРАНИТЕЛНА НАУКА, ТЕХНИКА И ТЕХНОЛОГИИ – 2013"
18-19 октомври 2013, Пловдив

SCIENTIFIC WORKS
VOLUME LX
„FOOD SCIENCE, ENGINEERING AND TECHNOLOGIES – 2013"
18-19 October 2013, Plovdiv

ПРЕДСКАЗВАЩ РЕГУЛАТОР С НЕВРОННО-РАЗМИТ МОДЕЛ С МНОЖЕСТВО РЕКУРЕНТНИ ВРЪЗКИ

Маргарита Терзийска[1], Янчо Тодоров[2], Михаил Петров[1]
[1]*Технически университет София, филиал – Пловдив*
[2] *Институт по информационни и комуникационни технологии, БАН, София*

PREDICTIVE CONTROLLER BASED ON NEURO-FUZZY MODEL WITH MULTIPLE RECURRENT NODES

Margarita Terziyska[1], Yancho Todorov[2], Michail Petrov[1]
[1]*Technical University Sofia, branch – Plovdiv*
[2] *Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Sofia*

Abstract
A nonlinear predictive controller based on a recurrent neuro-fuzzy model is presented in this paper. Neuro-fuzzy model is realized with the T-S inference mechanism and includes global and local (after the rules layer) feedbacks. The proposed model is coupled with an optimization approach for computation of the control actions into a model based predictive controller. The efficiency of the proposed control policy is proved by simulation experiments to control a Continuous Stirred Tank Reactor (CSTR).

Keywords: *nonlinear predictive controller, recurrent neuro-fuzzy model, CSTR.*

Introduction

The generalized predictive control (GPC) is an model predictive control algorithm that was proposed by Clarke et al [1]. It is the most popular model-based control methods both in industry and academia. GPC has been successfully implemented in many industrial applications [2].

GPC based on neural network model is also presented in the literature [3-6]. Neural networks are known as universal approximators. They are trainable dynamic systems which can learn a nonlinear input-output mapping from training data sets, and they can update their weights through adaptively repeating training cycles. Also, neural networks can solve problems with time-varying parameters, they can solve large-scale problems due to the inherent nature of parallel and distributed information processing and they can be implemented physically in designed hardware.

The last decade has significantly increased the use of recurrent neural networks. These networks include not only feedforward, but also feedback (recurrent) loops (local and / or global). These feedbacks play the role of "memory" and allow the model better to capture the dynamics of the process. That is why it could be said that recurrent neural networks are better approximators than feedforward neural networks. Many researchers are proposed GPC based on recurrent neural networks [7, 8].

From the other side, the fusion of the fuzzy logic with the neural networks allow to combine the learning and computational ability of neural networks with the human like IF-THEN thinking and reasoning of fuzzy system. This could be compared with the human brain [9] – neural network concentrate on the structure of human brain, i.e., on the "hardware" whereas logic system concentrate on "software". Combining neural networks and fuzzy systems in one unified framework has become popular in the last few years. Many methods have been proposed in the literature that combine fuzzy neural network and model predictive control algorithm. Predictive controllers based on recurrent neuro-fuzzy models are described in [10-13].

In this paper, an implicit GPC algorithm based on recurrent neuro-fuzzy network (RNFN) with global and local (after the rules layer) feedbacks and T-S fuzzy inference, is presented. The proposed model is coupled with a Gradient optimization approach into a model based control scheme. To prove the efficiency of the proposed control strategy, a simulation

experiments to control a nonlinear Continuous Stirred Tank Reactor (CSTR), are performed.

Recurrent Neuro-Fuzzy Model

Consider a class of nonlinear systems described by the following NARX (*Nonlinear AutoregRessive model with eXogenous inputs*) representation model:

$$y(k) = f_y(x(k)) \tag{1}$$

where the unknown nonlinear function f_y can be approximated by Takagi-Sugeno type fuzzy rules:

$$R^{(i)}: if\ x_1\ is\ \tilde{A}_1^{(i)}\ and\ x_p\ is\ \tilde{A}_p^{(i)}\ then\ f_y^{(i)}(k) \tag{2}$$

$$f_y^{(i)}(k) = a_1^{(i)} y(k-1) + a_2^{(i)} y(k-2) + \cdots + a_{ny}^{(i)} y(k-n_y) + b_1^{(i)} u(k) + b_2^{(i)} u(k-1) + \cdots + b_{nu}^{(i)} u(k-n_u) + c_0^{(i)} \tag{3}$$

where (i)=1,2,…N, N is the number of the fuzzy rules, A_i is an activated fuzzy set defined in the universe of discourse of the input xi and the crisp coefficients a_1, a_2,...a_{ny}, b_1, b_2,...b_{nu} are the coefficients into the Sugeno function $f_y(k)$.

This section describes developing on a multilayer recurrent neuro-fuzzy network for nonlinear system (1), whose structure is shown in Fig.1.

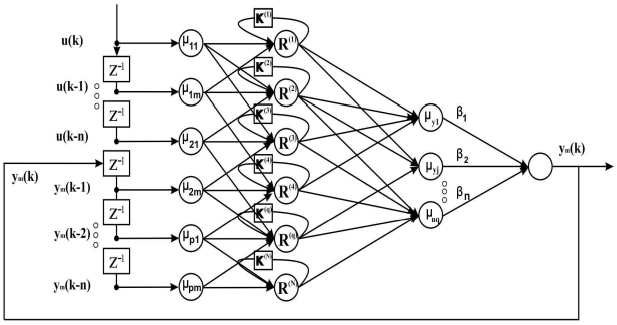

Fig. 1. Schematic diagram of the proposed recurrent neuro-fuzzy network

Layer 1: This layer accepts the input variables and then nodes in this layer only transmit the input values to the next layer directly.

Layer 2: Each node in this layer does the fuzzyfication via Gaussian membership function:

$$\mu_{Xp,m}^{(n)} = \exp\frac{-(x_p - c_{Xp,m})^2}{2\sigma_{Xp,m}^2} \tag{4}$$

where X_p are the input values, $c_{Xp,m}$ and $\sigma_{Xp,m}$ are the center and the standard deviation of the Gaussian membership function.

Layer 3: This layer is a kind of rules generator as it formed the fuzzy logic rules. Their number depends on the number of inputs *p* and the number of their fuzzy sets *m*, and is calculated according to the expression $N=m^p$. In this layer, each node represents a rule and act as a unit memory. The rules have the same form as in equation (2), but the f_y is modified as follow:

$$f_y^{(i)}(k) = a_1^{(i)} y(k-1) + a_2^{(i)} y(k-2) + \cdots + a_{ny}^{(i)} y(k-n_y) + b_1^{(i)} u(k) + b_2^{(i)} u(k-1) + \cdots + b_{nu}^{(i)} u(k-n_u) + c_0^{(i)} + \ldots + \theta^i f_y^{(i)}(k-1) \tag{5}$$

where θ^i denoted the feedback gain of the *i*th rules.

Layer 4: In the fourth layer is realized implication operation:

$$\mu_{yq}^{(n)}(k+j) = \mu_{x_1,m}^{(n)}(k+j) * \mu_{x_2,m}^{(n)}(k+j) * \ldots * \mu_{x_p,m}^{(n)}(k+j) \tag{6}$$

Layer 5: In the fifth last layer takes the removal decision which consists in determining the value of the model output by the expression:

$$\hat{y}(k+j) = \frac{\sum_{i=1}^{q} f_y^{(i)}(k+j)\mu_y^{(i)}(k+j)}{\sum_{i=1}^{q} \mu_y^{(i)}(k+j)} \tag{7}$$

The learning algorithm for the recurrent neuro-fuzzy model is very simple. It is based on minimization of an instant error measurement function between the real plant output and the process output, calculated by the fuzzy-neural model:

$$E(k) = \frac{(y(k) - \hat{y}(k))^2}{2} \tag{8}$$

where y(k) denotes the measured real plant output and ŷ(k) is the recurrent fuzzy neural network model output.

During the gradient learning procedure it is needed to adjusted three groups of parameters in the recurrent neuro-fuzzy network and they are: premise and consequent parameters on the fuzzy rules and the feedback gain θ^i. The consequent parameters are the coefficients a_1, a_2...a_{ny}, b_1, b_2...b_{nu} in the Sugeno function f_y and they are calculated at first step by the following equations:

$$\beta_{ij}(k+1) = \beta_{ij}(k) + \eta(y(k) - y_M(k))\bar{\mu}_y^{(j)}(k)x_i(k) \tag{9}$$

$$\beta_{0j}(k+1) = \beta_{0j}(k) + \eta(y(k) - y_M(k))\bar{\mu}_y^{(j)}(k) \tag{10}$$

where η is the learning rate and β_{ij} is an adjustable i^{-th} coefficient (a_i or b_i) in the Sugeno function of the j^{-th} activated rule.

The premise parameters are the centre c_{ij} and the deviation σ_{ij} of a Gaussian fuzzy set. They can be

calculated at second step using the following equations:

$$c_{ij}(k+1) = c_{ij}(k) + \eta(y(k) - y_M(k))\bar{\mu}_y^{(j)}(k)[f_y^{(i)}(k) - \hat{y}(k)]\frac{[x_i(k) - c_{ij}(k)]}{c_{ij}^2(k)} \quad (11)$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) + \eta(y(k) - y_M(k))\bar{\mu}_y^{(j)}(k)[f_y^{(i)}(k) - \hat{y}(k)]\frac{[x_i(k) - \sigma_{ij}(k)]^2}{\sigma_{ij}^3(k)} \quad (12)$$

The feedback gains θ^i, where i denote the activated rule, are updated by following expression, as well:

$$\theta^i(k+1) = \theta^i(k) + \eta_\theta(y(\kappa) - y_M(\kappa))\bar{\mu}_y^{(j)}(k) f_y^{(i)}(k-1) \quad (13)$$

Implicit Generalized Predictive Control

Using the designed RNFN model, the *Optimization Algorithm* computes the future control actions at each sampling period, by minimizing the proposed by Clarke et al [1] cost function:

$$J(k, u(k)) = \sum_{i=N_1}^{N_2}(r(k+i) - \hat{y}(k+i))^2 + \rho\sum_{i=1}^{N_u}\Delta u(k+i-1)^2 \quad (14)$$

where $\hat{y}$ is the predicted model output, r is the system reference, u is the control action, $N1$ is the minimum prediction horizon, $N2$ is the maximum prediction horizon, Nu is the control horizon and ρ is the weighting factor penalizing changes in the control actions.

When the criterion function is a quadratic one and there are no constraints on the control action, the cost function can be minimized analytically. If the criterion J is minimized with respect to the future control actions u, then their optimal values can be calculated by applying the condition:

$$\nabla J[k, U(k)] = \left[\frac{\partial J[k,U(k)]}{\partial u(k)}, \frac{\partial J[k,U(k)]}{\partial u(k+1)}, ..., \frac{\partial J[k,U(k)]}{\partial u(k+N_u-1)}\right] = 0 \quad (15)$$

Each element from this gradient vector can be calculated using the following equation:

$$\frac{\partial J[k,U(k)]}{\partial U(k)} = \begin{bmatrix} -2\left[R(k) - \hat{Y}(k)\right]^T \frac{\partial \hat{Y}(k)}{\partial U(k)} + \\ +2\rho\hat{U}(k)^T \frac{\partial \hat{U}(k)}{\partial U(k)} \end{bmatrix} \quad (16)$$

From the above expression can be seen that it is needed to obtain two groups of partial derivatives. The first one is $\left[\frac{\partial \hat{Y}(k)}{\partial U(k)}\right]$, and second one is $\left[\frac{\partial \hat{U}(k)}{\partial U(k)}\right]$. Each element from the first group of partial derivatives is calculated with the following equations:

$$\frac{\partial \hat{y}(k)}{\partial u(k)} = \sum_{i=1}^{N} b_1^{(i)}\bar{\mu}_y^{(i)}(k) \quad (17)$$

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N}\left[a_1^{(i)}\frac{\partial \hat{y}(k)}{\partial u(k)} + b_2^{(i)}\right]\bar{\mu}_y^{(i)}(k+1) + \frac{\partial \hat{y}(k)}{\partial u(k)} \quad (18)$$

$$\frac{\partial \hat{y}(k+2)}{\partial u(k)} = \sum_{i=1}^{N}\left[\begin{matrix} a_1^{(i)}\frac{\partial \hat{y}(k+1)}{\partial u(k)} + \\ +a_2^{(i)}\frac{\partial \hat{y}(k)}{\partial u(k)} \end{matrix}\right]\bar{\mu}_y^{(i)}(k+2) + \frac{\partial \hat{y}(k+1)}{\partial u(k)} \quad (19)$$

$$\frac{\partial \hat{y}(k+N_2)}{\partial u(k)} = \sum_{i=1}^{N}\left[\begin{matrix} a_1^{(i)}\frac{\partial \hat{y}(k+N_2-1)}{\partial u(k)} + \dots \\ +a_{ny}^{(i)}\frac{\partial \hat{y}(k+N_2-2)}{\partial u(k)} \end{matrix}\right]\bar{\mu}_y^{(i)}(k+N_2) + \frac{\partial \hat{y}(k+N_2-1)}{\partial u(k)} \quad (20)$$

The second group partial derivatives have the following form:

$$\frac{\partial \hat{U}(k)}{\partial U(k)} = \begin{bmatrix} \frac{\partial \Delta u(k)}{\partial u(k)} & .... & \frac{\partial \Delta u(k)}{\partial \Delta u(k+N_u-1)} \\ ..... & ..... & ..... \\ \frac{\partial \Delta u(k+N_u-1)}{\partial u(k)} & .... & \frac{\partial \Delta u(k+N_u-1)}{\partial \Delta u(k+N_u-1)} \end{bmatrix} \quad (21)$$

Since $\Delta u(k) = u(k) - u(k-1)$, this is:

$$\frac{\partial \hat{U}(k)}{\partial U(k)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad (22)$$

Afterwards, using the above equations, the calculation of the control actions are being calculated as:

$$\Delta u(k+N_u-1) = \rho^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-1)} + ... + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-1)}\right] \quad (23)$$

$$\Delta u(k+N_u-2) = \Delta u(k+N_u-1) + \rho^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k+N_u-2)} + ... + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k+N_u-2)}\right] \quad (24)$$

$$\Delta u(k) = \Delta u(k+1) + ... + \rho^{-1}\left[e(k+N_1)\frac{\partial \hat{y}(k+N_1)}{\partial u(k)} + ... + e(k+N_2)\frac{\partial \hat{y}(k+N_2)}{\partial u(k)}\right] \quad (25)$$

Finally, according to the used receding horizon strategy, only the first control action is being sent to the plant.

Results

Recurrent Neuro-Fuzzy Network validation

To investigate the modeling potentials of the proposed neuro-fuzzy network with multiple recurrent nodes, a Mackey-Glass chaotic system series benchmark model, has been used. The used time series will not converge or diverge, and the trajectory is highly sensitive to initial conditions. The MG time series is described by time-delay differential equation:

$$x(i+1) = \frac{x(i) + ax(i-s)}{(1+x^c(i-s)) - bx(i)} \qquad (26)$$

where a=0.2; b=0.1; C=10; and x(0)=0.1 and s= 17s. The model learning rate for the model η has a fixed value of 0.055.

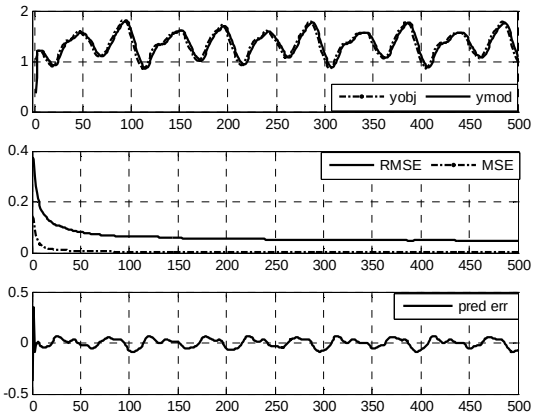

Fig. 2. Validation of the proposed recurrent neuro-fuzzy network by using Mackey-Glass chaotic time series

On Fig. 2 are shown results on model validation by using Mackey-Glass chaotic time series. As it can be seen, the proposed model structure predicts accurately the generated time series, with minimum prediction error and fast transient response of the RMSE, reaching value closer to zero. The value of the MSE in the 50-th time step is 0.045.

Implicit GPC based on Recurrent Neuro-Fuzzy Network

Additionally, to investigate the performance of the designed recurrent neuro-fuzzy network, it was incorporated into GPC scheme. To evaluate the obtained model based control strategy, a nonlinear model of a continuous stirred tank reactor (CSTR), was used in a simulation batch. The dynamic equations of the nonlinear CSTR are given by (14) as follow:

$$\dot{x}_1 = -x_1 + D_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\phi}\right) \qquad (27)$$

$$\dot{x}_2 = -(1+\delta)x_2 + BD_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\phi}\right) + \delta u \qquad (28)$$

where x_1 and x_2 represent the dimensionless reactant concentration and the reactor temperature, respectively. The control action u is dimensionless cooling jacket temperature. The physical parameters in the CSTR model equations are Da, φ, B and δ which correspond to the Damkhler number, the activated energy, the heat of reaction and the heat transfer coefficient, respectively. Based on the nominal values of the system parameters, Da=0.072, φ=20, B=8 and δ=0.69, the open-loop CSTR exhibits three steady states (x1,x2)A=(0.144, 0.886), (x1,x2)B=(0.445, 2.75) and (x1,x2)C=(0.765, 4.705), where the upper and the lower steady states are stable, whereas the middle one is unstable. The control objective here is to bring the nonlinear CSTR from the stable equilibrium point (x1,x2)A to the unstable one (x1,x2)B. All of the results presented are based on the reactor temperature x2, that is y(t)=x2(t).

The number of the Gaussian fuzzy membership functions for each input variable is chosen to be equal to three. Their initial parameters are normalized and uniformly distributed into the universe of discourse є [0, 1]. The RNFN based GPC controller has the following parameters: N1=1; Np=5; Nu=3; ρ=0.08. The results are shown in Figs. 3, 4, 5, 6. All of the figures show the behaviour of the system output and the control action. On Fig.3 it is shown transient process responses in case of variable system reference. Simulation results, in case of change of plant parameters are presented on Fig. 4 - after 400 steps the values of the heat coefficient δ and the heat of reaction B are suddenly changed to 0.8 and 7.5, respectively. It was also examined the disturbance rejection ability. Results in this case are shown on Fig. 5. It was used a step feed disturbance of amplitude 0.2, which added in the right hand side of the equation (28) after step 400. Fig. 6 shows comparison between RNFN based GPC and feedforward neuro-fuzzy network (FNFN) based GPC.

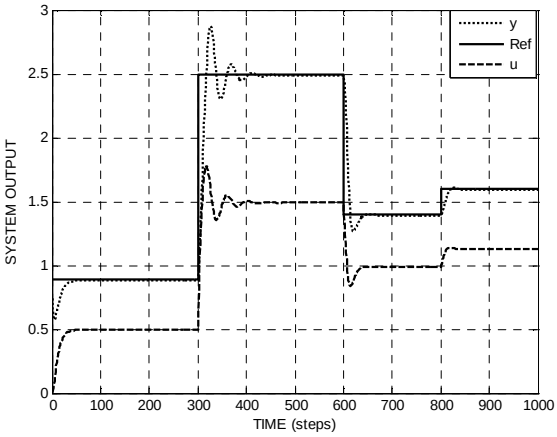

Fig. 3. Transient process responses in case of variable system reference

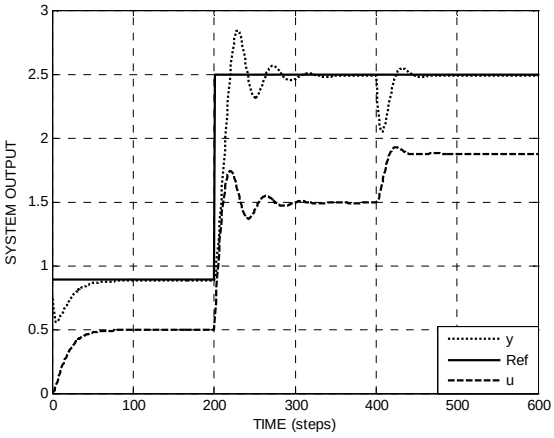

Fig. 4. System behavior after the plant parameter changes – B=7.5, δ=0.8 at the step 400

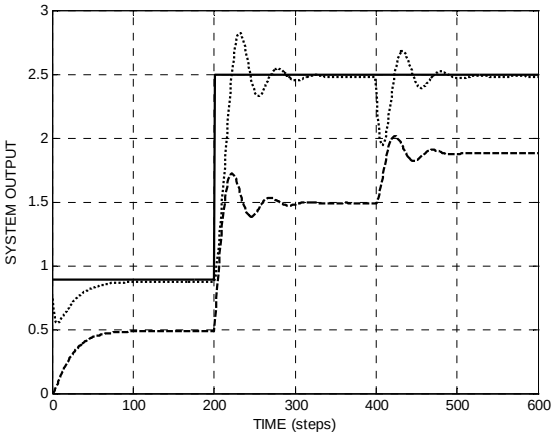

Fig. 5. System behavior after adding an additive output disturbance with an amplitude 0.2 at the step 400

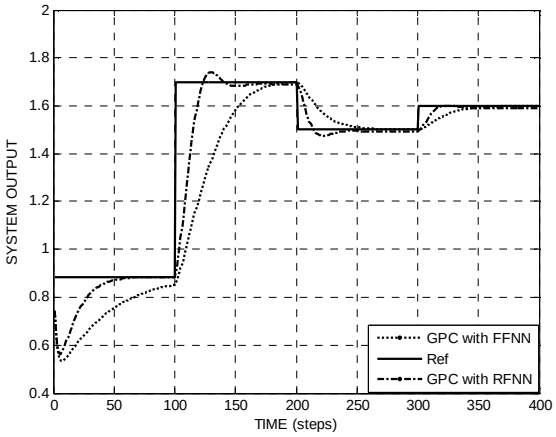

Fig. 6. Comparison between RNFN based GPC and FNFN based GPC

Conclusions

GPC based on RNFN with multiple recurrent nodes it is presented in this paper. Neuro-fuzzy model is with T-S inference mechanism and includes global and local (after the rules layer) feedbacks. To demonstrate the effectiveness of the proposed RNFN model is made two groups of tests. First, the model is used to predict Mackey-Glass chaotic system series. The simulation results shows that the RNFN predicts accurately the generated time series, with minimum prediction error and fast transient response of the RMSE, reaching value closer to zero. The value of the MSE in the 50-th time step is 0.045 (see Fig.2).

Secondly, the designed RNFN was incorporated into GPC scheme and the obtained model based control strategy is tested to control CSTR. The results show that the predictive controller based on proposed RNFN model works precisely in a cases of a variable system reference, plant parameter changes and in the presence of disturbances. It is also made comparison between RNFN based GPC and FNFN based GPC (Fig.6). It can be seen that RNFN based GPC provides a better and faster control.

References

[1] D. W. Clarke, C. Mohtai, P. S. Tuffs, "Generalized predictive control – part I. The basic algorithm"," Automatica, vol. 23, no. 2, 1987, pp. 137-148.
[2] E. F. Camacho, C. Bordons, Model Predictive Control, Springer-Verlag, 2nd ed. 2004.
[3] S. Chidrawar, B. Patre, Generalized Predictive Control and Neural Generalized Predictive Control, Leonardo

Journal of Sciences, Issue 13, July-December 2008, p. 133-152.

[4] Lin Niu and Junsheng Li, Adaptive Neural Network Generalized Predictive Control for Unknown Nonlinear System, TELKOMNIKA, Vol. 11, No. 7, July 2013, pp. 3611 ~ 3617.

[5] P. Shrivastava, Modeling and Control of CSTR using Model based Neural Network Predictive Control, International Journal of Computer Science & Information Security, Jul2012, Vol. 10 Issue 7, p. 38.

[6] A.Vasičkaninová, M. Bakošová, Neural Network Predictive Control of a Chemical Reactor, Acta Chimica Slovaca, Vol.2, No.2, 2009, pp. 21 – 36.

[7] K. Patan, J. Korbicz, Nonlinear Model Predictive Control of a Boiler Unit: A Fault Tolerant Control Study, Int. J. Appl. Math. Comput. Sci., 2012, Vol. 22, No. 1, 225–237.

[8] V. Rankovic, J. Radulovic, N. Grujovic, D. Divac, Neural Network Model Predictive Control of Nonlinear Systems Using Genetic Algorithms, INT J COMPUT COMMUN, ISSN 1841-9836, Vol.7 (2012), No. 3 (September), pp. 540-549.

[9] Chennakesava R. Alavala, Fuzzy Logic and Neural Networks: Basic Concepts and Applications, New Age International Pvt Ltd Publishers (December 1, 2008).

[10] Chi-Huang Lu and Ching-Chih Tsai, Generalized predictive control using recurrent fuzzy neural networks for industrial processes, Journal of Process Control 17 (2007) 83–92.

[11] Jérôme Mendes, Nuno Sousa, and Rui Araújo. "Adaptive predictive control with recurrent fuzzy neural network for industrial processes." Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on. IEEE, 2011.Литературен източник. Литературен източник. Литературен източник.

[12] Chi-Huang Lu, Chi-Ming Liu, and Yuan-Hai Charng. "Stable predictive control based on recurrent fuzzy neural networks." Control Conference (ASCC), 2011 8th Asian. IEEE, 2011.

[13] Y. Todorov, M. Terzyiska, M. Petrov, Recurrent Fuzzy-Neural Network with Fast Learning Algorithm for Predictive Control, Artificial Neural Networks and Machine Learning – ICANN 2013, Lecture Notes in Computer Science Volume 8131, 2013, pp 459-466.

[14] W. H. Ray, Advanced Process Control, McGraw-Hill, New York, 1981.