Journal of Mechatronics, Automation and Identification Technology
Vol. 1, No.2. pp. 1 - 6

An Adaptive Neural IMC Design of Nonlinear Dynamic Processes

Milorad BOŽIĆ, Igor KRČMAR, Jasmin IGIĆ

Faculty of Electrical Engineering, University of Banja Luka, Bulevar vojvode Petra Bojovića 1A, 78000 Banja Luka, Bosnia and Herzegovina
Faculty of Electrical Engineering, University of Banja Luka, Bulevar vojvode Petra Bojovića 1A, 78000 Banja Luka, Bosnia and Herzegovina
Mtel a.d., Trg Mladena Stojanovića 8, 78000 Banja Luka
vdragana@uns.ac.rs, mbozic@etfbl.net, ikrcmar@etfbl.net, jasmin.igic@mtel.ba

***Abstract*— In contrast to usually applied Neural Network (NN) based controllers where the structure and fixed parameters of NN were obtained off line, here we present a fully adaptive Internal Model-based Neural Control design aimed to unknown nonlinear industrial processes with stable dynamics. The internal model of the control plant is implemented by the NN provided with a Stochastic Gradient Descent (SGD) learning algorithm. To cope with a high variability of process gain at different operational points and possible high errors in estimation of the corresponding sensitivities of nonlinear process model we proposed one practical solution to eliminate offset in a steady state at constant system inputs. Some illustrations and performance testing of the proposed adaptive NN controller are given by using an example.**

***Keywords*— artificial neural networks, nonlinear internal model control, process control, zero steady-state error.)**

I. INTRODUCTION

It is well known for a rather long time that the Internal Model Control (IMC) structure has many positive characteristics in terms of stability, robustness and accuracy in a steady-state with respect to a conventional one in the cases of linear models [1]. The IMC design algorithms based on linear models have attracted much attention of control theorists, as well as of practitioners, especially within industrial applications in 1980s. Almost in parallel with such developments many successful trials were conducted using the IMC structure combined with some kind of adaptation mechanism applied to control slowly varying processes [2],[3]. As Multilayer Neural Networks (MLNNs) have been provided with universal approximation capabilities, they were used in IMC structures to control time invariant nonlinear processes [4],[5],[6],[7]. To achieve zero steady state error when constant disturbances and the possible mismatch between real plant and its NN model are present, the Approximate Internal Model-based Neural Control (AIMNC) was proposed for unknown nonlinear discrete time processes [6],[8],[9],[10],[11]. It should be noticed that NNs used in the proposed algorithms were trained off line, i.e. they have a fixed architecture and parameters at implementation. In order to achieve high approximation accuracy, the pre-trained NNs had to have complex architectures (for example, MLNNs with two hidden layers and high number of neurons in each of both) [11], [12]. Besides of NNs complexity required for modeling of high order nonlinear dynamical processes, in some control algorithms it is required to estimate the values of gains and incorporate them in a control variable calculation.

Having in mind that the dynamics of the controlled process can be slowly varying implies that NNs used in control algorithms should have at least some variable parameters and a possibility for their tuning in real time. In that sense high advantages have models which are linear in parameters such as Radial Basis Function (RBF) NNs with fixed hidden neuron weights and parameters, Extreme Learning Machines (ELM) [13], [14], etc. An essential part of adaptive control system is the parameter estimation algorithm that must have a rapid convergence. It is well known that such characteristics have algorithms based on a Least Mean Squares (LMS) performance criterion, i.e. some variants of Recursive LMS (RLMS) or SGD algorithms are very suited to adaptive control system designs [15],[16],[17].

The fully adaptive Internal Model-based Neural Control design proposed here is based on the Fast Clustered Radial Basis Function Network (FCRBFN) [16] and an estimation algorithm of the Nonlinear Autoregressive Moving Average with eXogenous input (NARMAX) model of the controlled process inside IMC system structure provided with the SGD learning algorithm. Instead of using a pre-trained NN model and its corresponding derivative as in the AIMNC [9],[11], we use the FCRBFN and a simple on line calculation of the appropriate control gain in order to deal with a possible offset at steady state.

The rest of the paper is organized as follows. In the Section II, the plant modeling, NN plant model and learning algorithm SGD with a momentum term are given. The Section III gives a description of the fully adaptive neural controller together with the simple way of its gain

adjustment. Simulation results demonstrating performance of the proposed control law design procedure are given in the Section IV. In the Section V we give conclusions of the work.

II. The On-Line Plant Identification

A. The plant modeling

To formulate a control problem for which we propose neural adaptive controllers, it is assumed that the plant dynamics can be modeled by following Nonlinear Autoregressive Moving

Average with eXogenous input (NARMAX) model

$$y(k+1) = f(y,u) + d(k), \tag{1}$$

where f stands for a nonlinear function, y = [y(k), y(k - 1),…, y(k - n + 1)]^T and u =[u(k),u(k - 1),…,u(k - m + 1)]T represent vectors composed of the output and input values of the plant at sample instant k, and previous n-1 and m-1 outputs and inputs, respectively, and d(k) represents an effect of the slowly varying disturbances to the plant. Different models of nonlinear system representations and the existence of controllers for such systems were considered in [18]. We decided for this type of an input/output representation in order to loose sometimes strict assumptions regarding a model structure, the order of the system, its relative degree (time delay) and other model parameters used at adaptive controller designs based on linear as well as nonlinear plant models [15],[19]. On the other hand this model form (1) is often used to capture the dominant nonlinear system dynamics when present immeasurable disturbances and/or possible model errors. Two approximations of NARMA model implemented by pre-trained feed-forward NNs have been proposed in [20] in order to compute a control signal to the plant to track a desired reference signal. Based on this model the off line trained Recurrent NN (RNN) is used for prediction of future outputs needed at calculations of the predictive control law based on a solution of constrained optimization problem [21]. On other hand, the RNN model in [22] is only acting as a medium for the estimation of the parameters for the linear model, in a similar manner to the role provided by recursive identification algorithms within an indirect self-tuning structure. An adaptive control law based on discrete-time plant model in pure-feedback form with state and output prediction has been used in [23]. A state space controllability model has been used in [24] and the on line tuned RNN in local linearized subsystem was used to approximate a time-delay-free nonlinear plant model. As it was noted in [20], because in all above mentioned cases some kind of the preceding NN training is required. Strictly speaking we cannot consider those cases as adaptive control problems. The latter problems occur when the identification and control are carried out simultaneously. By this we conclude that the identification part of the adaptive control algorithms must have a fast convergence.

B. The neural network plant model

In this paper, the FCRBFN [25] is used for the estimation of the output of the plant given by (1), and the predicted plant output at sample instant (k +1) follows

$$\hat{y}(k+1) = N(x,c,\sigma), \tag{2}$$

where N(x,c,σ) is the output of the FCRBFN, x is an input vector of this network (x =[yT ,uT]T) , c and σ are the vectors of centers and spreads of activation functions of neurons in its hidden layer, respectively. A calculation of the network output in (2) is carried by

$$N(x,c,\sigma) = W^T \Phi(x,c,\sigma), \tag{3}$$

where W is a weight vector of connections between hidden layer neurons of the FCRBFN model and its output, and Φ is a vector composed of activations of hidden layer neurons. Activation function of neurons in the hidden layer is Gaussian function

$$\varphi(x) = e^{-\frac{(x-c_y)^2}{\sigma_x}}, \tag{4}$$

with a center c_x and so called spread σ_x of the activation function.

Hidden layer neurons of the FCRBFN are clustered according to its inputs, that means a number of clusters is equal to the number of inputs, i.e. n+m in total. In our work, it is naturally to distribute activation function of neurons in each cluster in such a way to uniformly cover the range of the possible values of each input. Because, the range widths of the plant outputs and inputs given as

$$wr_x = \max(x) - \min(x), \tag{5}$$

Where $\boldsymbol{x}$ is substituted by y and u, respectively, are known in advance, we only have to decide about their range expansions e_x (usually expressed in percents) and the number of neurons n_{cy} in each of n clusters related to each of the plant output values $y(k)$, $y(k$ -1),…, $y(k$ - n +1) used as the first network inputs, and the number of neurons n_{cu} in each of m clusters related to each of the plant input values $u(k)$,$u(k$ -1),…,$u(k$ - m +1) used as the second m network inputs. Let the expanded output and input ranges be, respectively,

$$er_y = [\min y - e_y wr_y, \max y + e_y wr_y], \tag{6}$$

$$er_u = [\min u - e_u wr_u, \max u + e_u wr_u], \tag{7}$$

then corresponding expanded range widths are given by, respectively,

$$wer_y = \max er_y - \min er_y \tag{8}$$

$$wer_u = \max er_u - \min er_u \tag{9}$$

Let the centers of neuron activation functions inside one neuron cluster are positioned equidistantly in the expanded ranges, then the distances between two adjacent centers in neuron clusters with corresponding inputs obtained from the tapped plant output and input are given accordingly to

$$dc_y = \frac{wer_y}{n_{cy}} \;\; and \;\; dc_u = \frac{wer_u}{n_{cu}}, \tag{10}$$

where dc_y is the distance between two adjacent centers of activation functions in neuron clusters whose inputs correspond plant outputs and dc_u is the distance equivalently defined. Thus, the centers of activation functions in neuron clusters whose inputs correspond plant outputs and inputs are as follow, respectively,

$$c_{y,j} = \min er_y + \frac{dc_y}{2} + (j-1)dc_y,\ j = 1,...,n_{cy}, \tag{11a}$$

$$c_{u,j} = \min er_u + \frac{dc_u}{2} + (j-1)dc_u,\ j = 1,...,n_{cu}. \tag{11b}$$

Also, it is naturally in our work to take all neurons in the first n clusters with the equal spreads of the activation function σ_y and for the rest m neuron clusters in hidden layer to have equal spreads of the neuron activation functions σ_u . Denoting ratios of the distances between centers of adjacent neuron activation function and their corresponding spreads with sr_y and sr_u for the first n and second m neuron clusters in hidden layer, respectively, the spreads σ_y and σ_u are given by

$$\sigma_y = sr_y dc_y \;\; and \;\; \sigma_u = sr_u dc_u. \tag{12}$$

We can conclude this subsection of the paper by mentioning that our neural network plant model has $n+m$ inputs with all weights of connections from input to the hidden layer neurons put to one and with a total number of neurons in hidden layer equal to

$$n_{tn} = nn_{cy} + mn_{cu} \tag{13}$$

C. An adaptive estimation algorithm

At the begging we suppose a classical IMC control structure shown in Fig.1. in which an internal model of the plant $\widetilde{P}$ will be implemented by the FCRBFN above described. In sequel, we will present an algorithm by which NN parameters can be adjusted on line for all time without the preceding off line NN training requested at some earlier published research results [6],[8],[9],[10],[11],[12],[19],[20],[21],[22],[24]. We propose this algorithm by having in mind recent results and discussions on universal learning machines (MLNNs, ELM, the Kernel Least Mean Squares (KLMS) [13],[14],[26],[27]) and parsimonious characteristics of the FCRBFN [25] in view of the number of weights required to adjust during a learning process.

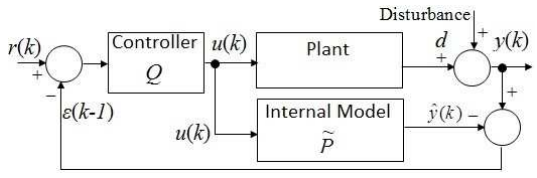

Fig. 1 The IMC structure

Subtracting a prediction of the plant output $\widetilde{y}(k+1)$ given by (2) from its true value y(k +1) at time instant (k +1) , for the model output estimation error we obtain

$$\begin{aligned} \varepsilon(k) &= y(k+1) - \hat{y}(+1) = \\ &= u(k+1) - W^T \Phi(x,c,\sigma). \end{aligned} \tag{14}$$

Based on this model estimation error $\varepsilon(k)$ we use a performance criterion of the adaptive estimation algorithm proposed here as follows

$$J(k) = \frac{1}{2}(\varepsilon(k))^2. \tag{15}$$

After the substitution of (14) into (15) we have

$$J(k) = \frac{1}{2}(y(k+1) - W^T \Phi(x,c,\sigma))^2. \tag{16}$$

Applying a SGD algorithm to adopt weights of FCRBFN defined by $\boldsymbol{W}$, an increment of the weight vector should be

$$\Delta W(k) = -\eta \nabla J(k) = \eta \Phi(x,c,\sigma)\varepsilon(k), \tag{17}$$

where η is a learning rate parameter. On other side, from (14) we see that the model output estimation error $\varepsilon(k)$ is a linear function with respect to weights given by the vector $\boldsymbol{W}(k)$. It follows from (14), that at the next time instant (k+1) the model output estimation error $\varepsilon(k+1)$ becomes

$$\begin{aligned} \varepsilon(k+1) &= y(k+2) - \\ &- (W(k) + \Delta W(k))^T \Phi(x + \Delta x, c, \sigma). \end{aligned} \tag{18}$$

After subtracting (18) from (14) and using Taylor series expansion of the vector function $\boldsymbol{\Phi}(\boldsymbol{x} + \Delta\boldsymbol{x},\boldsymbol{c},\boldsymbol{\sigma})$ on the right hand side of (18) around $\boldsymbol{x}$ we obtain

$$\begin{aligned} &\varepsilon(k+1) - \varepsilon(k) = -W(k)^T (\Phi + \frac{\partial \Phi}{\partial x}\Delta x + ...) + \\ &+ W(k)^T \Phi - \Delta W(k)^T (\Phi + \frac{\partial \Phi}{\partial x}\Delta x + ...) + \\ &+ y(k+2) - y(k+1), \end{aligned} \tag{19}$$

where $\frac{\partial \Phi}{\partial x}$ is a matrix of the first order derivatives of the $\partial\boldsymbol{x}$ neuron activation functions whose arguments are omitted for simplicity. Because all neurons in one neuron cluster i (i=1,…, n+m) of the FCRBFN have the same input x_i this matrix has diagonal form with elements

$$\left.\frac{d\varphi(x)}{dx}\right|_{x=x_i} = -\frac{2(x_i - c_x)}{\sigma_x}\varphi(x_i), \tag{20}$$

where c_x and σ_x represent corresponding center and spread of the neuron activation function inside the given neuron cluster. By inserting (17) into (19) and after some rearranging of terms, it follows

$$\begin{aligned} \varepsilon(k+1) &= [1 - \eta \Phi^T \Phi]\varepsilon(k) + \Delta y(k+2) - \\ &- W(k)^T \frac{\partial \Phi}{\partial x}\Delta x + h.o.t., \end{aligned} \tag{21}$$

where $\Delta y(k+2) = y(k+2) - y(k+1)$ and *h.o.t.* represent terms that contain products of the first order differences of plant input and plant output mutually or with weight increments defined by the vector $\Delta \boldsymbol{W}(k)$. At some operating conditions in control applications, the plant inputs and outputs usually take constant values and consequently Φ becomes a constant vector, $\frac{\partial \Phi}{\partial x}$ constant matrix and *h.o.t.* as well as $\Delta y(k+2)$ vanishes to zero. Thus, after omitting on right hand side of (21), the terms that contain first order differences of variables, for the model output estimation error one obtains a next time difference equation

$$\varepsilon(k+1) \cong [1 - \eta \Phi^T \Phi] \varepsilon(k). \tag{22}$$

Taking into account (6), the Euclidian norm of the vector neuron activation functions satisfies next inequality

$$\left\| \Phi \right\| \le \sqrt{n_{tn}}. \tag{23}$$

By this condition (23), a convergence in (22) and of the SGD algorithm in general, can be achieved if the learning rate parameter is chosen according to

$$\eta < \frac{2}{n n_{cy} + m n_{cy}} \tag{24}$$

In contrast to off line training of the NN where general recommendation on a choice of learning rate to be between 0 and 1, it is needed to dedicate a special attention at choosing right learning rate for on line training inside adaptive systems. Perhaps, in other applications related to off line data processing, if a divergence of the algorithm appears then one could try it with a smaller value of η applied to the same data set. However, we also should have in mind that at cases where the noises and outliers are present in data, those can influence an unacceptable zigzag movement of the algorithm to its minimum. Consequently, to the purpose of this work we propose to use the SGD algorithm with a momentum term, i.e. instead of (17) the parameter adaptation of the FCRBFN should be performed by

$$\Delta W(k) = \alpha \Delta W(k-1) + \eta \Phi \varepsilon(k), \tag{25}$$

where α (0 <α<1)determines a filtering characteristic of this parameter update and should be appropriately chosen.

Some advantages of the FCRBFN in respect to the standard feedforward neural network with a single hidden layer trained with the backpropagation algorithm, ELM and Support Vector Machine has been illustrated on solving benchmark problems and by simulation results in [25]. At this moment we would like to mention only two advantages of the above given SGD learning algorithm with respect to the originally used RLMS, i.e. its computational simplicity and without the need to cope with known drawbacks of RLMS algorithms [21]. Simulation results (some of these will be given in continuation of this paper) show other advantages of the SGD algorithm with the momentum term.

III. An Adaptive IMC Controller

A. The fully adaptive control design

Thanks to robustness characteristics of the IMC structure with respect to plant modeling errors as we mentioned in the introduction of this paper, many researchers used this structure as a basis to develop different control strategies. Further on, we also use the basic structure shown on Fig.1. and its corresponding modification shown on Fig.2. in order to fulfill usual requirements of the control of slow varying dynamical processes.

At the first case, we use the FCRBFN equipped with the SGD learning algorithm (25) to implement the internal model of the plant $\tilde{P}$. At cases where the nonlinearities of controlled processes are not so strongly present, it is possible to use some kind of linear controller Q in the IMC structure. Thus, instead of using linear approximation of the nonlinear plant model obtained by the RLMS estimation algorithm of its parameters [17], for example, performance improvements of the whole control system can be expected if we have better estimation of the true nonlinear plant dynamics as proposed here by (25). On the other side, a fixed part of linear controller Q can be chosen in accordance with a prior knowledge of the controlled process, that is an estimation of a range of its possible gain changes and dominant dynamics. A variable part of the linear controller Q has to be chosen so that total gain in a direct signal path (from the reference to output) is equal to 1.

Thus, let the fixed part of the Q controller is given by a low order transfer function

$$Q_f(z^{-1}) = \frac{N_Q(z^{-1})}{D_Q(z^{-1})}, Q_f(1) = 1, \tag{26}$$

then to achieve a zero steady state error at constant reference signal, it is necessary to adjust a gain of the variable part so that a total gain in the direct signal path satisfies a condition

$$Q_f(1) K_Q K_P = 1, \tag{27}$$

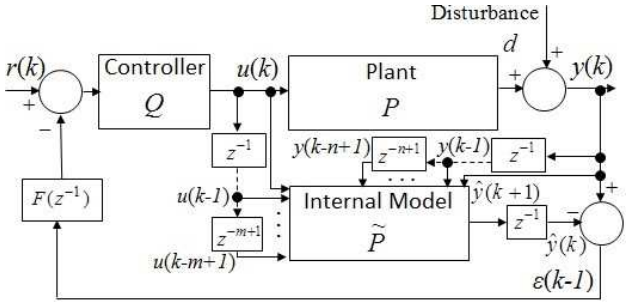

Fig. 2 The modified IMC structure

K_P is a steady state gain of the plant at the given operational point and K_Q should be adjusted. As we have assumed that this gain might change during system operation, then in order to provide zero steady state error

$$e_{ss} = \lim_{k \to \infty} e(k) = \lim_{k \to \infty} (r(k) - y(k)), \tag{28}$$

it is possible to use a simple approximation to determine the steady state gain of the plant by

$$K_P = \frac{y_{SS}}{u_{SS}}, \tag{29}$$

where y_{SS} and u_{SS} are some averaged values of plant outputs and inputs, respectively. These averaged values can be computed as averages of a few preceding values, before and including a current time sample k , i.e.

$$y_{SS} = \frac{\sum_{i=0}^{N-1} y(k-i)}{N} \quad and \quad u_{SS} = \frac{\sum_{i=0}^{N-1} u(k-i)}{N}, \tag{30}$$

or by a low pass filtering of the plant outputs and inputs, respectively.

Finally, the gain of the variable part of the linear controller Q should be adjusted according

$$K_Q = \frac{1}{K_P}. \tag{31}$$

In the second case, it is supposed that the control plant has much higher changes of its dynamic characteristics in an operational range and the plant estimation model is not enough accurate as a consequence of not having inputs that satisfies the persistent excitation condition [15],[17]. For example, at some steady states there always exist some offsets in the model estimation error $\varepsilon(k)$. Although these offsets might be of small values when plant gains are high, one cannot provide above mentioned condition related to the direct signal path.

IV. An Illustrative Example and Performance Comparison

Example 1: In performed simulations, the adaptive control system based on the plant internal model implemented by FCRBFN and provided with the SGD algorithm with the momentum term has been compared to the adaptive system under same conditions except the estimation algorithm has been replaced by the RLMS one. The true plant model in the considered systems is given by a discrete time state space equations as follows

$$\begin{aligned} x_1(k+1) &= 0.9x_1(k) - 0.4x_2(k) + 0.1x_2(k) + 0.9x_4(k), \\ x_2(k+1) &= x_1(k), \\ x_3(k+1) &= x_4(k), \\ x_4(k+1) &= u(k), \end{aligned} \tag{32}$$

and the plant output is represented by

$$\begin{aligned} v(k) &= 0.9x_1(k) - 0.4x_2(k) + 0.1x_3(k) + 0.9x_4(k), \\ y(k) &= \frac{y(k-1)v(k-1) + v(k)}{1 + y(k-1)^2}, \end{aligned} \tag{33}$$

The FCRBFN architecture in both adaptive system is the same with parameters as follow: n= 4 , n_{cy} = 2, max y=1.5, min y= 0, e_y = 0.5 , sr_y =1, m = 3 , n_{cu} = 2 , max$u$ =1, min$u$ = 0 , e_u = 0.5, sr_u =1.

Parameters of the SGD learning algorithm with the momentum term are chosen as: η=0.11, α=0.2. An initial value of the weight vector is chosen as a random vector with components inside [0, 1].

In order to compare performance, the RLMS learning algorithm in another adaptive system has been started with the same initial weight vector as above mentioned and the covariance matrix set to be a unit diagonal matrix. Comparison of the performance is based on three criteria, that is on computing of following indexes

$$\begin{aligned} \varepsilon_{MSE} &= \frac{1}{N}\sum_{k=1}^{N} \varepsilon(k)^2, u_{MSE} = \frac{1}{N}\sum_{k=1}^{N} u(k)^2, \\ e_{AAE} &= \frac{1}{N}\sum_{k=1}^{N} |e(k)|, \end{aligned} \tag{34}$$

whose values are summarized in Table I. It is apparent from results given in Table I. that the adaptive system equipped with the SGD learning algorithm has better performance than the adaptive system with the RLMS algorithm. Results of simulation related to the adaptive system that uses the SGD learning algorithm are shown on Fig.3, Fig.4 and Fig.5.

TABLE I Performance Comparison

eAAE	εMSE	uMSE	eAAE
SGD	0.0432	0.0142	0.1299
RLS	0.0791	0.0154	0.1344

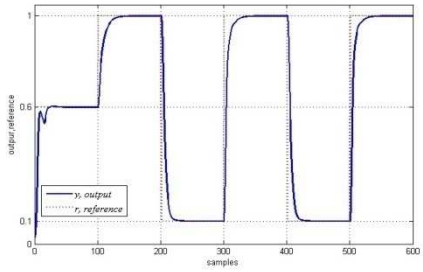

Fig. 3 Output y and reference r of the system

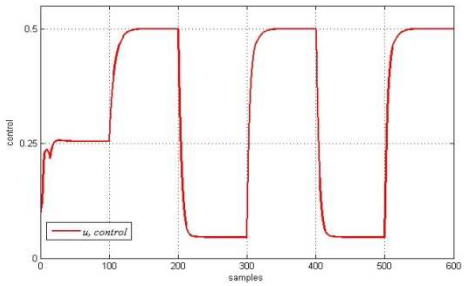

Fig. 4 Control action u

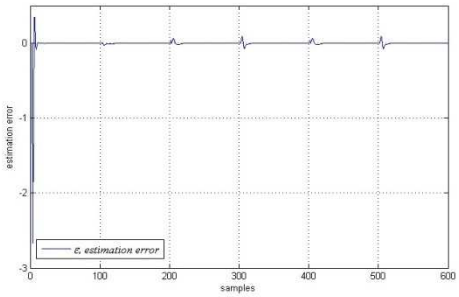

Fig. 5 Estimation error ε

V. Conclusion

In this paper we have presented a fully adaptive neural controller in the IMC system structure. An internal plant model as the essential part of the used control structure is realized by the simple RBF NN architecture provided with the SGD learning algorithm. For this SGD algorithm we have proved a convergence and shown some performance testing through an example. We have also presented a simple way for an adjustment of the gain in the proposed adaptive neural controller design required to provide zero steady-state error in the cases of constant reference signals and constant disturbances.

References

[1] M. Morari and E. Zafiriou, *Robust Process Control*, Prentice-Hall Int, 1989.

[2] J. M. Sanches and J. Rodellar, Adaptive Predictive Control: From the concepts to plant optimization, Prentice Hall, 1996.

[3] J. Igic, M. Bozic, P. Maric, and I. Krcmar, "Adaptive Control based on Internal Model", *Proc. XLV ETRAN, Conference* vol. 1, pp. 209-212, Bukovicka Banja, June, 2001.

[4] K.J. Hunt, and D. Sbarbaro, "Neural networks for nonlinear internal model control", *IEE Proceedings-D*, vol. 138, no. 5,pp. 431-438, September, 1991.

[5] I.Rivals, and L.Personnaz, "Nonlinear internal model control using neural networks: Application to processes with delay and design issues," *IEEE Transactions Neural Networks*, vol. 11, no.1, pp 80–90, January, 2000.

[6] M. Mohammadzaheri, L. Chen, and S. Grainger, "A critical review of the most popular types of neuro control", *Asian Journal of Control*, vol. 14, no. 1, pp. 1-11, January 2012.

[7] V.G. Krishnapura, and A. Jutan, "A neural adaptive controller", *Chemical Engineering Science 55*, pp. 3803-3812, 2000.

[8] H Deng, and H.-X. Li, "A Novel Neural Approximate Inverse Control for Unknown Nonlinear Discrete Dynamical Systems", *IEEE Transactions on Systems, Man, and Cybernetics*—Part B: Cybernetics,vol. 35, no. 1, pp. 553-567, February, 2005.

[9] H.-X. Li, and D. Hua, "An Approximate Internal Model-Based Neural Control for Unknown Nonlinear Discrete Processes," *IEEE Trans. on Neural Networks*, vol. 17, no. 3,pp. 659-670, May, 2006.

[10] Y-N. Wang, and X-F. Yuan, "SVM Approximate-based Internal Model Control Strategy", *Acta Automatica Sinica*, vol. 34, no. 2, pp. 172-179, February, 2008.

[11] J. Igic and M. Bozic, "A Modified Approximate Internal Model-based Neural Control for the Typical Industrial Processes", *Electronics*, vol.18, no. 1, pp. 46 –53, June, 2014.

[12] K.S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 1–27, 1990.

[13] G-B. Huang and Q-Y. Zhu, Siew C-K. Extreme learning machine: theory and applications. *Neurocomputing*; 70:489–501, 2006.

[14] G-B. Huang, "What are Extreme Learning Machines? Filling the Gap between Frank Rosenblatt's Dream and John von Neumann's Puzzle", Springer Science+Business Media New York, Cogn Comput 7:263–278, DOI 10.1007/s12559-015-9333-0, 2015.

[15] K.J. Astrom, B. Wittenmark, "Adaptive control", 1989.

[16] M. Bozic, Lj. Kuljaca: "Square Root Algorithm for LS Type Estimators and Comparison of Adaptive Systems with Extended GPC", Automatika 32, no 5-6, pp. 155-164,1991.

[17] L. Ljung, *System Identification-Theory for the User*, 2nd Edition, A Simon & Schuster Company, N.J., Prentice-Hall PTR, 1999.

[18] J.B.D. Cabrera, and K.S. Narendra, "Issues in the application of neural networks for tracking based on inverse control", *IEEE Transactions on Automatic Control*, vol. 44, no. 11, pp. 2007–2027, November 1999.

[19] P.M. Mills, A.Y. Zomaya and M.O. Tade, *Neuro-Adaptive Process Control: A Practical Appoach*, John Willey&Sons, 1996.

[20] K.S. Narendra and S. Mukhopadhyay, "Adaptive Control Using Neural Networks and Approximate Models", *IEEE Transactions on Neural Networks*, vol. 8, pp. 475–485, 1997.

[21] K. Patan, "Neural Network-Based Model Predictive Control: Fault Tolerance and Stability", *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 1147-1155, 2015.

[22] G. Lightbody and G.W. Irwin, "Nonlinear Control Structures Based on Embedded Neural System Models",*IEEE Transactions on Neural Networks*, vol 8, no. 3, pp. 553-567, 1997

[23] S.S. Ge, C. Yangand T.H. Lee, "Adaptive Predictive Control Using Neural Network for a Class of Pure-Feedback Systems in Discrete Time",*IEEE Transactions on Neural Networks*, vol. 19, no. 9,pp. 1599-1614, 2008.

[24] J.Q. Huang, F.L. Lewis,"NN Predictive control for nonlinear Dynamic systems with time delays", *IEEE Transactions on Neural Networks*, vol.14, no2, 2003.

[25] D. Kosic, "Fast Clustered Radial Basis Function Network as an adaptive predictive controller", *Neural Networks, Elsevier*, no 63, pp. 79-86, 2015.

[26] J.C. Principe and B. Chen, "Universal Approximation with Convex Optimization: Gimmick or Reality?", *IEEE Computational Intelligence Mag.*, vol. 10, no.2, pp. 68-77, 2015.

[27] W. Liu, P. Pokarel and J. Principe, "Kernel LMS algorithm" *IEEE Trans. Signal Processing*, vol 56, pp. 543-554, 2008.