International Journal of
Applied Mathematics,
Electronics and Computers
ISSN: 2147-8228 http://ijamec.atscience.org
Advanced Technology and Science
Original Research Paper

EKF Based Generalized Predictive Control of Nonlinear Systems

Erdem Dilmen *[1], Selami Beyhan[2]

Accepted 3rd September 2016

Abstract: In this paper, Autoregressive with exogenous input (ARX) and dynamic neural network (DNN) based generalized predictive control (GPC) methods are designed to control of nonlinear systems. ARX and DNN models adaptively approximate the plant dynamics and predict the future behavior of the nonlinear system. While control process goes on, the poles of the ARX and DNN models are constrained in a stable region using a projection operator for structural stability. Simulation results are given to compare the tracking performances of the methods. ARX-GPC and DNN-GPC both yield good tracking performances while keeping the changes in control signal as low as possible. The simulation results show that even though ARX is a linear model, it provides acceptable tracking results as well as DNN model.

Keywords: *Generalized predictive control, ARX, dynamic neural network, Kalman filter and extended Kalman filter, nonlinear systems and adaptive learning rate.*

1. Introduction

Identification of systems with unknown mathematical model, control and fault detection studies keep their importance up-to-date [1]. It is a hard task to obtain a mathematical model in complex systems. In addition, need for system modelling emerges due to uncertainty, disturbances and noises. In such cases, robust or adaptive control techniques should be adopted to design a controller [8]. In adaptive control, an appropriate model is selected to approximate the actual system model. Artificial neural network (ANN) and fuzzy networks are widely used as function estimators [25, 2].

GPC is a class of model-based predictive control. When dynamics of the system are not known, the model which approximates the system has a crucial role in both predicting the future behaviour of the system and generating the future control signals. In each sampling instant first one of the produced control signals is applied to the system [4, 3]. The models which are able to adapt to the changing dynamics of systems improve GPC performance. Artificial neural network (ANN) [21, 17] and support vector machine (SVM) [9, 14] can be adopted in GPC studies.

DNNs [20] have a recursive structure. Internal states and external inputs constitute the actual inputs of these networks which are widely used in function approximation. DNNs can approximate complex nonlinear dynamics due to the merits of recursive structure and nonlinear activation function [22]. Usually they are used in prediction [6], fault detection [12], adaptive control [16] and GPC [24].

In this study, particular type DNN and AutoRegressive with eXagenous input (ARX) model based GPC are performed. ARX is a linear model while both linear and nonlinear dynamics exists in the DNN model. Models are trained by extended Kalman filter (EKF).

In addition to adaptation of the parameters, poles are adapted such that they are bounded to maintain stability of the model. Thus, quick and stable online identification is obtained. In numerical simulations, bioreactor and continuously-stirred tank reactor (CSTR) benchmark systems are controlled using two methods and results are compared.

Rest of the paper is organized as follows. In Section 2, DNN and ARX models are presented; model stability and EKF-based training are introduced. GPC and ARX and DNN based GPC are detailed in Section 3. Simulations are given in Section 4. Finally, paper is concluded in Section 5.

2. Identification Models

In this section, ARX model and the DNN model which is used in continuous time adaptive control in [13] will be studied for discrete time GPC. Section 2.1 and 2.2 present the ARX and DNN models while Section 2.3 introduces the model stability. Finally, EKF-based training of the models is given in Section 2.4.

2.1. ARX Model

ARX model is simple and linear. Consider n_u and n_y are the past input and output samples to construct the ARX data. Let (1) give the output equation of a system at time instant k whose explicit output expression is unknown.

$$\begin{aligned} y_k &= f(c_k) \\ c_k &= (u_k, u_{k-1}, \ldots, u_{k-n_u}, y_{k-1}, \ldots, y_{k-n_y}) \end{aligned} \quad (1)$$

Output can be approximated linearly by a parameter vector $\boldsymbol{\theta}_k \in R^{n_u+1+n_y}$ such as

$$\hat{y}_k = c_k \boldsymbol{\theta}_k^T \quad (2)$$

2.2. Dynamic Neural Network

Figure 1 presents internal dynamics of one neuron in the dynamic

[1] *Mechatronics Engineering, Pamukkale University, Technology Faculty, Denizli 20020, Turkey.*
**Corresponding Author: Email: edilmen@pau.edu.tr*
[2] *Electrical and Electronics Engineering, Pamukkale University, Denizli 20070, Turkey. Email: sbeyhan@pau.edu.tr*
Note: This paper has been presented at the 3rd International Conference on Advanced Technology & Sciences (ICAT'16) held in Konya (Turkey), September 01-03, 2016.

NN. States of the neurons are updated in discrete-time as given in (3) and (4).

$$x_{k+1} = Dx_k + AT(x_k) + Bc_k \tag{3}$$

$$y_k = Cx_k \tag{4}$$

Here, $x \in R^n$ is state vector, $c \in R^m$ and $y \in R^m$ are input and output vectors respectively. Nonlinear activation function $T(.)$ is tangent hyperbolic function and is expressed as

$$T(x_k) = \begin{bmatrix} \tanh(\hat{x}_{1k}) \\ \vdots \\ \tanh(\hat{x}_{nk}) \end{bmatrix} \tag{5}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6}$$

$B \in R^{n \times m}$, $C \in R^{m \times n}$ and the matrices D and A are as follows.

$$D = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}_{n \times n} \tag{7}$$

$$A = \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{bmatrix}_{n \times n} \tag{8}$$

DNN model consists of linear ARX (AutoRegressive with eXogeneous input) model and nonlinear function tanh(.). Thus, identification of linear and nonlinear dynamic systems is aimed in a quick and efficient way.

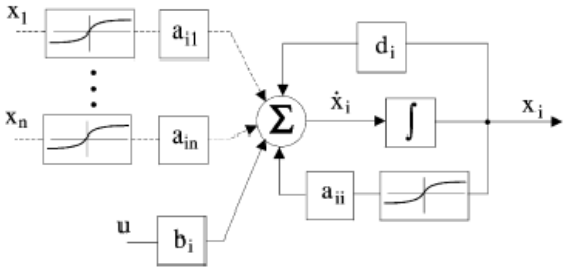

Figure 1: Dynamic neuron structure.

This model can be considered as sparsed Hopfield type network in means of inter-neuron connections. Connection of each neuron with itself brings to the network dynamic property. Dashed lines in Figure 1 show nonexisting connections. From this aspect, model is diagonal. Figure 2 presents diagonal dynamic model.

2.3. Model Stability

The results in [7, 10] were reviewed to analyse the stability of the models and stability conditions given in [15] were followed in this study.

First, ARX model stability is analysed. In the simulations ARX model is used as a second order IIR (Infinite Impulse Response) filter since $n_y = 2$ is taken (see Section 4).

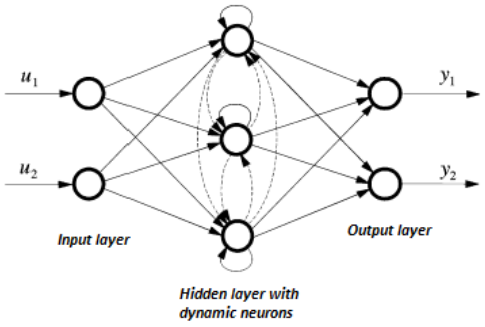

Figure 2: Diagonal dynamic model.

Parameters in the parameter vector $\boldsymbol{\theta}_k$ which correspond to the past output samples are constrained with the stability condition as following.

$$\begin{cases} 1 - \theta_{k,n_u+2} + \theta_{k,n_u+3} > 0 \\ 1 + \theta_{k,n_u+2} + \theta_{k,n_u+3} > 0 \\ 1 - \theta_{k,n_u+3} > 0 \end{cases} \tag{9}$$

Second, DNN model stability is analysed. State equation belonging to the i_{th} neuron in the discrete-time DNN model is given by

$$\hat{x}_i(k+1) = d_i \hat{x}_i(k) + a_i T(\hat{x}_i(k)) + b_i u_i(k), \quad i = 1,...,n \tag{10}$$

k is the time index, definition of the variables and model matrices are given in the previous section. Discrete-time state solution of the model given $x(0) = x_0$ as the initial state is obtained as follows.

$$x(k) = D^k x_0 + \sum_{i=0}^{k-1} D^{k-1-i}(AT(x(i)) + Bu) \tag{11}$$

Stability of the DNN model is directly related to the stability of the states since output vector of the DNN model is linearly related to the states as given in Equation (4). Neuron in Equation (10) can be considered as a first order ARX model. Stability of the DNN model expressed in Equation (3) is guaranteed by satisfying the following conditions.

$$\begin{aligned} |d_i| &< 1 \\ a_i &\neq 0 \\ b_{ij} &\neq 0, i = 1,...,n, j = 1,...,m \end{aligned} \tag{12}$$

Generally, $b_{ij} \neq 0$ is needed to be satisfied not for the internal stability but for the input to have effect on the output.

2.4. Inequality Constrained EKF Based Training

Inequality constrained EKF (ICEKF) is used to train model parameters. Parameter adaptation equations of ICEKF will be given in Section 2.4.1 and training of models will be detailed in Section 2.4.2.

2.4.1. ICEKF

State-space representation of a nonlinear discrete-time system is given in (13).

$$x_k = f(x_{k-1}) + w_{k-1} \tag{13}$$

$$y_k = h(x_k) + v_k \tag{14}$$

Here, $x \in R^n$ is state vector, $y \in R^m$ is system output, $f(.)$ is nonlinear process function vector, $h(.)$ is nonlinear measurement function vector, $w \in R^n$ is process noise and $v \in R^m$ is measurement noise, which both is Gaussian white noise. Extended Kalman filter can be summarized in two steps given Q and R are covariance matrices corresponding to process and measurement noises respectively.

Time Update

$$\begin{aligned} x_k^- &= f(x_{k-1}) \\ P_k^- &= J\ f(x_{k-1})P_{k-1}J\ f^T(x_{k-1}) + Q_{k-1} \\ K_k &= P_k^- J\ h^T(x_k^-)(J\ h(x_k^-)P_k^- J\ h^T(x_k^-) + R_k)^{-1} \end{aligned} \tag{15}$$

Measurement Update

$$\begin{aligned} x_k &= x_k^- + K_k(y_k - h(x_k^-)) \\ P_k &= (I - K_k J\ h(x_k^-))P_k^- \end{aligned} \tag{16}$$

In (15) and (16), $P \in R^{n \times n}$ is state estimation error covariance matrix and K_k is Kalman gain. $J_f(.)$ and $J_h(.)$ are given in (17) and (18).

$$J_f(x_{k-1}) = \frac{\partial f(x)}{\partial x}|_{x=x_{k-1}} \tag{17}$$

$$J_h(x_k^-) = \frac{\partial h(x)}{\partial x}|_{x=x_k^-} \tag{18}$$

Let us consider there exist inequality constraints on states as given below in state estimation by ICEKF.

$$Mx \le N \tag{19}$$

Parameters must be within the constraint space to satisfy the constraints. By this reason, projection is taken onto the constraint space [19].

$$\begin{aligned} x_k &= x_k - P_k M^T (MP_k M^T)^{-1}(Mx_k - N) \\ dP_k &= P_k M^T (MP_k M^T)^{-1} M \\ P_k &= P_k - dP_k P_k \end{aligned} \tag{20}$$

(20) presents the projection onto the constraint space as it yields the final optimum state estimation. x_k and P_k stand for projected state estimation and corresponding error covariance matrix respectively.

2.4.2. Training of Models by ICEKF

Parameters of the ARX model exist directly in the parameter vector θ while parameters of the DNN model are in the matrices A, B, C ve D. In ARX case $\theta \in R^{n_u+1+n_y}$. In DNN case, a parameter vector $\theta \in R^{2(n+n\times m)}$ is generated as in (21) similar to [11] and [18].

$$\begin{aligned} a &= [a_1 \ldots a_n]^T \\ b &= [b_{11} \ldots b_{nm}]^T \\ c &= [c_{11} \ldots c_{mn}]^T \\ d &= [d_1 \ldots d_n]^T \\ \theta &= [a^T b^T c^T d^T]^T \end{aligned} \tag{21}$$

When the parameter vector is obtained, parameters in both models are updated as follows.

$$\begin{aligned} \hat{\theta}_k &= \hat{\theta}_k^- + K_k(y_k - y(\hat{\theta}_k^-)) \\ K_k &= P_k^- J\ h^T(\hat{\theta}_k^-)(J\ h(\hat{\theta}_k^-)P_k^- J\ h^T(\hat{\theta}_k^-) + R_k)^{-1} \\ P_k &= (I - K_k J\ h(\hat{\theta}_k^-))P_k^- \end{aligned} \tag{22}$$

In (22), $P \in R^{t \times t}, t = 2(n + n \times m)$ is estimation error covariance matrix of the parameters in the vector θ and J_h is updated as follows.

$$J_h(\hat{\theta}_k^-) = \frac{\partial y(\hat{\theta}_k^-)}{\partial \hat{\theta}_k^-} \tag{23}$$

Projection onto the constraint space is taken as in (20) to satisfy the constraints in (9) for ARX model while in (12) for DNN model. For the DNN case, states are updated as in (3) using the parameters which are updated satisfying the constraints. Thus, states are not the ones which are estimated optimally but parameters in the vector θ are.

3. ARX and DNN Based Generalized Predictive Control

Section 3.1 briefly introduces GPC while Section 3.2 and 3.3 detail the ARX-based and DNN-based GPC respectively.

3.1. GPC

Let us have a NARX (Nonlinear Auto-Regressive with exogenous input) data model for a nonlinear system [9].

$$y_n = f(u_n, u_{n-1}, \ldots, u_{n-n_u}, y_{n-1}, \ldots, y_{n-n_y}) \tag{24}$$

As $\tilde{y}_n$ is the reference signal and $\hat{y}_n$ is the model output, GPC aims to have the model output track the reference signal with minimum tracking error possible while keeping the changes in the control signal as low as possible. Constraints exist on the control signal.

$$\begin{aligned} \min_u f(u) = & \sum_{k=1}^{K_y} (\tilde{y}[n+k] - \hat{y}[n+k])^2 + \\ & \lambda \sum_{k=0}^{K_u} (u[n+k] - u[n+k-1])^2 \\ = & (Y - Y)^T (Y - Y) + \lambda u^T L u \\ & - 2\lambda u[n]u[n-1] + \lambda u^2[n-1] \\ \text{Constraints}: & \ u_{min} \le u[n+k] \le u_{max}, k = 0,1,\ldots,K_u \\ & \ |u[n+k] - u[n+k-1]| \le \Delta u_{max}, k = 1,\ldots,K_u \end{aligned} \tag{25}$$

$$L=\begin{bmatrix} 2 & -1 & 0 & \dots & 0 & 0 \\ -1 & 2 & -1 & \dots & 0 & 0 \\ 0 & -1 & 2 & \ddots & 0 & 0 \\ 0 & 0 & -1 & \ddots & -1 & 0 \\ 0 & 0 & 0 & \ddots & 2 & -1 \\ 0 & 0 & 0 & \ddots & -1 & 1 \end{bmatrix} \tag{26}$$

λ, penalizes the abrupt changes in control signal. K_u and K_y are horizon values up to how many future control inputs and model outputs will be calculated. Always, $K_u < K_y$ is satisfied. $Y=[\tilde{y}_{n+1}\dots\tilde{y}_{n+K_y}]^T \in R^{K_y}$ and $Y=[\hat{y}_{n+1}\dots\hat{y}_{n+K_y}]^T \in R^{K_y}$. At each time instant, to make the system output track the reference signal for future K_y samples, control input vector $u=[u_n\dots u_{n+K_u}]^T \in R^{K_u+1}$ is generated and first element in the vector is applied to the system. To predict the future behaviour of the system whose output function is unknown, an appropriate model to approximate that system is used. Also that model is utilised for the gradient information which is necessary to update the control signal at each time instant. This update is expressed as follows.

$$u_k = u_{k-1} + \Delta u \tag{27}$$

When Gauss-Newton modification is employed,

$$\Delta u = -\mu\left(\frac{\delta^2 f(u)}{\delta u^2}\right)^{-1}\frac{\delta f(u)}{\delta u} \tag{28}$$

is obtained and μ denotes the step size. It can be calculated optimally using one of the one-dimensional optimization methods in the literature. Gradient vector is the following.

$$\frac{\delta f(u)}{\delta u} = -2\left(\frac{\delta Y_n}{\delta u}\right)^T (Y_n - Y_n) + 2\lambda L u - 2\begin{bmatrix} \lambda u[n-1] \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{29}$$

If derivative of (29) is taken to obtain Hessian matrix, a term $\left(\frac{\delta^2 Y_n}{\delta u^2}\right)$ is generated. But it can be ignored since it has small value. Thus, Hessian is approximated as

$$\frac{\delta^2 f(u)}{\delta u^2} \cong 2\left(\frac{\delta Y_n}{\delta u}\right)^T\left(\frac{\delta Y_n}{\delta u}\right) + 2\lambda L \tag{30}$$

Considering (24), model output depends on the control signals that have a time index value which is smaller than or equal to that of the model output. Thus, the term $\left(\frac{\delta Y_n}{\delta u}\right)$ is expressed as follows.

$$\frac{\delta Y_n}{\delta u} = \begin{bmatrix} \frac{\delta\hat{y}[n+1]}{\delta u[n]} & \frac{\delta\hat{y}[n+1]}{\delta u[n+1]} & 0 & \dots & 0 \\ \frac{\delta\hat{y}[n+2]}{\delta u[n]} & \frac{\delta\hat{y}[n+2]}{\delta u[n+1]} & \frac{\delta\hat{y}[n+2]}{\delta u[n+2]} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\delta\hat{y}[n+K_y]}{\delta u[n]} & \frac{\delta\hat{y}[n+K_y]}{\delta u[n+1]} & \frac{\delta\hat{y}[n+K_y]}{\delta u[n+2]} & \dots & \frac{\delta\hat{y}[n+K_y]}{\delta u[n+K_u]} \end{bmatrix} \tag{31}$$

Computational load of the derivatives depends on the model.

3.2. ARX Based GPC

Since the system is approximated by ARX model, $\frac{\delta Y_n}{\delta u}$ is obtained based on that model. Considering (2), future output prediction by the model is given by

$$\hat{y}_{k+l} = c_{k+l}\boldsymbol{\theta}_k^T, \quad l=1,\dots,K_y \tag{32}$$

It can be written explicitly

$$\hat{y}_{k+l} = \sum_{i=1}^{n_y}\theta_{k,n_u+1+i}\hat{y}_{k+l-i} + \sum_{i=0}^{n_u}\begin{cases}\theta_{k,i+1}u_{k+l-i} & l-K_u < i \\ \theta_{k,i+1}u_{k+K_u} & l-K_u \geq i\end{cases} \tag{33}$$

Thus, partial derivative expression will be

$$\frac{\delta\hat{y}_{k+l}}{\delta u_{k+h}} = \sum_{i=1}^{n_y}\theta_{k,n_u+1+i}\frac{\delta\hat{y}_{k+l-i}}{\delta u_{k+h}}\delta_1(l-i-1) + \sum_{i=0}^{n_u}\begin{cases}\theta_{k,i+1}\delta_{k-i,h} & l-K_u < i \\ \theta_{k,i+1}\delta_{K_u,h} & l-K_u \geq i\end{cases} \tag{34}$$

δ_1 is the unit step function. Obtained partial derivative is substituted in (29) and (30) to obtain the gradient vector and Hessian matrix.

3.3. DNN Based GPC

Since the system is approximated by DNN model, $\frac{\delta Y_n}{\delta u}$ is obtained based on that model. Using (3) and (4), model output can be written alternatively as following.

$$\hat{y}_k = \sum_{j=1}^{n} c_j(d_j x_j + a_j T(x_j) + [b_{j1}\dots b_{jm}]u_k) \tag{35}$$

$$v_{jk} = [b_{j1}\dots b_{jm}]u_k \tag{36}$$

$$\hat{y}_k = \sum_{j=1}^{n} c_j(d_j x_j + a_j T(x_j) + v_{jk}) \tag{37}$$

And the future output prediction by the model is given by

$$\hat{y}_{k+l} = \sum_{j=1}^{n} c_j(d_j x_j + a_j T(x_j) + v_{j,k+l}), \quad l=1,\dots,K_y \tag{38}$$

where

$$v_{j,k+l} = \sum_{i=1}^{n_y} b_{j,n_u+1+i}\hat{y}_{k+l-i} + \sum_{i=0}^{n_u}\begin{cases}b_{j,i+1}u_{k+l-i} & l-K_u < i \\ b_{j,i+1}u_{k+K_u} & l-K_u \geq i\end{cases} \tag{39}$$

Thus, partial derivative expression will be

$$\frac{\delta\hat{y}_{k+l}}{\delta u_{k+h}} = \sum_{j=1}^{n} c_j\frac{\delta v_{j,k+l}}{\delta u_{k+h}} \tag{40}$$

$$\frac{\delta v_{j,k+l}}{\delta u_{k+h}} = \sum_{i=1}^{n_y} b_{j,n_u+1+i}\frac{\delta\hat{y}_{k+l-i}}{\delta u_{k+h}}\delta_1(l-i-1) + \sum_{i=0}^{n_u}\begin{cases}b_{j,i+1}\delta_{k-i,h} & l-K_u < i \\ b_{j,i+1}\delta_{K_u,h} & l-K_u \geq i\end{cases} \tag{41}$$

δ_1 is the unit step function. Obtained partial derivative is substituted in (29) and (30) to obtain the gradient vector and Hessian matrix.

4. Computer Simulations

ARX-based and DNN-based GPC controllers were tested on bioreactor [5] and continuously stirred tank reactor [23] systems.

4.1. Bioreactor Control

Bioreactor is a second order benchmark system to test tracking performance of the controllers.

$$\begin{aligned} \dot{x}_1(t) = & -x_1(t)u(t) + x_1(t)(1-x_2(t))e^{\left(\frac{x_2(t)}{\eta}\right)} \\ \dot{x}_2(t) = & -x_2(t)u(t) \\ & + x_1(t)(1-x_2(t))e^{\left(\frac{x_2(t)}{\eta}\right)}\left(\frac{1+\beta}{1+\beta-x_2(t)}\right) \\ & \eta_{nom} = 0.48, \beta = 0.02, y(t) = x_1(t) \end{aligned} \quad (42)$$

$x_1(t)$ and $x_2(t)$ are cell concentration and amount of nutrients per volume. Control signal $u(t)$ is the flow rate. η is time-varying parameter of the process and has a nominal value of 0.48. Controllers were employed to make the reactor output track desired reference value with minimum tracking error possible. GPC parameters are $K_y = 10$, $K_u = 2$, $\lambda = 0.01$, $u_{min} = 0$, $u_{max} = 2$, $\Delta u_{max} = 0.2$, $n_u = 2$, $n_y = 2$. Sampling period is $T_s = 0.01$ s. For the DNN-based GPC case, number of states is set $n = 5$. In the simulation, $\eta(t) = 0.48 + 0.06sin(0.05\pi t)$ oscillates around its nominal value. Figure 3 shows reference tracking and produced control signal by ARX-based GPC controller while Figure 4 shows time evolution of parameters.

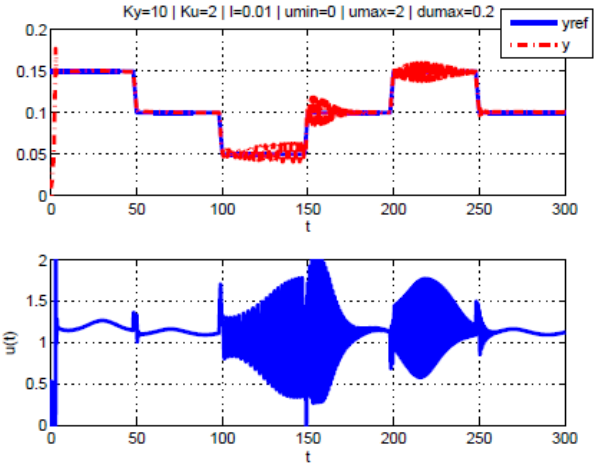

Figure 3: Bioreactor reference tracking and control signal produced by ARX-based GPC controller.

Table 1: Comparison of tracking RMSE and P_u by ARX-based and DNN-based GPC controllers in bioreactor system.

Controller	RMSE tracking	P_u
ARX-based GPC	**0.0105**	**1.3401**
DNN-based GPC	**0.0099**	**1.2952**

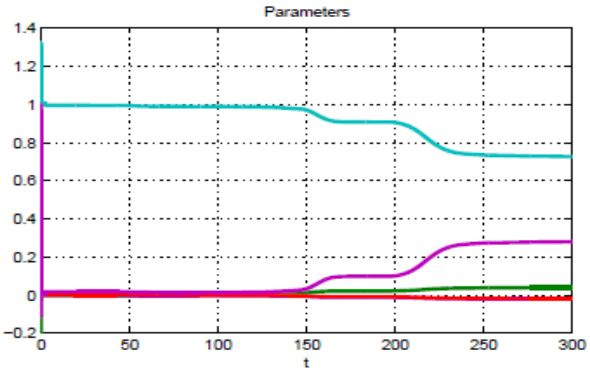

Figure 4: Time evolution of parameters in control of bioreactor by ARX-based GPC controller.

Similarly, Figure 5 shows reference tracking and produced control signal by DNN-based GPC controller while Figure 6 shows time evolution of parameters. Tracking performances of the controllers are compared in terms of Root-Mean-Squared-Error (RMSE) and power of the produced control signal ($P_u = \frac{1}{N}\sum_{k=1}^{N} |u[n]|^2$) in Table 1.

ARX-based GPC controller tracks the reference signal with little oscillations while DNN-based GPC controller performs smoother tracking as expected. However, it has more overshoot compared to ARX-based GPC. Control signal is less aggressive in DNN-based GPC. ARX-based GPC has still acceptable tracking performance.

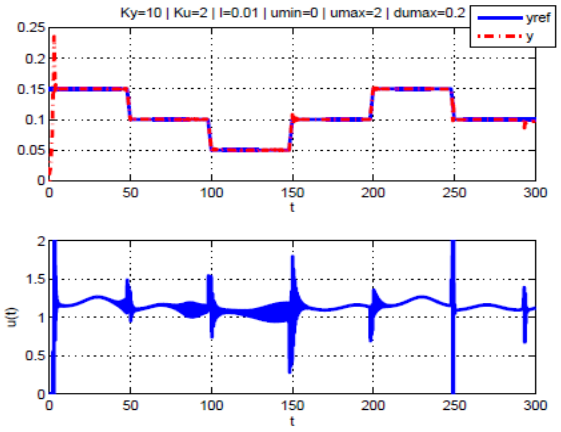

Figure 5: Bioreactor reference tracking and control signal produced by DNN-based GPC controller.

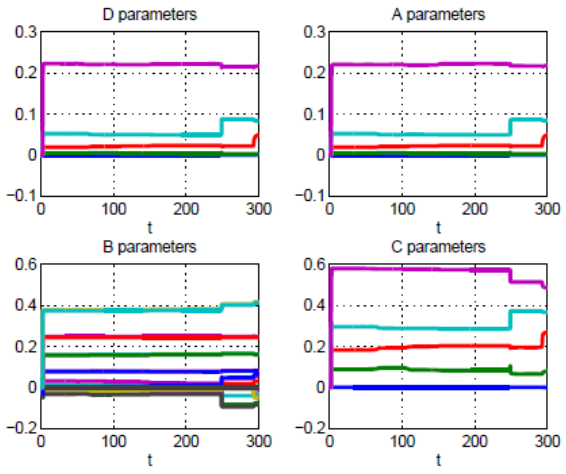

Figure 6: Time evolution of parameters in control of bioreactor by DNN-based GPC controller.

4.2. CSTR Control

Continuously stirred tank reactor is a third order highly nonlinear system on which tracking performance of the controllers were tested.

$$\begin{aligned} \dot{x}_1 &= 1 - x_1 - D_{a1}x_1 + D_{a2}x_2^2 \\ \dot{x}_2 &= -x_2 + D_{a1}x_1 - D_{a2}x_2^2 - D_{a3}d_2x_2^2 + u \\ \dot{x}_3 &= -x_3 + D_{a3}d_2x_2^2 \\ & D_{a1} = 3, D_{a2} = 0.5, D_{a3} = 1, d_2 nom = 1, y = x_3 \end{aligned} \quad (43)$$

$u(t)$ is the control signal and d_2 is time-varying parameter of the process which has a nominal value of 1. Controllers were employed to make the reactor output track desired reference value with minimum tracking error possible. GPC parameters are $K_y = 10$, $K_u = 2$, $\lambda = 0.1$, $u_{min} = 0$, $u_{max} = 1$, $\Delta u_{max} = 0.1$, $n_u = 2$, $n_y = 2$. Sampling period is $T_s = 0.1$ s. For the DNN-based GPC case, number of states is set $n = 5$. Figure 7 shows reference tracking and produced control signal by ARX-based GPC controller while Figure 8 shows time evolution of parameters.

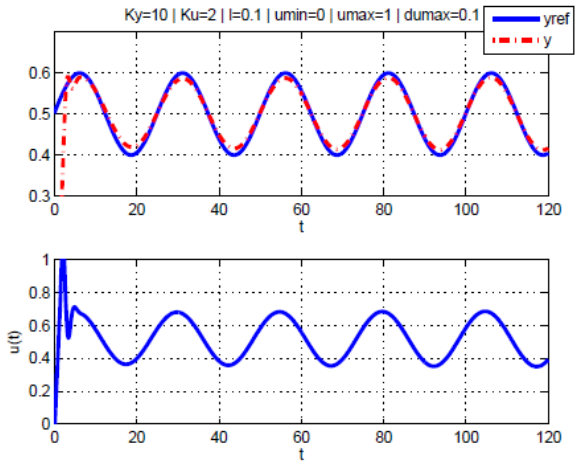

Figure 7: CSTR reference tracking and control signal produced by ARX-based GPC controller.

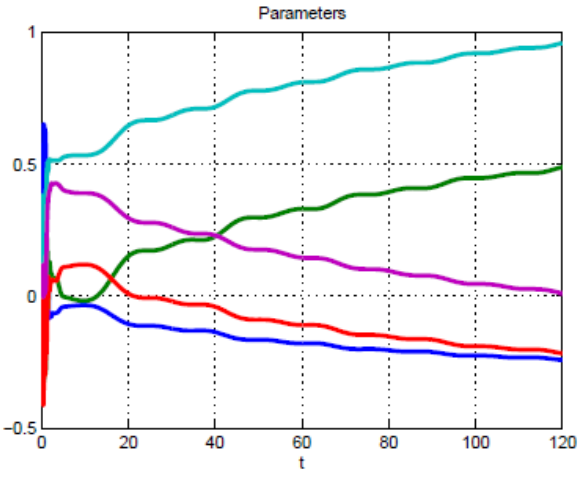

Figure 8: Time evolution of parameters in control of CSTR by ARX-based GPC controller.

Similarly, Figure 9 shows reference tracking and produced control signal by DNN-based GPC controller while Figure 10 shows time evolution of parameters. Tracking performances of the controllers are compared in terms of RMSE and power of the produced control signal (P_u) in Table 2. It is noticed that in the first 40 seconds, smoother control signal is produced by ARX-based GPC controller. After the parameters of DNN-based GPC controller sets approximately to their stationary values, it begins performing as smooth tracking as ARX-based GPC does. This is the reason why ARX-based GPC has little smaller tracking RMSE value.

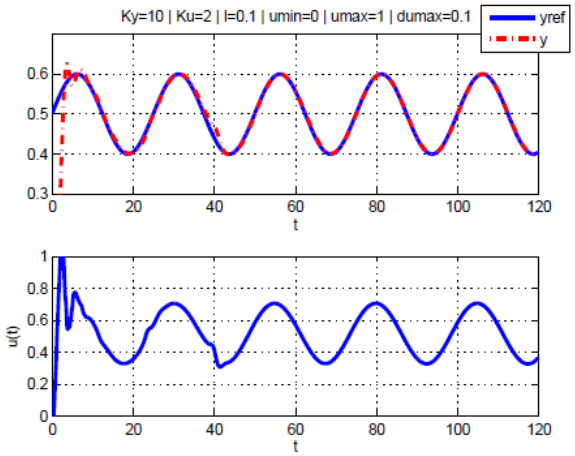

Figure 9: CSTR reference tracking and control signal produced by DNN-based GPC controller.

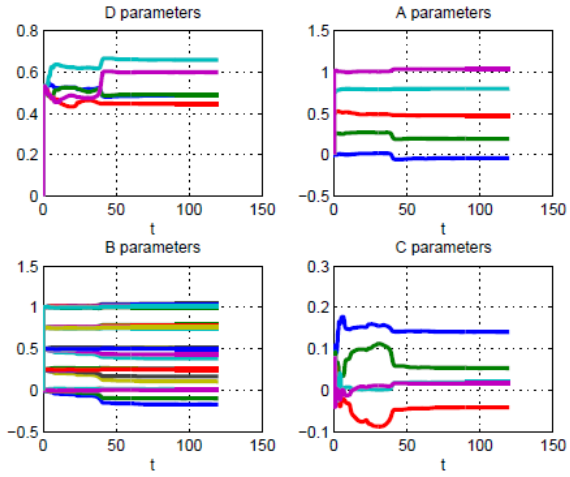

Figure 10: Time evolution of parameters in control of CSTR by DNN-based GPC controller.

Table 2: Comparison of tracking RMSE and P_u by ARX-based and DNN-based GPC controllers in CSTR system.

Controller	RMSE tracking	P_u
ARX-based GPC	**0.0505**	**0.2891**
DNN-based GPC	**0.0522**	**0.2924**

5. Conclusions

This study presents a comparison of ARX and DNN models in GPC scheme. Tracking performance of models were compared on bioreactor and CSTR systems which are both nonlinear benchmark systems. Even though ARX is a linear model, it has proven to be an acceptable model for generalized predictive control of nonlinear systems. Moreover, structural stability is maintained in both models by constraining the poles in a stable region. As a general result, ARX and DNN models are adopted successfully in GPC scheme for both systems.

Acknowledgment

This work is partly supported by Pamukkale University Scientific Research Projects (BAP) Department.

References

[1] K.J. Astrom. Theory and applications of adaptive control-a survey. Automatica, 19(5):471 – 486, 1983.

[2] Selami Beyhan and Musa Alc. Extended fuzzy function model with stable learning methods for online system identification. International Journal of Adaptive Control and Signal Processing, 25(2):168–182, 2011.

[3] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized predictive control part i. the basic algorithm. Automatica, 23(2):137 – 148, 1987.

[4] D W Clarke, C Mohtadi, and P S Tuffs. Generalized predictive control part ii. extensions and interpretations. Automatica, 23(2):149–160, March 1987.

[5] Kaynak O Efe M O, Abadoglu E. A novel analysis and design of a neural network assisted nonlinear controller for a bioreactor. International Journal of Robust and Nonlinear Control, 9:799–815, 1999.

[6] M. Ghiassi, H. Saidane, and D.K. Zimbra. A dynamic artificial neural network model for forecasting time series events. International Journal of Forecasting, 21(2):341 – 362, 2005.

[7] Sanqing Hu and Jun Wang. Global stability of a class of discrete-time recurrent neural networks. Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, 49(8):1104–1117, Aug 2002.

[8] Petros A. Ioannou and Jing Sun. Robust Adaptive Control. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[9] Serdar Iplikci. A support vector machine based control application to the experimental three-tank system. ISA Transactions, 49(3):376 – 386, 2010.

[10] Liang Jin, Peter N. Nikiforuk, and Madan M. Gupta. Absolute stability conditions for discrete-time recurrent neural networks. IEEE Transactions on Neural Networks, 5(6):954–964, 1994.

[11] Kyoung Kim, Jin Park, and Yoon Choi. The adaptive learning rates of extended kalman filter based training algorithm for wavelet neural networks. In Alexander Gelbukh and Carlos Reyes-Garcia, editors, MICAI 2006: Advances in Artificial Intelligence, volume 4293 of Lecture Notes in Computer Science, pages 327–337. Springer Berlin / Heidelberg, 2006.

[12] J. Korbicz, K. Patan, and A. Obuchowicz. Dynamic neural networks for process modelling in fault detection and isolation systems. International Journal of Applied Mathematics and Computer Science, Vol. 9, no 3:519–546, 1999.

[13] Grzegorz J. Kulawski and Mietek A. Brdy›. Stable adaptive control with recurrent networks. Automatica, 36(1):5 – 22, 2000.

[14] CHU Jian LI Li-Juan, SU Hong-Ye. Generalized predictive control with online least squares support vector machines. Acta Automatica Sinica, 33(11):1182, 2007.

[15] K. Patan. Stability analysis and the stabilization of a class of discrete-time dynamic neural networks. Neural Networks, IEEE Transactions on, 18(3):660 –673, May 2007.

[16] G. A. Rovithakis and M. A. Christodoulou. Adaptive control of unknown plants using dynamical neural networks. IEEE Transactions on Systems, Man, and Cybernetics, 24(3):400–412, Mar 1994.

[17] P.H. Srensen, M. Nrgaard, O. Ravn, and N.K. Poulsen. Implementation of neural network based non-linear predictive control. Neurocomputing, 28(13):37 – 51, 1999.

[18] Dan Simon. Training fuzzy systems with the extended kalman filter. Fuzzy Sets Syst., 132:189–199, December 2002.

[19] D. Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. Control Theory & Applications, IET, 4(8):1303–1318, August 2010.

[20] N. K. Sinha, M. M. Gupta, and D. H. Rao. Dynamic neural networks: an overview. In Industrial Technology 2000. Proceedings of IEEE International Conference on, volume 1, pages 491–496 vol.2, Jan 2000.

[21] Y. Song, Z. Chen, and Z. Yuan. New chaotic pso-based neural network predictive control for nonlinear process. IEEE Transactions on Neural Networks, 18(2):595–601, March 2007.

[22] Shiqian Wu and Meng Joo Er. Dynamic fuzzy neural networks-a novel approach to function approximation. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 30(2):358–364, Apr 2000.

[23] Wei Wu and Yi-Shyong Chou. Adaptive feedforward and feedback control of non-linear time-varying uncertain systems. International Journal of Control, 72(12):1127–1138, 1999.

[24] Sung Jin Yoo, Yoon Ho Choi, and Jin Bae Park. Generalized predictive control based on self-recurrent wavelet neural network for stable path tracking of mobile robots: adaptive learning rates approach. IEEE Transactions on Circuits and Systems I: Regular Papers, 53(6):1381–1394, June 2006.

[25] Wen Yu. State-space recurrent fuzzy neural networks for nonlinear system identification. Neural Processing Letters, 22(3):391–404, 2005.

