2014 13th International Conference on Control, Automation, Robotics & Vision
Marina Bay Sands, Singapore, 10-12th December 2014 (ICARCV 2014)



# An Adaptive Predictive Control based on a Quasi-ARX Neural Network Model

Mohammad Abu Jami'in (1,2); Imam Sutrisno (1,2); Jinglu Hu (1);
Norman Bin Mariun (3) and Mohd Hamiruce Marhaban (3)

(1) Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan
(2) Politeknik Perkapalan Negeri Surabaya, Surabaya, Indonesia
(3) Department of Electrical and Electronic Engineering, Universiti Putra Malaysia, Serdang, Malaysia
Email: mohammad@ruri.waseda.jp; imam@moegi.waseda.jp; jinglu@waseda.jp; norman@upm.edu.my; mhm@upm.edu.my

***Abstract*—A quasi-ARX (quasi-linear ARX) neural network (QARXNN) model is able to demonstrate its ability for identification and prediction highly nonlinear system. The model is simplified by a linear correlation between the input vector and its nonlinear coefficients. The coefficients are used to parameterize the input vector performed by an embedded system called as state dependent parameter estimation (SDPE), which is executed by multi layer parceptron neural network (MLPNN). SDPE consists of the linear and nonlinear parts. The controller law is derived via SDPE of the linear and nonlinear parts through switching mechanism. The dynamic tracking controller error is derived then the stability analysis of the closed-loop controller is performed based Lyapunov theorem. Linear based adaptive robust control and nonlinear based adaptive robust control is performed with the switching of the linear and nonlinear parts parameters based Lyapunov theorem to guarantee bounded and convergence error.**

## I. Introduction

If the dynamics model of the controlled system can be known exactly, then the ideal control can be calculated to obtain the desired reference trajectory . Generally, the system mixed with noise or nonlinear, so that the parameters of the systems are uncertain. The solution for those problems become difficult because the ideal control can not be obtained. To resolve those, a conventional linear robust control has been adopted, robustness and performance accuracy are the the terms that to be considered into controller design. However, increasing robustness can reduce the accuracy of the control system. To improve the control performances, nonlinear adaptive control based neural network (NN) model has been used as an identifier which is generally applicable to the systems with mathematically poorly modeled.

To facilitate the controller scheme of nonlinear systems, the technique of feedback linearization approach of nonlinear systems has been adopted in some research [1], [2], [3], [4], [5]. A nonlinear system can be transformed into a linear to the controller input with the use of feedback and coordinate transformation, then it is called feedback linearizable. Feedback linearization offers methods of building a stabilizing or a tracking controller for a nonlinear system by designing one for the equivalent linear system and utilizing the transformation along with its inverse [5]. With feedback linearization technique, a hybrid control system has been adopted to overcome the system uncertainties that is comprised of an ideal controller and a compensation controller. The ideal controller performed by NN with uncertainty observer is the principle controller, and the compensation controller is designed to compensate the minimum approximation error of the uncertainty observer [6], [4]. Hence, A robust training methodology is derived using the Lyapunov stability theorem with online learning ability [4].

Nonlinear system with the parameters always uncertain has more than one stability region, for those problems, with only nonlinear controller can't guarantee the bounded of the input-output closed-loop control [7]. To relax nonlinear control and to guarantee bounded controller, two controllers were used with switching mechanism [8], [7]. Thought, by using the linearization technique which states that the nonlinear system is linear to the input controller, then we can derive the control law [5]. With a different approach of linearization technique, we can state that a nonlinear system can be expressed as a linear relationship between the nonlinear coefficients and the input vector that called as a quasi-linear model. Thus the control law can be derived by a simple linear inverse via the nonlinear coefficients. We propose the controller strategy based on a quasi-linear model to control nonlinear system. Fortunately, the quasi-linear model has linear and nonlinear part parameters. It can be used to facilitate the controller approach based on the estimation of linear and nonlinear models.

A quasi-ARX neural networks (QARXNN) model is a nonlinear model. It can be simplified as a linear correlation between the input vector and its nonlinear coefficients. An embedded system is a sub-system that used to give a coefficient for each element of the input vector. The coefficients called as state dependent parameters estimation (SDPE) that consists of two parts: linear parameters and nonlinear parameters. The linear parameters estimator is performed by least square error (LSE) algorithm, which is set as bias vector for the output nodes of MLPNN [9], [10], [11]. In view of a nonlinear system is modeled under a quasi-linear autoregressive (quasi-ARX) model, nonlinear nature is placed on to the coefficients of the autoregressive (AR) or autoregressive moving average

978-1-4799-5199-4/14/$31.00 ©2014 IEEE

(ARMA). If the system is linear than SDPE will converge at the fixed value, whereas if the system is nonlinear then SDPE is a variable that will change at any time [12]. In this paper, an adaptive control based QARXNN prediction model is proposed to control of nonlinear systems. The controller is comprised of a linear robust adaptive controller (LRAC), a nonlinear robust adaptive controller (NRAC), and a switching mechanism. NRAC controller is designed based on SDPE that is the output of MLPNN, whereas LRAC is designed based on linear part parameter estimator (LPE) with the LSE algorithm.

## II. Quasi-ARX Neural Networks Model

Consider a single-input single-output (SISO) black-box time invariant whose the input-output relation described by

$$y(t) = g(\phi(t)) \tag{1}$$

where $g(\cdot)$ denotes a nonlinear function, $y(t) \in R$ denotes the system output and $t = 1, 2, \cdots$ denotes the sampling of time. By performing expansion in Taylor series [9], [13], nonlinear system can be presented as

$$y(t) = \phi^T(t)\aleph(\phi(t)) \tag{2}$$

where $\aleph(\phi(t)) = [a_{(1,t)} \cdots a_{(n_y,t)}\ b_{(1,t)} \cdots b_{(n_u,t)}]^T$ and $\phi(t) = [y(t-1) \cdots y(t-n_y)\ u(t-1) \cdots u(t-n_u)]^T$ are the coefficients of input vector and the input vector, respectively. $\phi(t) \in R^{n=n_u+n_y}$ where $n_u$ and $n_y$ denote the orders of time delay in the input and the orders of time delay in the output, respectively. An embedded system with MIMO MLPNN is used to model $\aleph(\phi(t)) \in R^{n=n_u+n_y}$ that called as SDP model is to parameterize the input vector.

The estimated output by a QARXNN model can be rewritten in the following form:

$$\hat{y}(t) = \phi^T(t)\aleph(\phi(t)) \tag{3}$$

if there exists $\aleph^*(\phi(t))$ such that

$$y(t) = \phi^T(t)\aleph^*(\phi(t)) \tag{4}$$

The adaptation error $\Xi$ is defined as

$$E(t) = \phi^T(t)\Xi \tag{5}$$

$$\Xi = \aleph(\phi(t)) - \aleph^*(\phi(t)) \tag{6}$$

where the adaptation error $E(t)$ can be stated as $y(t) - \hat{y}(t)$.

In our main theoretical result, the following assumption are made.

**A1.** The pairs of the input-output of training data are bounded.

**A2.** The coefficients of the regression vector $\aleph(\phi(t))$ are bounded.

**A3.** There exists optimal weights of the regression coefficient $\aleph^*(\phi(t))$

Suppose that the assumption **A2.** is valid, then there are positif real number $\alpha$ for $\| \aleph(\phi(t)) - \aleph^*(\phi(t)) \| < \alpha$. The regression vector of $\phi(t)$ has been fixed, by (5) and (6), the accuracy and convergence error characteristics for the QARXNN is determined by the adaptation error. Fig. 1 shows a QARXNN model with the MLPNN is used as an embedded system.

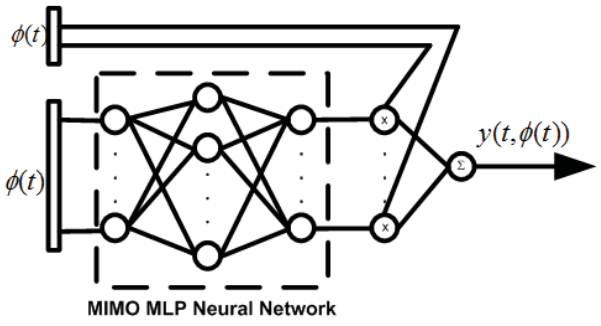


Fig. 1. A QARXNN with an embedded MIMO-MLPNN .

The coefficients of the input vector by QARXNN consists of linear parameters (LP) and nonlinear parameters (NP). The LP is executed using the least-squares error (LSE) algorithm, while the NP is performed by MLPNN with the estimated LP is used as bias vector of output nodes. The coefficients of the regression vector are the sum of LP and NP as follows:

$$\aleph(\phi(t), \Omega) = \delta(\phi(t)) + \theta \tag{7}$$

where $\Omega = \{W_1, W_2, \theta\}$ is the network parameters that consists of the weights of the hidden layer, the weights of output layer, and the bias vector of the output nodes. Nonlinear part executed by MLPNN is described by

$$\delta(\phi(t), W) = W_2 \Gamma W_1(\phi(t)). \tag{8}$$

where $\Gamma$ is the diagonal nonlinear operator with the identical sigmoidal elements on hidden nodes. $W = \{W_1, W_2\}$ denotes the weights space, where $W_1$ and $W_2$ are the weight matrix for the first layer and second layer, respectively.

Considering of LP and NP by QARXNN, we divide the model into two parts as follows:

$$SM1 \quad z_l = \phi^T(t)\theta \tag{9}$$

$$SM2 \quad z_n = \phi^T(t)\delta(\phi(t), W). \tag{10}$$

where $SM1$ and $SM2$ are linear sub-model and nonlinear sub-model, respectively. The adaptation error were distributed into two sub-models as follows:

$$\begin{aligned} E(t) &= y(t) - \hat{y}(t) \\ &= y(t) - \phi^T(t)\aleph(\phi(t), \Omega) \\ &= y(t) - \phi^T(t)(\delta(\phi(t)) + \theta) \\ &= y(t) - \phi^T(t)\delta(\phi(t)) - \phi^T(t)\theta \end{aligned} \tag{11}$$

$$E(t) = O_n - z_n \tag{12}$$

$$E(t) = O_l - z_l \tag{13}$$

where $O_l$ and $O_n$ are the adaptation error of linear sub-model and nonlinear sub-model, respectively. If linear parameters (LP) has been determined and set as bias vector, then the performances of the QARXNN will be greatly influenced by an embedded network of MLPNN, and vice versa. LP $\theta$ is calculated by LSE algorithm and is set as bias vector for MLPNN, then to optimize the performances of QARXNN

will be executed by nonlinear sub-model of MLPNN. The output using hierarchical learning for two sub-models can be expressed as

$$O_n = y(t) - \phi^T(t)\theta \quad (14)$$
$$O_l = y(t) - \phi^T(t)\delta(\phi(t), W). \quad (15)$$

The learning bases BP algorithm is described as follows:

Step 1.Calculate $\theta$ using LSE algorithm and small initial values of $W_1(k)$ and $W_2(k)$. Set $k = 1$, $k$ is the sequence of learning number.

Step 2.Calculate $z_n$ using sub-model SM2 in (10), calculate $O_n$ using (14), then the estimated linear parameters denoted as $\hat{\theta}$ is set as bias vector of output nodes MLPNN.

Step 3.Update $W_1(k), W_2(k)$ based on (16) that the update rules are performed by BP learning.

Step 4.Calculate network error using (12) and SDPE using (7).

Step 5.Stop if pre-specified condition is met, otherwise goto **Step 2.**, set $k = k + 1$

Usually, to update the network weights is done by minimizing the quadratic cost function as follows:

$$J = \frac{1}{2}\sum_{k=1}^{N}(O_n - z_n)^2. \quad (16)$$

The gradient descent (GD) algorithm [14] is used to update the network weights of MLPNN with respect to reducing errors.

## III. Control Strategy

A model in (2) is simplified and can be rewritten in the form of the relationship between the input vector and its coefficients as follows:

$$\begin{aligned} y(t) = {} & \hat{a}_{(1,t)}y(t-1) + \hat{a}_{(2,t)}y(t-2) + \\ & \hat{a}_{(n_y,t)}y(t-n_y) + \hat{b}_{(1,t)}u(t-1) + \\ & \hat{b}_{(2,t)}u(t-2) + \hat{b}_{(n_u,t)}u(t-n_u). \end{aligned} \quad (17)$$

where $\hat{\aleph}(\phi(t)) = [\hat{a}_{(1,t)} \cdots \hat{a}_{(n_y,t)}\hat{b}_{(1,t)} \cdots \hat{b}_{(n_u,t)}]^T$ is the coefficient of the input vector and $\phi(t) = [y(t-1)\, y(t-2) \cdots y(t-n_y)\, u(t-1)\, u(t-2) \cdots u(t-n_u)]^T$ is the input vector. We define the tracking error vector as follows:

$$\begin{aligned} \dot{y}_p(t) &= \dot{y}_p^d(t) + K^T E(t) \\ \dot{y}_p(t) - \dot{y}_p^d(t) &= -k_p\dot{e}_{p-1}(t) - k_{p-1}\dot{e}_{p-2}(t) - \cdots - k_1 e(t) \\ \dot{e}_p &= -k_p\dot{e}_{p-1}(t) - k_{p-1}\dot{e}_{p-2}(t) - \cdots - k_1 e(t) \\ 0 &= \dot{e}_p + k_p\dot{e}_{p-1}(t) + k_{p-1}\dot{e}_{p-2}(t) + \cdots + k_1 e(t) \end{aligned} \quad (18)$$

where $K = [k_p \cdots k_1] \in R^p$, $k_i (i = 1, \cdots, p)$ are positive constants, $y^d$ is the reference input trajectory, and $p$ is the degree of tracking error derivative.

To derive the control signal, model in (17) can be rewritten as

$$u(t-1) = \frac{1}{\hat{b}_{1,t}}(y(t) + g(t)) \quad (19)$$

$$\begin{aligned} g(t) = {} & -\hat{a}_{(1,t)}y(t-1) - \hat{a}_{(2,t)}y(t-2) \\ & -\hat{a}_{(n_y,t)}y(t-n_y) - \hat{b}_{(2,t)}u(t-2) \\ & -\hat{b}_{(n_u,t)}u(t-n_u)). \end{aligned} \quad (20)$$

If model in (2) simplified with (17) satisfies to mapping the input-output of the system, assumption **A1.**,**A2.** and **A3.** are fulfilled, then we can predict the output at time $(t + d)$. The equation (2) is regressed at time $(t + d)$ to calculate the output at $d$ step ahead prediction described as,

$$y(t+d) = \phi^T(t+d)\hat{\aleph}(\phi(t+d)) \quad (21)$$

where $\hat{\aleph}(\phi(t+d)) = [\hat{a}_{(1,t+d)} \cdots \hat{a}_{(n_y,t+d)}\hat{b}_{(1,t+d)} \cdots \hat{b}_{(n_u,t+d)}]^T$ is the coefficient of the input vector and $\phi(t+d) = [y(t+d-1)\, y(t+d-2) \cdots y(t+d-n_y)\, u(t+d-1)\, u(t+d-2) \cdots u(t+d-n_u)]^T$ is the input vector at $d$ step ahead prediction, for online step ahead prediction $d$ is equal to one. By (21), we have

$$u(t) = \frac{1}{\hat{b}_{1,t+1}}(y(t+1) + g(t+1)) \quad (22)$$

where $u(t)$ may correspond to networks output of $y(t)$ and $u^*(t)$ is desired control signal which correspond to $y^d$ denotes the desired control output. However, in cases of nonlinear systems, the parameters always uncertain that may cause unstable system.

To maintain the accuracy of the closed-loop control and stability, the switching line is used between the linear and nonlinear parameters. The parameters of QARXNN model can be divided by linear and nonlinear parts, so the proposed control law is defined as:

$$u(t) = \chi(t)u_n + (1 - \chi(t))u_l(t). \quad (23)$$

where $u_l$ is a linear robust control by using linear part parameters of $\hat{\theta}$ and $u_n$ is a nonlinear robust control by using nonlinear part parameters of $\hat{\aleph}(\cdot)$. $\chi(t) = 0(or1)$ is a switching line which $\chi(t) = 1$ denotes nonlinear robust control and $\chi(t) = 0$ denotes linear robust control. By (22) and (23), we define a controller estimation error as

$$\begin{aligned} u(t) - u^*(t) = {} & \frac{1}{\hat{b}_{1,t+1}}(y(t+1) + \hat{g}(t+1)) - \\ & \frac{1}{\hat{b}_{1,t+1}}(y^d(t+1) + g(t+1)) \\ U(t) = {} & \frac{1}{\hat{b}_{1,t+1}}(e(t+1) + G) \end{aligned} \quad (24)$$

where $U = u(\cdot) - u^*(\cdot)$,$G = \hat{g}(\cdot) - g(\cdot)$, $\hat{g}(\cdot)$ is calculated using the output of prediction model. The error tracking can be obtained

by

$$\begin{aligned} e(t+1) &= y(t+1) - y^d(t+1) = \hat{b}_{1,t+1}U(t) - G(t+1) \\ \dot{e}(t+1) &= e(t+1) - e(t) \\ &= \hat{b}_{1,t+1}U(t) - \hat{b}_{1,t}U(t-1) - G(t+1) + G(t) \\ \ddot{e}(t+1) &= \dot{e}(t+1) - \dot{e}(t) \\ &= \hat{b}_{1,t+1}U(t) - 2\hat{b}_{1,t}U(t-1) \\ &\quad +\hat{b}_{1,t-1}U(t-2) - G(t+1) + 2G(t) - G(t-1) \end{aligned} \tag{25}$$

Through (18) and (25), the dynamic tracking error can be stated as follows:

$$\dot{E} = AE + BU + G. \tag{26}$$

where

$$A = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \\ -k_p & -k_{p-1} & \cdots & -k_1 \end{pmatrix}$$

$$B = \begin{pmatrix} \hat{b}_{1,t+1} & 0 & 0 & 0 \\ \hat{b}_{1,t+1} & -\hat{b}_{1,t} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ c_1\hat{b}_{1,t+3-p} & -c_2\hat{b}_{1,t+2-p} & \cdots & (-1)^p c_{p+1}\hat{b}_{1,t+2-p} \end{pmatrix}$$

$$U = \begin{pmatrix} U(t) \\ U(t-1) \\ \cdots \\ U(t-p+1) \end{pmatrix}$$

$$G = \begin{pmatrix} -G(t+1) \\ -G(t+1) + G(t) \\ \vdots \\ -c_1 G(t+3-p) + \cdots + (-1)^{p+1} c_{p+1} G(t+2-p) \end{pmatrix}$$

where $A$ is a nonsingular matrix, $c_n$ are binomial series coefficients such as $\binom{p}{r} = \frac{p!}{r!\,(p-r)!}, 0 \le r \le p$.

By (18) and (26), we can calculate $K$ such that the roots of the characteristic of (26) can be chosen strictly that in the left half of the complex plane, then we have $\lim_{t\to\infty} e(t) = 0$. A minimum approximation control error is defined as follows:

$$\varepsilon = u^* - u(E|\aleph^*(\cdot)). \tag{27}$$

The controller objective is to maintain stability and accuracy of the closed-loop system by considering $\varepsilon$ such that

$$\aleph^*(\cdot) = \arg\min_{\aleph(\cdot)\in R} [\sup_{E\in R} |U|],$$

where $\aleph^*(\cdot)$ is an optimal network weight that achieves the minimum approximation error obtained through network learning. Assume that there are positif real number of ($|U| < \varepsilon$) if the system dynamic (26) are bounded. By introducing of $\varepsilon$, (26) can be rewritten as:

$$\dot{E} = AE + B(U(E|\aleph(\cdot)) - U(E|\aleph^*(\cdot)) - \varepsilon) + G. \tag{28}$$

Consider a Lyapunov function stated as

$$V(t) = \frac{1}{2}E^T PE \tag{29}$$

where $P$ is a symmetric positive definite matrix. Since $V(t)$ was selected to be positive definite, to make uniformly stable, we need that $\dot{V}(t)$ has to be negative semidefinite. Therefore, we require $\dot{V}(t) = -\dot{E}^T QE$ to be a negative semidefinite that implies $V(t) \le V(0)$, a negative semidefinite matrix $Q$ is stated as

$$Q = -(A^T P + PA) \tag{30}$$

**Theorem 1:** Suppose a dynamic tracking error is described by

$$\dot{E} = f(E,t) \tag{31}$$

where $f(0,t) = 0$ for all $t$. If there exists scalar function $V(E,t)$ having continuous first partial derivative and satisfying the condition:

1) $V(E,t)$ is a positive definite
2) $\dot{V}(E,t)$ is a negative semidefinite,

then the equilibrium state at the origin is uniformly stable.

The system of (28), an equilibrium state $E_e$ is defined as $f(E,t) = 0, \forall t$. For nonlinear systems, there are one or more $E_e$. We denote a spherical region of radius $r$ about an equilibrium state as $\| E - E_e \| \le r$ that is called as the Euclidean norm defined by

$$\| E - E_e \| = ((E_1 - E_{1e})^2 + \cdots + (E_p - E_{pe})^2)^{\frac{1}{2}}. \tag{32}$$

Let $S(\gamma)$ consist of all point such that $\| E - E_e \| \le \gamma$ where $\gamma \ge \varepsilon$. The time derivative of Lyapunov function along any trajectory is

$$\begin{aligned} \dot{V}(t) &= \frac{1}{2}\dot{E}^T PE + \frac{1}{2}E^T P\dot{E}. \\ &= \frac{1}{2}(AE + B(U(E|\aleph(\cdot)) - U(E|\aleph^*(\cdot)) - \varepsilon) + G)^T PE + \\ &\quad \frac{1}{2}E^T P(AE + B(U(E|\aleph(\cdot)) - U(E|\aleph^*(\cdot)) - \varepsilon) + G) \\ &= \frac{1}{2}(E^T A^T PE + E^T PAE) + \frac{1}{2}(B(\tilde{U} - \varepsilon) + G)^T PE + \\ &\quad \frac{1}{2}E^T P(B(\tilde{U} - \varepsilon) + G) \\ &= -\frac{1}{2}(E^T QE) + \frac{1}{2}((B(\tilde{U} - \varepsilon) + G)^T PE \\ &\quad +E^T P(B(\tilde{U} - \varepsilon) + G)) \\ &= -\frac{1}{2}(E^T QE) + (B(\tilde{U} - \varepsilon) + G)^T PE \\ &= -\frac{1}{2}(E^T QE) + (\tilde{U} - \varepsilon)^T B^T PE + G^T PE \end{aligned} \tag{33}$$

where $\tilde{U} = U(E|\aleph(\cdot)) - U(E|\aleph^*(\cdot))$.

**Theorem 2:** Using prediction model of (2) and the controller law is (23). By a positive constant $\varepsilon$ such that

$$\rho \le -\frac{1}{2}(E^T QE) + (\tilde{U} - \varepsilon)^T B^T PE + G^T PE, \rho \le 0 \tag{34}$$

then $\lim_{t\to\infty} E(t) = 0$, $E(t) \to 0$ at $t \to \infty$, the tracking error $e$ will converge to zero.

According **Theorem 2:**, we propose a switching line to change control action of linear part controller and nonlinear

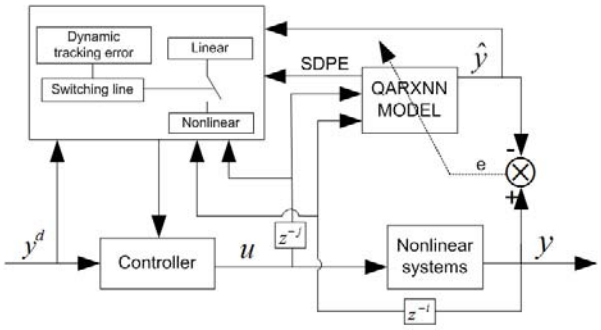


Fig. 2. Nonlinear adaptive predictive controller based on QARXNN prediction model. $i = 1, \cdots, n_y, j = 1, \cdots, n_u$

part controller. The model only with linear parameters has to work until the use of nonlinear parameters does not damage the stability of closed loop system. Therefore, the controller with using linear parameters $\hat{\theta}$ will work all the time, but the nonlinear parameters $\hat{\aleph}(\phi(t))$ will work under the switching sequence. The controller law (23) work under the switching line as follows:

$$\chi(t) = \begin{cases} 1, & \text{if } \rho \le 0 \\ 0, & \text{otherwise} \end{cases} \tag{35}$$

In the following, the design algorithm of the proposed control law is summarized as follows.

Step 1.Identify the system under QARXNN model online described in Section 2.

Step 2.Find the estimated parameter of SDPE by simplified QARXNN prediction model

Step 3.By using SDPE, calculate dynamics tracking error shown by dynamic matrix of $A$ in (26). By introducing $\varepsilon$, find a new state of dynamics tracking error in (28) to obtain stability region with specific $\varepsilon$, where $\varepsilon \le \gamma$.

Step 4.Check the stability of NRAC controller by satisfying (34) and switching line of (35).

Step 5.Calculate controller signal using (22), two controller can be obtained by using the linear and nonlinear parts parameters of SDPE via switching mechanism in (23),(34),and (35).

Step 6.Goto **Step 1.**.

## IV. Simulation Result

In this section, the experiment and simulation are performed to confirm the performance of the proposed method. In the following simulations, all systems are modeled under QARXNN model online.

**Example:** Considering a discrete time nonlinear systems with unstable zero-dynamics is described as follows [15], [8]:

$$\begin{aligned} y(t) &= 2.6y(t-1) - 1.2y(t-2) + u(t-1) \\ &+ 1.5u(t-2) + 0.5y(t-1)\mathtt{sin}(u(t-1) \\ &+ u(t-2) + y(t-1) + y(t-2)) \end{aligned} \tag{36}$$

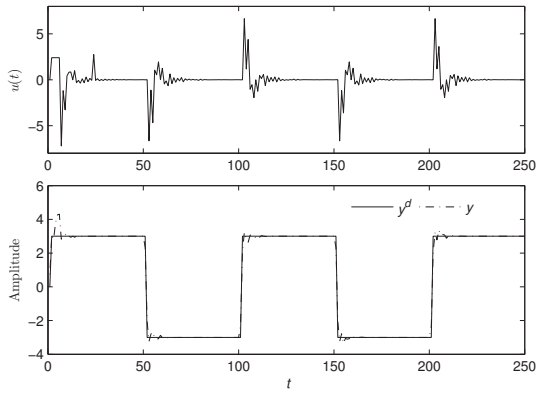


Fig. 3. Control signals and output responses

The objective is to make the system output $y(t)$ tracks a input reference (desired output trajectory) $y^d(t)$ specified by

$$y^d(t) = 3\mathtt{sign}(\mathtt{sin}(\pi t/50)), \quad 0 < t \le 250. \tag{37}$$

From the system model (36), by using the QARXNN model, An embedded system of MLPNN is constructed by three layer neural network. The input vector of $\phi(t)$ is specified by $\phi(t) = [y(t-1)\,y(t-2)\,u(t-1)\,u(t-2)]^T$ with $n = 4$ equal to the sum of $n_u = 2$ and $n_y = 2$. The number of input nodes, hidden nodes, and output nodes is also the same as $n$. Constant learning rate of BP algorithm is selected by $\eta_{bp} = 0.1$ and gain of adaptive tracking control based QARXNN model are given by: $\gamma = 0.02$, $p = 2$, $Q = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$. Fig. 3 shows the performance of the closed-loop switching control and Fig. 4 shows the error tracking and rooted mean square (RMS) error stated as

$$RMS = \sqrt{\frac{\sum_{t=1}^{N}(y(t) - y^d(t))^2}{N}} \tag{38}$$

where $y^d(t)$, $y(t)$, $t = 1, 2, \cdots, N$ are desired output used as control reference, the output of controlled system, time sampling, and $N$ is the length of the input-output controlled system. Fig. 5 shows the switching sequence, where $\chi(t) = 1$ denotes the use of NRAC, and $\chi(t) = 0$ denotes the LRAC.

## V. Discussion and Conclusion

This paper presents an adaptive controller based prediction model under quasi-ARX neural network (QARXNN). This study has successfully demonstrated the effectiveness of the proposed controller. The tracking error of closed-loop system is derived via SDPE, the stability analysis of closed-loop system is performed using Lyapunov stability theory. The switching line between linear and nonlinear parts of SDPE is conducted to guarantee boundedness and convergence of error. Finally, the control performance using the proposed algorithm has been confirmed. The main contributions of this study are: (1) the successful development of a quasi-ARX neural network based adaptive control. By using QARXNN model, we have advantages, that the controller law and switching line

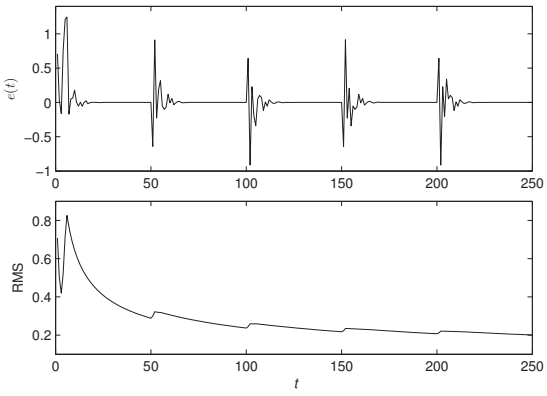


Fig. 4. Tracking error and RMS error

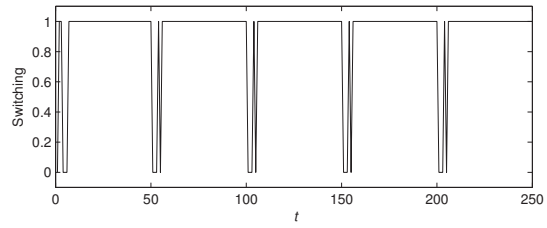


Fig. 5. Switching sequence (0: linear; 1: nonlinear)

are performed using simplified parameter that called SDPE; (2) the successful development of adaptive control based switching line to guarantee stability of closed-loop systems. The switching line is constructed via SDPE that more easy to be used.

## VI. Acknowledgments

The first author would like to thank to Indonesian Government Scholarship Directorate General of Higher Education (DGHE) (Beasiswa Luar Negeri DIKTI - Kementerian Pendidikan dan Kebudayaan Republik Indonesia) for the scholarship.

## References

[1] F. J. Lin, R. J. Wai, and H. P. Chen, "A hybrid computed torque controller using fuzzy neural network for motor-quick-return servo mechanism," *IEEE/ASME Trans. on Mechatronics*, vol. 6(1), pp. 75–89, 2001.

[2] C.-L. Hwang and C.-Y. Kuo, "A stable adaptive fuzzy sliding-mode control for affine nonlinear systems with application to four-bar linkage systems," *IEEE Trans. on Fuzzy Systems*, vol. 9(2), pp. 868–879, 2001.

[3] C. M. Lin and C. F. Hsu, "Neural-network-based adaptive control for induction servomotor drive system," *IEEE Trans. Ind. Electron.*, vol. 49(1), pp. 238–252, 2002.

[4] R.-J. Wai, "Hybrid fuzzy neural-network control for nonlinear motor-toggle servomechanism," *IEEE Trans. on Control Systems Tech.*, vol. 10(4), pp. 519–532, 2002.

[5] K. Nam, "Stabilization of feedback linearizable systems using a radial basis function network," *IEEE Trans. on Automatic Control*, vol. 44(5), pp. 1026–1031, 1999.

[6] C.-M. Lin and C.-F. Hsu, "Neural-network hybrid control for antilock braking systems," *IEEE Trans. on Neural Networks*, vol. 14(2), pp. 351–359, 2003.

[7] T. Chai, Y. Zhang, H. Wang, C. Y. Su, and J. Sun, "Data-based virtual unmodeled dynamics driven multivariable nonlinear adaptive switching control," *IEEE Transactions on Neural Networks*, vol. 22(12), pp. 2154–2172, 2011.

[8] Y. Zhang, T. Chai, H. Wang, J. Fu, L. Zhang, and Y. Wang, "An adaptive generalized predictive control method for nonlinear systems based on ANFIS and multiple models," *IEEE Trans. on Fuzzy Systems*, vol. 18(6), pp. 1070–1081, 2010.

[9] J. Hu, K. Kumamaru, and K. Hirasawa, "A Quasi-ARMAX approach to modelling of non-linear systems," *Int. J. Control*, vol. 74(18), pp. 1754–1766, 2001.

[10] Y. Cheng, L. Wang, and J. Hu, "Quasi-ARX wavelet network for SVR based nonlinear system identification," *Nonlinear Theory and its Applications (NOLTA), IEICE*, vol. 2(2), pp. 165–179, 2011.

[11] M. A. Jami'in, I. Sutrisno, and J. Hu, "Lyapunov learning algorithm for quasi-ARX neural network to identification of nonlinear dynamical system," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics (Seoul)*, 2012, pp. 3141–3146.

[12] M. A. Jami'in, I. Sutrisno, and J. Hu, "Deep searching for parameter estimation of the linear time invariant (LTI) system by using quasi-ARX neural network," in *Proc. IEEE International Joint Conference on Neural Network (Dallas)*, 2013, pp. 2759–2762.

[13] J. Hu and K. Hirasawa, "A method for applying multilayer perceptrons to control of nonlinear systems," in *Proc. 9th International Conference on Neural Informassion Processing (Singapure)*, 2002.

[14] S. Haykin, *Neural Networks, A Comprehensive Foundation*. Prentice-Hall, 1999.

[15] L. Chen and K. S. Narendra, "Nonlinear adaptive control using neural networks and multiple models," *Automatica*, vol. 37, pp. 1245–1255, 2001.