Applied Intelligence
https://doi.org/10.1007/s10489-019-01615-0

Hierarchical linear and nonlinear adaptive learning model for system identification and prediction

Mohammad Abu Jami'in[1] · Khairul Anam[2] · Riries Rulaningtyas[3] · Urip Mudjiono[1] · Adianto Adianto[1] · Hui-Ming Wee[4]

Abstract

In this paper, we propose a method to increase the model accuracy with linear and nonlinear sub-models. The linear sub-model applies the least square error (LSE) algorithm and the nonlinear sub-model uses neural networks (NN). The two sub-models are updated hierarchically using the Lyapunov function. The proposed method has two advantages: 1) The neural networks is a multi-parametric model. Using the proposed model, the weights of NN model can be summarized into the coefficients or parameters of auto-regressive eXogenous/auto-regressive moving average (ARX/ARMA) model structure, making it easier to establish control laws, 2) learning rate is updated to ensure the convergence of errors at each training epoch. One can improve the accuracy of model and the whole control system. We have demonstrated by the experimental studies that the proposed technique gives better results when compared to the existing studies.

Keywords System identification · Hierarchical algorithm · Adaptive learning · Prediction · Parameter estimation

1 Introduction

Many researchers have proposed various techniques in order to estimate the parameters of ARX/ARMA model. A linear regression model in least square error (LSE) or recursive least square (RLS) estimation was proposed [9, 15, 19, 22, 25]. Both accuracy and convergence of time are the main issues to investigate. The convergence of RLS estimation is a function over time. Convergence analysis is solved by differential equations that require all information about the asymptotic behavior of the system [19]. In the deterministic cases, the convergence behavior of RLS tends to the actual parameters when the data of information vector go to infinity [22].

To improve the LSE or RLS performance, some researchers propose several methods such as recursive algorithm of multi variable auto-regressive moving average (ARMA) system [20], dichotomous coordinate descent recursive least square (DCDRLS) (algorithm [1], a technique of refined instrumental variable for continuous systems (RIVC) [18], a portable alternating least squares algorithm applied for factorization of parallel matrix [6]. The methods were developed by linear regression model and its modification. It is difficult to capture the convergence behavior of system that may have a nonlinear property.

Artificial neural network model has attracted the attention of many researchers. The neural network (NN) model is able to map input-output of system very well through training [16]. However, most of NN is black-box (non-parametric) or multi-parametric models that are difficult to be used in analyzing the system modeling such as the properties of linear structure [10, 14]. Using linear structure, the prediction model becomes friendly user and

✉ Mohammad Abu Jami'in
jammysby@gmail.com

Khairul Anam
khairul@unej.ac.id

Riries Rulaningtyas
riries-r@fst.unair.ac.id

Hui-Ming Wee
weehm@cycu.edu.tw

[1] Politeknik Perkapalan Negeri Surabaya, Jalan Teknik Kimia Kampus ITS Sukolilo Surabaya, 60111, Surabaya, Indonesia

[2] Jember University (UNEJ), Jember, Indonesia

[3] Airlangga University (UNAIR), Surabaya, Indonesia

[4] Chung Yuan Christian University (CYCU), Taoyuan City, Taiwan

Published online: 10 February 2020

is suitable for nonlinear control applications [13]. Some researchers have developed NN for time series model with ARX/ARMA structure modeling [3, 4, 28]. The analysis of the NN model focuses on the ability of the model to improve mapping accuracy. Discussion about estimating parameters is not included. However, the model parameter of autoregressive eXogenous (ARX) structure has some advantages to estimate the system stability and to derive the controller law [13].

For the designing of ARX/ARMA structure under NN model, the technique for finding parameters in the autoregressive model has been greatly developed. Hu developed quasi-linear ARX neural networks (QARXNN) models [10, 27], Peng working on RBF-ARX networks [23], Jami'in developed Hybrid hierarchical learning algorithm of QARXNN model [14]. They developed NN networks to provide parameters on ARX or ARMA models for nonlinear systems. Related to the development of autoregressive model, other researcher developed nonlinear autoregressive with eXogenous Inputs (NARX) network [5] and hybrid model with combine both linear model and nonlinear under neural networks model [24]. However, the identified parameters with the structure of NARX model is still black-box or multi-parametric. It is difficult to estimate the parameter of linear-like system using NARX. Hybrid model in [24], did not discuss about the parameters of ARX structure modeling.

The problems we are going to solve in this research are: 1) to increase the accuracy of the ARX/ARMA-like model structure, 2) to increase the convergence speed of the estimated parameters. The novelty of our works is to develop an adaptive learning method for the linear and nonlinear sub-models which are trained hierarchically. The LSE algorithm is combined with NN model to estimate the parameters of prediction models. The contributions of this research are 1) The estimated parameters using LSE algorithm are improved through updating the NN network weights. The estimated parameter of LSE algorithm converges to more accurate parameters when the data of regression vector goes to infinity. It is improved by increasing the number of NN training. Thus, the estimated parameters can convergence faster. Moreover, the proposed technique can capture the nonlinear dynamics of a system as well. 2) The traditional back-propagation algorithm (BP Learning) on NN model uses a constant learning rate and may result in local minima. In Lyapunov learning algorithm, the learning rate is updated during each training to ensure the convergence of errors in each epoch, thus one can achieve global minima. Therefore, the proposed hierarchical adaptive learning of linear and nonlinear sub-models of quasi linear-ARX neural networks (HAL-QARXNN) can improve the accuracy of the estimated parameters. 3) The proposed method can be applied to identify the hydraulic robot actuator and to design a control system to maximize the power tracking controller of wind energy conversion systems (WECS).

Compared to the LSE algorithm, the proposed technique or algorithm is more complex and speed has become critical in the calculation process. The research is limited by the availability of high speed processing unit. Another method for increasing the accuracy and speed of training is the Extreme Learning Machines (ELMs) algorithm. The ELMs algorithm is focused on the hidden network where the parameters of the hidden network are generated randomly. Some methods for finding optimal parameters of hidden networks are linear discrimination analysis (LDA), neighborhood components analysis (NCA), locality sensitive discriminant analysis (LSDA), isometric projection (IsoP), maximum margin projection (MPP), neighborhood preserving embedding (NPE), principal component analysis (PCA), linear graph embedding (LGE), unsupervised locality preserving projection (LPP) and its modifications [30]. For the future works, we will test the ELMs algorithm for the training of nonlinear part of hierarchical training of QARXNN model.

In the identification scheme, we term a surface submodel in the first step. It is a linear regression model used to estimate the parameters under LSE algorithm. At the first step, we assume that the system is linear. From the first step we have linear parameter which is used to calculate residual error of surface sub-model. The residual error of surface sub-model is modeled using nonlinear neural network multilayer perceptron model injected in QARXNN termed as bottom sub-model performed in second step. Bottom sub-model is nonlinear sub-model and surface sub-model is linear sub-model. Because the structure of bottom sub-model is parallel to the surface sub-model then the parameter of surface sub-model can be updated using nonlinear bottom sub-model at every training epoch. Through training, the parameters of surface sub-models are updated and the residual errors of surface sub-model are reduced by using nonlinear sub-models. Thus, we can 1) improve the mapping accuracy by reducing residual error of LSE algorithm, 2) by adjusting the learning rate, the prediction model is ensured to reach global minima. Finally, we show that the proposed method gives better results through experimental studies.

2 Linear time invariant

Let us consider a discrete time linear system presented by:

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{j=1}^{m} b_j u(k-j) + \omega(k) \qquad (1)$$

where a_i and b_j are the parameters. The input and output of system are denoted by $u(k)$, $y(k)$. The variable of $\omega(k)$ denotes a stochastic white noise with zero mean and variance σ_ω^2. The orders of system are denoted by n and m.

Rewritten in matrix equation, the system (1) will be:

$$y(k) = \phi^T(k)\theta + \omega(k) \tag{2}$$

where,

$$\begin{aligned} \theta &= [a_1, a_2, ..., a_n, b_1, b_2, ..., b_m]^T \in R^{n+m} \\ \phi(k) &= [-y(k-1)... - y(k-n)\, u(k-1)...u(k-m)]^T. \end{aligned}$$

The parameter of system and the regression vector are denoted by θ and $\phi(k)$, respectively.

3 Quasi-ARX neural network model (QARXNN)

Quasi linear-ARX Neural Networks (QARXNN) model is divided into linear and nonlinear sub-models. The linear sub-model is a macro-part, while nonlinear sub-model is a core part. The core-part sub-model is to provide nonlinear coefficients for the input regression vector performed by multi input and output neural networks. QARXNN is a model for nonlinear system where NN is used to give nonlinear coefficients for the input vectors. Through performing Taylor expansion series, a nonlinear system can be derived as a linear correlation between the input vector and its coefficients expressed as [10, 12, 29]:

$$y(k) = \phi^T(k)\aleph(\xi(k)). \tag{3}$$

where $\aleph(\xi(k)) = [a_{(1,k)} \cdots a_{(n_k,t)}\ b_{(1,k)} \cdots b_{(n_u,k)}]^T$ is a nonlinear function of core part sub-model to parameterize the regression vector. $\xi(k) = [y(k-1) \cdots y(k-n_y)\, u(k-2) \cdots u(k-n_u)\ v(k)]^T$ is the input for core-part sub-model in which virtual input $v(k)$ is added. The output of the nonlinear or core-part sub-model is the nonlinear coefficients of the regression vector and the input is the regression vector itself. The number of input elements of the regression vector is equal to the number of output elements, namely the order system (n+m). Thus, we need a multi-input multi-output model. An appropriate model by using NN is a multi-layer perceptron neural network. The MIMO neural network is selected as core-part sub-model stated as:

$$\begin{aligned} \aleph(\xi(k), \Omega) &= W_2 \Gamma W_1(\xi(k)) + \theta \quad (4) \\ &= \delta(\xi(k)) + \theta \quad (5) \end{aligned}$$

where $\Omega = \{W_1, W_2, \theta\}$ is the network's weights, Γ is an operator of sigmoidal element for hidden nodes.

4 Hierarchical training

Incorporating to hierarchical training, the output of dynamic system is presented as:

$$y(k) = y_{LS}(k) + e_{LS}(k). \tag{6}$$

The linear sub-model is performed by using LS algorithm which is designated as surface sub-model described as:

$$\hat{y}_{LS}(k) = \phi^T(k)\hat{\theta}_{LS}(N). \tag{7}$$

The residual error of LS estimation is modeled under NN which is termed as bottom sub-model stated as:

$$\begin{aligned} e_{LS}(k) &= \phi^T(k)\Delta\theta(\phi(k)). \\ &= \phi^T(k)W_2\Gamma W_1((\phi(k)). \end{aligned} \tag{8}$$

The hierarchical training process for the updating of the estimated parameters is shown in Fig. 1. In the first step, we perform LSE algorithm to estimate the parameters denoted by $\theta_{LS}(N)$ where N is the number of input-output data. The residual error of LSE e_{LS} is set as the output for MIMO-NN injected to QARXNN model for estimating $\Delta\theta$. The model of QARXNN is performed to refine residual parameters by mapping of residual error in second step shown in Fig. 2. Finally, the estimated parameter is update by summing $\theta_{LS}(N)$ and $\Delta\theta$ displayed in (18).

In the first step, the system is identified by using surface sub-model and completed by LS algorithm. The linear parameter denoted by $\hat{\theta}_{LS}(N)$ is estimated by minimizing cost function stated as:

$$J_N(\theta) = \sum_{k=1}^{N}(y(k) - \phi^T(k)\theta)^2. \tag{9}$$

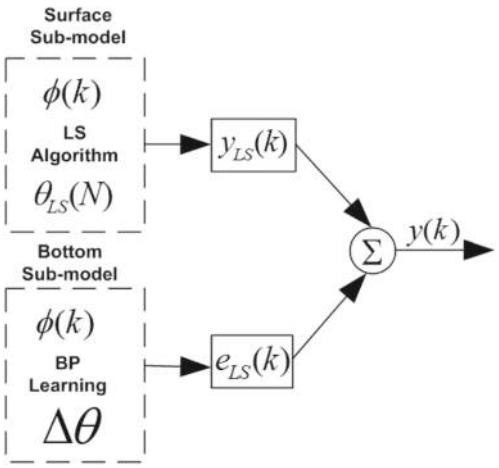

Fig. 1 Hierarchical Training of linear and nonlinear sub-models

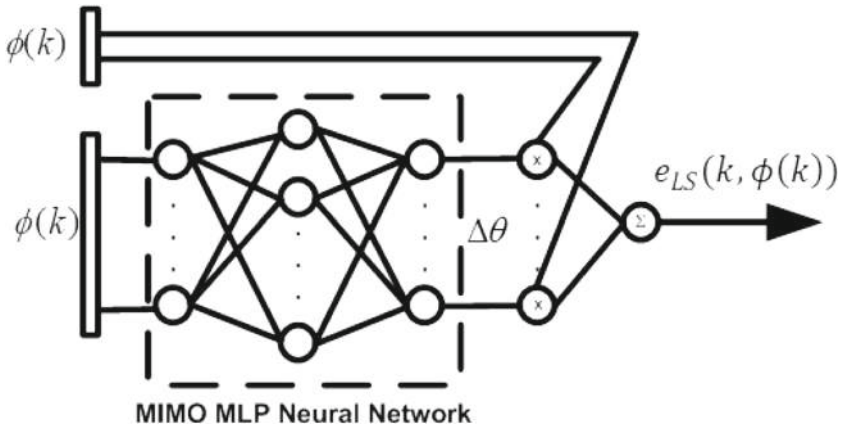

Fig. 2 Bottom sub-model incorporating to QARXNN

The estimated parameters subject to minimize (9) analytically leads to [8]:

$$\hat{\theta}_{LS}(N) = \left[\sum_{k=1}^{N} \phi^T(k)\phi(k)\right]^{-1} \left[\sum_{k=1}^{N} \phi^T(k)y(k)\right] \tag{10}$$

where,

$$\hat{\theta}_{LS}(N) = [\hat{a}_1, \hat{a}_2, ..., \hat{a}_n, \hat{b}_1, \hat{b}_2, ..., \hat{b}_m]^T \in R^{n+m} \tag{11}$$

Through (6), the residual error of LSE estimation is rewritten as:

$$e_{LS}(k) = y(k) - y_{LS}(k) \tag{12}$$

By substituting (7) to (12), we have (13) called as a bottom sub-model. It will be implemented under MIMO-NN of QARXNN model shown in Fig. 2:

$$\begin{aligned} e_{LS}(k) &= \phi^T(k)\theta - \phi^T(k)\hat{\theta}_{LS}(N) \\ &= \phi^T(k)(\theta - \hat{\theta}_{LS}(N)) \\ &= \phi^T(k)\Delta\theta(\phi(k)) \end{aligned} \tag{13}$$

The $\Delta\theta^T(\phi(k))$ is a residual parameters which is the output of bottom sub-model performed using NN. Hence, the update of the estimated parameters $\hat{\theta}(k)$ will be:

$$\hat{\theta}(k) = \hat{\theta}_{LS}(N) + \Delta\theta(\phi(k)) \tag{14}$$

$\Delta\theta(\phi(k)) = [\Delta\theta_{a_1}, \Delta\theta_{a_2}, \cdots, \Delta\theta_{a_n}, \Delta\theta_{b_1}, \Delta\theta_{b_2}, \cdots, \Delta\theta_{b_m}]^T$. The estimated output of system will be:

$$\hat{y}(k) = \phi^T(k)\hat{\theta}(k) \tag{15}$$

In difference (7) and (15) are rewritten as follows:

$$\begin{aligned} \hat{y}_{LS}(k) = &-\hat{a}_1 y(k-1) - \hat{a}_2 y(k-2) - ... - \hat{a}_n y(k-n) \\ &+\hat{b}_1 u(k-1) + \hat{b}_2 u(k-2) + ... + \hat{b}_m u(k-m) \end{aligned} \tag{16}$$

and

$$\begin{aligned} \hat{y}(k) = &-(\hat{a}_1 + \Delta\theta_{a_1})y(k-1) \\ &-(\hat{a}_2 + \Delta\theta_{a_2})y(k-2) - ... \\ &-(\hat{a}_n + \Delta\theta_{a_n})y(k-n) \\ &+(\hat{b}_1 + \Delta\theta_{b_1})u(k-1) \\ &+(\hat{b}_2 + \Delta\theta_{b_2})u(k-2) \\ &+(\hat{b}_m + \Delta\theta_{b_m})u(k-m). \end{aligned} \tag{17}$$

The update of estimated parameters using hierarchical algorithm will be:

$$\hat{\theta}(k) = \hat{\theta}_{LS}(N) + \sum_{L=1}^{N_L} \Delta\theta_L(\phi(k)) \tag{18}$$

where, L is the learning sequence (epoch) and N_L is learning number. The error of the proposed method is the error of LS algorithm corrected by the output of bottom sub-model as follows:

$$e(k) = e_{LS}(N) - \sum_{L=1}^{N_L} \phi^T(k)\Delta\theta_L(\phi(k)) \tag{19}$$

based on (19), we can see that the error of LS estimation in surface sub-model can be improved by using bottom sub-model performed under NN. The error of hierarchical algorithm by combining the linear and nonlinear sub-models in (19) is shown in Fig. 3

The LS algorithm will converge to the real parameters if the sampling data measurement tends to infinity resulted in the longer memory of the past data of the information vector. In this case, the convergence of LS algorithm is slow. We propose hierarchical training to improve the convergence of LS estimation. The estimated parameters $\hat{\theta}(k)$ is the summing between the parameter of surface sub-model under LS algorithm $\hat{\theta}_{LS}(N)$ and the bottom sub-model performed by neural network $\Delta\theta_L(\phi(k))$. The output of hierarchical learning will be:

$$\hat{y}(k) = \phi^T(k)\hat{\theta}_{LS}(N) + \phi^T(k)\Delta\theta_L(\phi(k)). \tag{20}$$

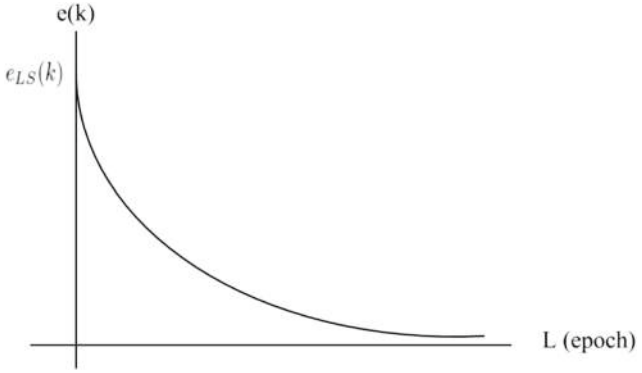

Fig. 3 The correction of LS error by training number

We can see on (20) that the estimated output of LS estimation is corrected by the training of NN.

The residual error of LS estimation is modeled using NN in the second step, where the residual parameters is stated as:

$$\Delta\theta(\phi(k)) = W_2 \Gamma W_1((\phi(k)). \tag{21}$$

Hence, the bottom sub-model is expressed as:

$$\begin{aligned} y_b(k) &= \phi^T(k)\Delta\theta(\phi(k)). \\ &= \phi^T(k) W_2 \Gamma W_1((\phi(k)). \end{aligned} \tag{22}$$

The set of network parameters is denoted by $W = W_1, W_2 \in R^{(n+m)\times(n+m)}$ that is the weight matrix at the the first and second layer. Incorporating to NN, the error of system modeling (19) can be rewritten as:

$$\begin{aligned} e(k) &= e_{LS}(N) - y_b(k) \\ &= e_{LS}(N) - \phi^T(k) W_2 \Gamma W_1((\phi(k)) \end{aligned} \tag{23}$$

5 Adaptive learning

In order to train the bottom sub-model, we consider a Lyapunov-function based learning. The selected Lyapunov function is generally a quadratic cost function that is a positive definite or positive semi definite function presented as:

$$V = \frac{1}{2} e^T e \tag{24}$$

e is the error of modeling system presented in (23). Based on the Lyapunov theorem, the convergence condition can be achieved if a selected Lyapunov function is a positive definite or positive semi definite function, and the derivative of Lyapunov function is a negative semi definite function. In order to ensure the convergence of the training of NN, we select the trajectory convergence based on Lyapunov function as follows:

$$\dot{V} = -\|e\|^2 \tag{25}$$

The time derivative of Lyapunov function of (24) will be:

$$\begin{aligned} \dot{V} &= \frac{1}{2}\dot{e}^T e + \frac{1}{2} e^T \dot{e} \\ &= e^T \dot{e} \end{aligned} \tag{26}$$

we define $\frac{\partial y_b}{\partial w} = H$, then the weights update will be:

$$\begin{aligned} -\|e\|^2 &= -e^T \frac{\partial y_b}{\partial w} \dot{W} \\ \dot{W} &= \frac{\|e\|^2}{H e^T} \end{aligned} \tag{27}$$

Written in difference equation that $\dot{W} = W(k+1) - W(k)$, thus the update of the network's weight will be:

$$\begin{aligned} W(k+1) &= W(k) + \frac{\|e\|^2}{H e^T} \\ W(k+1) &= W(k) + \frac{\|e\|^2}{\|\frac{\partial y_b}{\partial w}^T e\|^2} \frac{\partial y_b}{\partial w}^T e. \end{aligned} \tag{28}$$

The structure of the update law of the proposed algorithm is presented in (28) that the learning rate η is stated as $\frac{\|e\|^2}{\|\frac{\partial y_b}{\partial w}^T e\|^2}$. Thus, the learning rate is changed in every epoch. The hierarchical adaptive training will be greatly influenced by the training performance. The LSE estimation will be improved under the proposed technique in which the time convergence is corrected by the number of training.

6 Learning steps

We divide the system modelling into two sub-models; surface and bottom sub-models. The former is performed using LSE algorithm and the latter is carried out by using MIMO-NN. The target output of surface sub-model is calculated by $s(k) = y(k) - e_{LS}$ and that of the bottom sub-model is by $b(k) = y(k) - y_{LS}(k)$. The target output for the

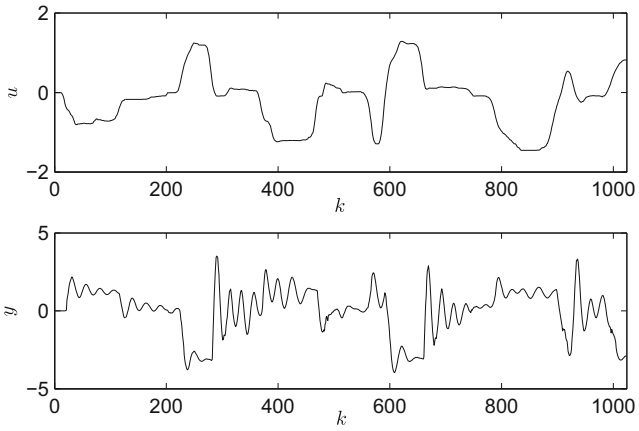

Fig. 4 The input and output of hydraulic robot actuator

Table 1 The Prediction performance of hydraulic robot arm

	RMS	Input vector
Method 1 [7]	NaN	ϕ_5
Method 2 [7]	0.598	ϕ_5
Method 3 [7]	0.750	ϕ_1
Method 4 [7]	0.552	ϕ_9
Wavelet-NARX [16]	0.467	ϕ_5
Two-NN-NARX [16]	0.328	ϕ_8
HAL-QARXNN	0.2213	ϕ_1
HAL-QARXNN	0.1148	ϕ_2
HAL-QARXNN	0.112	ϕ_3
HAL-QARXNN	NaN	ϕ_4
HAL-QARXNN	NaN	ϕ_5
HAL-QARXNN	0.111	ϕ_6
HAL-QARXNN	NaN	ϕ_7
HAL-QARXNN	NaN	ϕ_8

training of two sub-models are defined as:

$$SM1s(k) = \phi(k)\hat{\theta}_{LS}(N). \tag{29}$$

$$SM2b(k) = \phi(k)\Delta\theta^T(\phi(k)). \tag{30}$$

The output of $SM1$ $s(k)$ is performed under LSE algorithm and that of S$SM2$ $b(k)$ is performed by MIMO-NN of quasi linear-ARX model. The learning processes of hierarchical adaptive learning of linear and nonlinear sub-models of quasi linear-ARX neural networks model (HAL-QARXNN) are presented as:

1. As the initial condition, set e_{LS}=0 and set $i = 1$, i is the training sequence.
2. Estimate $\hat{\theta}_{LS}(N)$ by using LSE algorithm for SM1 using (10).
3. Calculate the output of surface sub-model $s(k)$ in SM1. Set $s(k)$ as $y_{LS}(k)$ and calculate $b(k) = y(k) - y_{LS}(k)$. Use $b(k)$ as the target output for SM2.
4. Estimate $\Delta\theta^T(\phi(k))$ using MIMO-NN injected in quasi linear-ARX neural network model (22). Use the update law in (28).

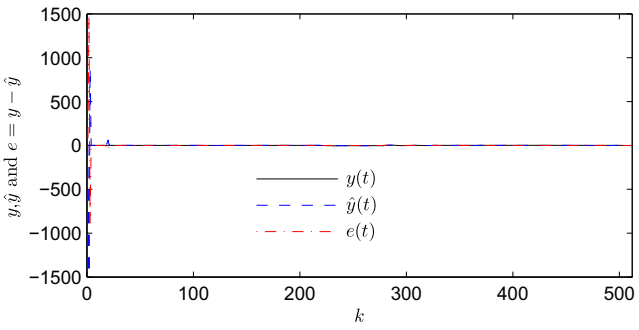

Fig. 5 The results of system identification

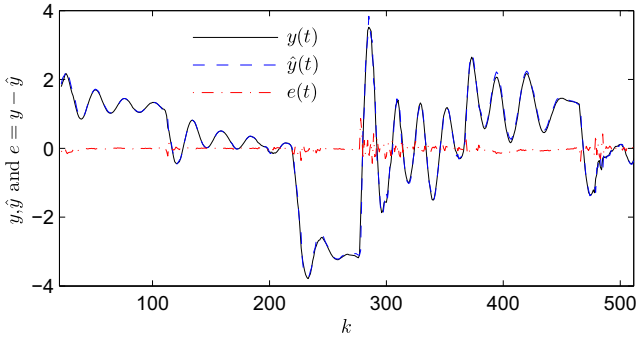

Fig. 6 The results of system identification started at twenty-second sampling

5. Update the estimated parameters $\hat{\theta}(k)$ using (18).
6. stop if a predetermined condition has been met such as the number of training or accuracy. Stop if the predetermined conditions are meet. If it is not, then go to 3). set $i = i + 1$.

7 The identification and prediction of hydraulic robot arm

In this section we will examine the proposed techniques for identification and prediction of hydraulic robot actuator taken from [2, 7, 16]. The hydraulic actuator controls the robot arm. The oil flows to the actuator by controlling the valve opening. The oil pressure is controlled to adjust the position of the robot arm. Therefore, the position of the robot arm is a function of the hydraulic oil pressure. Figure 4 shows the input-output of the hydraulic robot actuator system in which the size of valve opening is the input denoted by u and the oil pressure is the output denoted by y.

The number of data set of hydraulic robot actuator is 1024 samplings. A half of the data was used for the training and the rest was used for prediction. In the previous research, the identification and prediction of hydraulic robot arm using some methods have been studied in [7] as follows:

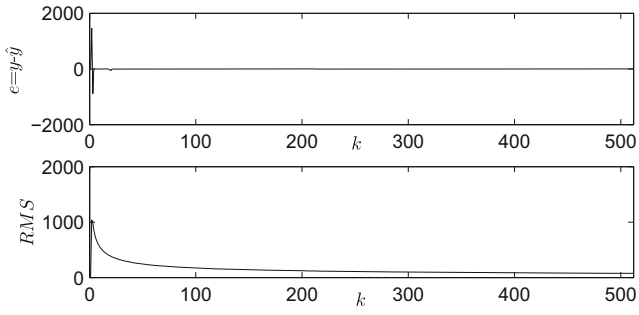

Fig. 7 The error and RMS error of system identification

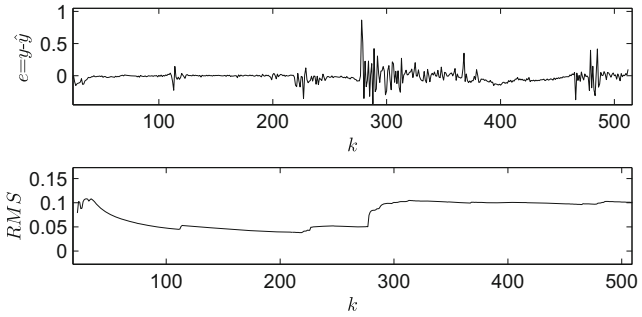

Fig. 8 The error and RMS error of system identification started at twenty-second sampling

- Method 1 : The identification scheme under Quasi-ARX Neuro Fuzzy (Q-ARX-NF) model under LS method without adding the input selection technique.
- Method 2 : The identification scheme under Quasi-ARX Neuro Fuzzy (Q-ARX-NF) model under the approach of support vector regression (SVR) without adding the input selection technique.
- Method 3 : The identification scheme under Quasi-ARX Neuro Fuzzy (Q-ARX-NF) model under LS method with adding the input selection technique performed heuristically.
- Method 4 : The identification scheme under Quasi-ARX Neuro Fuzzy (Q-ARX-NF)model under the approach of support vector regression (SVR) by adding the input selection technique using genetic algorithm (GA).

The results of prediction are compared with the NARX model based on a wavelet network and two-NN-NARX models [16]. The prediction performance is measured with rooted mean square (RMS) stated as:

$$RMS(t) = \sqrt{\frac{\sum_{t=1}^{N}(\hat{y}(t) - y(t))^2}{t}}. \tag{31}$$

Table 1 shows the results of prediction compared with the other techniques.

- ϕ_1= [y(t-1),u(t-1)]

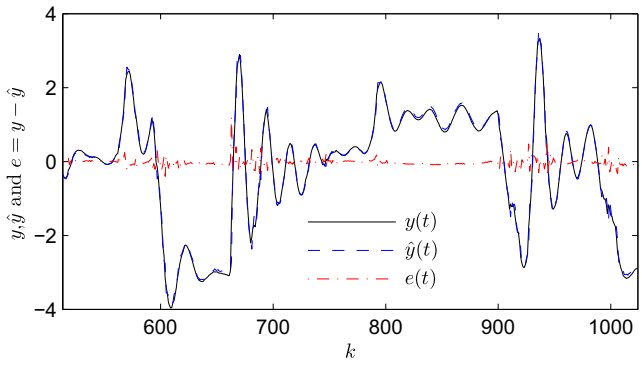

Fig. 9 The testing of prediction model of hydraulic robot arm

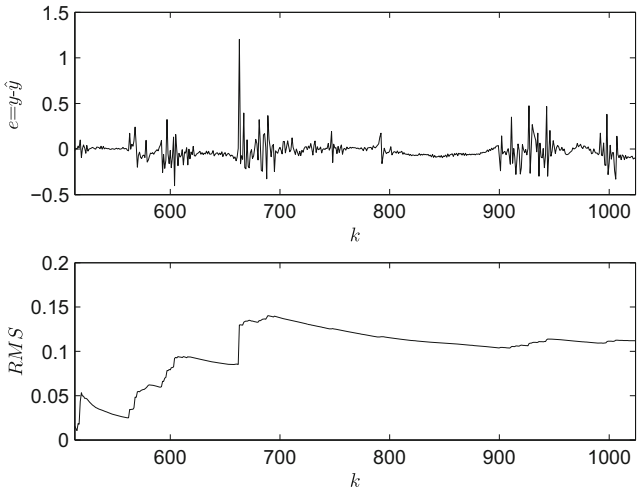

Fig. 10 The error and RMS error of prediction model

- ϕ_2= [y(t-1),y(t-2),u(t-1)]
- ϕ_3= [y(t-1),y(t-2),y(t-3),u(t-1)]
- ϕ_4= [y(t-1),y(t-2),u(t-1),u(t-2)]
- ϕ_5= [y(t-1),y(t-2),y(t-3),u(t-1),u(t-2)]
- ϕ_6= [y(t-1),y(t-2),y(t-3),y(t-4),u(t-1)]
- ϕ_7= [y(t-1),y(t-2),y(t-3),y(t-4),u(t-1),u(t-2)]
- ϕ_8= [y(t-1),y(t-2),y(t-3),y(t-4),u(t-1),u(t-2),u(t-3)]
- ϕ_9= [y(t-2),u(t-2)]

We can see on Table 1, using the input of ϕ_6 the proposed of prediction model gives better accuracy. However, the number of monomial input is more compared to the input of ϕ_3. Using the input of ϕ_3, the prediction accuracy measured by RMS index is 0.112. Hence, we use the input of ϕ_3 to analyze system identification and prediction.

The results of the system identification are shown in Fig. 5 which displays the estimated output and error. The figure shows that the modeling error is very large from the first to the twenty first sampling data. We present Fig. 6 to show the detail of the results. Meanwhile, the results which start from twenty-second sampling are shown in Fig. 6.

Table 2 Performance results and simulation of hydraulic robot arm

Models	mean	min	max	std
HAL-QARXNN	$1.97\text{x}10^{-6}$	$1.97\text{x}10^{-6}$	$1.97\text{x}10^{-6}$	0
D-MKSOM	$1.26\text{x}10^{-4}$	$1.21\text{x}10^{-4}$	$1.38\text{x}10^{-4}$	$3.27\text{x}10^{-6}$
P-MKSOM	$5.82\text{x}10^{-4}$	$3.63\text{x}10^{-4}$	0.0010	$1.35\text{x}10^{-4}$
ELM	0.0012	0.0001	0.0026	$3.22\text{x}10^{-4}$
KSOM	0.0019	0.0002	0.0247	$3.39\text{x}10^{-3}$
LLM	0.0347	0.0181	0.0651	$1.26\text{x}10^{-2}$
ARX	0.0380	0.0380	0.0380	0.0911
MLP-LM	0.0722	0.0048	0.3079	0.0640
MLP-1h	0.3485	0.2800	0.4146	0.0223
MLP-2h	0.3516	0.0980	2.6986	0.3103

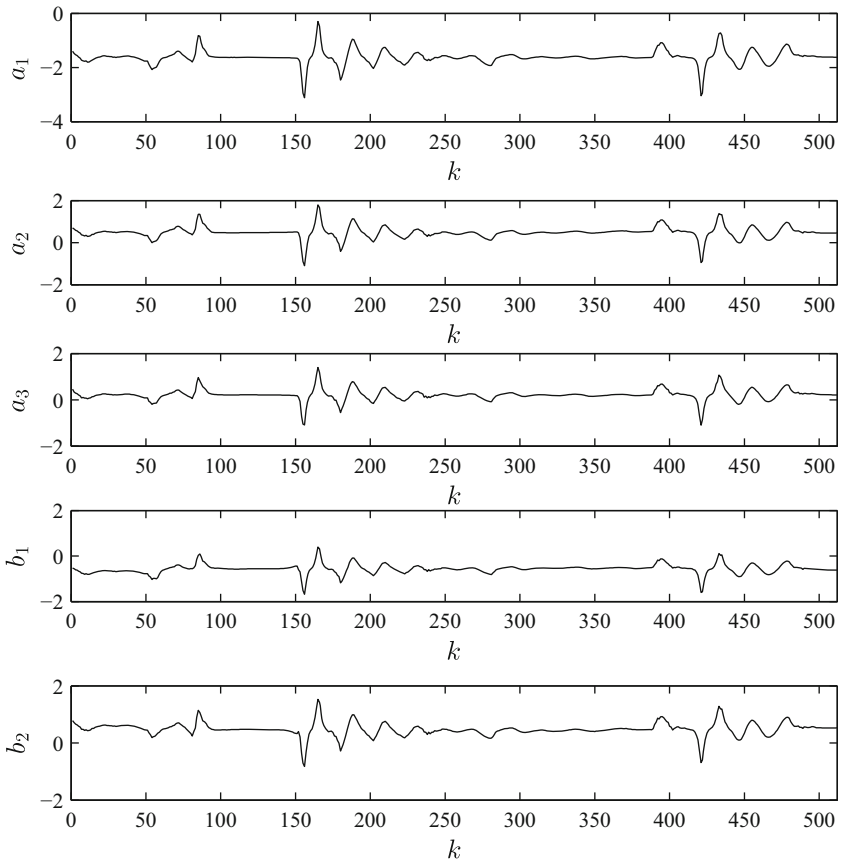

Fig. 11 The coefficients of input vector

The error and RMS error of system identification for all data training are shown in Fig. 7, and the RMS started from twenty-second sampling to the end of data training are shown in Fig. 8.

The parameters obtained from system identification is used to perform the prediction which also includes the use of the rest data sheet from 513th sampling to 1024th sampling data. We can see from the results of the prediction that, from the first to twenty-first sampling (transient data-set), the average error of transient data-set is 21.6311 $\pm$ 373.5492. Meanwhile, the average error of the rest of training data from twenty-second sampling to 512th sampling (stable data-set) is -0.0203 $\pm$ 0.0988.

The RMS error of system identification of all training data is 75.79 and the RMS error for steady state data is 0.1024. The network parameters obtained from the system identification is tested to predict the output of the system. Figure 9 shows the output of the prediction. The performance of the prediction model is measured with RMS error shown in Fig. 10. The results of the prediction show that higher RMS error in system identification does not mean that the results of prediction falls into the poor prediction performance. It is shown that the RMS error of system identification is 75.79 and the RMS of prediction is 0.112.

The performances of system identification is also compared with the other method with the similar testing procedure in [2]. The models were trained using the first 512 samples of the input-output time series. The remaining 512 samples was used for testing the prediction models. The number of hidden neurons are varied from 2 to 10 hidden neurons. The results of simulation are compared with the

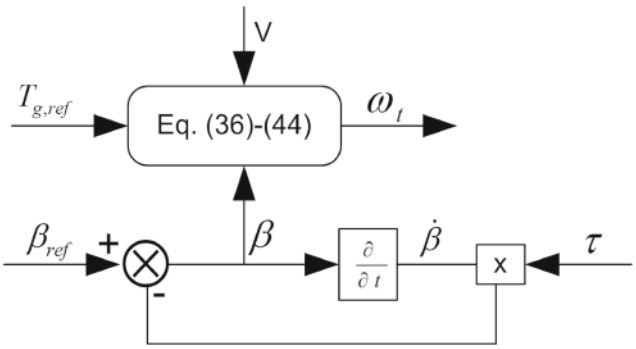

Fig. 12 Block diagram of nonlinear dynamic of WECS

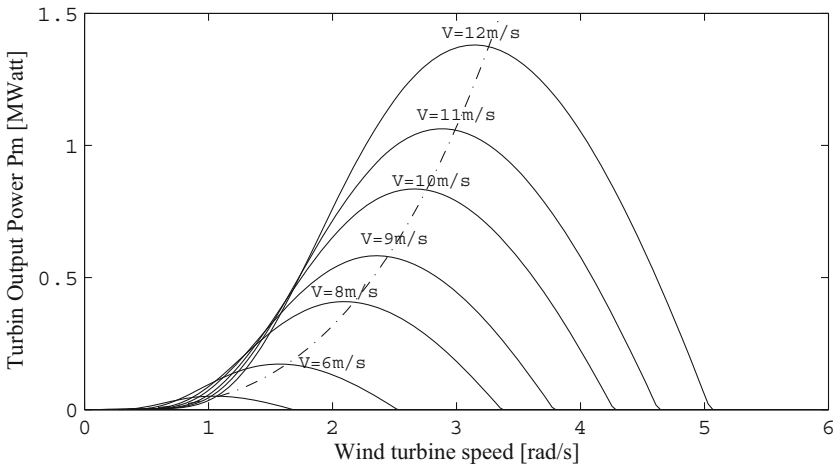

Fig. 13 Power-speed characteristics of wind turbines for various wind speeds at a pitch of 0^o

others that are shown in Table 2. The evaluation of the models is measured using the statistics of the normalized mean-squared estimation error (NMSE) stated as [2]:

$$NMSE = \frac{\sum_{t=1}^{N}(y(t) - \hat{y})^2}{N\hat{\sigma}_y^2}. \tag{32}$$

Table 2 displays the mean, minimum (min), maximum (max) and standard deviation (std) of the NMSE index for 100 independence training/testing runs where the number of hidden neurons is varied randomly. The obtained network models are tested by using the rest of 512 samples data for doing the simulation.

As we can see on Table 2, the performance of the proposed model has better performance compared with the other techniques. The learning for the network's models generally is based on the random process optimization or gradient learning. The learning rates of self-organizing map (SOM) based model (P-MKSOM and D-MKSOM models) are set with the initial learning rate 0.5 and the final learning rate 0.01, with K nearest neighbors ranging from 1 to 20. However, the selected learning rate of P-MKSOM and D-MKSOM models is not yet proved to guarantee that the optimal solution is able to reach global minima model. As we can see on the Table 2, D-MKSOM and P-MKSOM models have different maximum and minimum NMSE values. Therefore, we can interpret that the optimal solutions of network weights of SOM based model can stuck at the local minima point. We compare the proposed technique with the others, the proposed model has the consistent values of NMSE in all trial numbers. Thus, we can conclude that with the different independence trial number, the optimal solution is able to reach global minima solutions.

In the system identification techniques for control applications, the performance of a model is not only measured by accuracy but the model should have a linear nature between the input vector and its coefficient. Neural network-based models are nonlinear black-box models with non-parametric or multi-parametric models. It is difficult to inverse the model in order to create controller law. In the proposed system modeling, the NN models is used to give parameters for input vector. Thus, the parameter

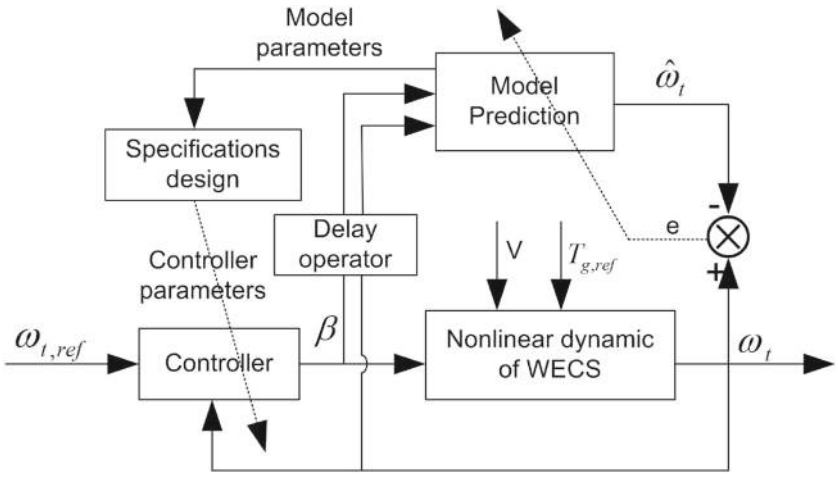

Fig. 14 Block diagram of MPPT controller of WECS

Table 3 Simulation results and comparison for tracking controller of WECS

Model Based Control	Prediction RMS	Control RMS
ARX-MVC	0.5816	0.7571
QARXNN-MVC	0.2085	0.2926
HAL-QARXNN-MVC	0.1098	0.1892

or coefficients of model can be extracted to derive the controller and to estimate the control stability [13]. The nonlinear coefficients of the prediction model displayed in Fig. 11.

8 Tracking controller of wind energy conversion system (WECS)

We create a model-based control applied for WECS tracking controller by using the proposed technique. The controller law is derived using the coefficients of ARX-like model structure [10, 11, 29]. The coefficients is calculated using a hierarchical adaptive learning of linear and nonlinear sub-models of quasi linear-ARX neural networks (HAL-QARXNN). The purpose of controller is to track the optimal performance coefficient $Cp(\lambda, \beta)$ (35) by regulating tip speed ratio λ. Hence, the controller regulates the rotor speed depend on the wind speed value to maximize the absorbed power. The power absorbed by a wind turbine is written as:

$$P_m = 0.5\rho\pi Cp(\lambda, \beta)R^2V^3 \tag{33}$$

where ρ, R, $Cp(\lambda, \beta)$, V, β, and λ are the parameters of extracted power of wind turbine model, which denote density (typically 1.25 kg/m3), radius of blades (in meter), wind-turbine power coefficient, wind speed (in m/s), pitch of the blades (in degrees) and tip-speed ratio respectively. The tip speed ratio is the ratio between the linear velocity of the blade tip to the wind speed stated as

$$\lambda = \frac{\omega_t R}{V} \tag{34}$$

where ω_t is a wind turbine shaft speed (in rad/s). The Cp is a function of λ and β which is commonly used in wind turbine simulators [26]:

$$Cp(\lambda, \beta) = 0.5176(\frac{116}{\lambda_i} - 0.4\beta - 5)e^{-21/\lambda_i} + 0.0068\lambda \tag{35}$$

$$\frac{1}{\lambda_i} = \frac{1}{\lambda + 0.008\beta} - \frac{0.035}{\beta^3 + 1}. \tag{36}$$

The aerodynamic torque is written as:

$$T_m = \frac{P_m}{\omega_t} = \frac{\rho\pi Cp(\lambda, \beta)R^3V^2}{2\lambda}. \tag{37}$$

The dynamic models of WECS is shown in Fig. 12 where the system modeling equations are stated as follows:

$$\dot{\theta} = \omega_t - \omega_g \tag{38}$$

$$J_g\dot{\omega}_g(t) = K_s\theta + B_s\omega_t - B_s\omega_g + T_g(\omega_g, T_{g,ref}) \tag{39}$$

$$J_t\dot{\omega}_t(t) = -K_s\theta - B_s\omega_t + B_s\omega_g + T_m(\beta, V) \tag{40}$$

The generator torque T_g is a nonlinear function with the generator speed ω_g and the reference electromagnetic torque $T_{g,ref}$ as a variable. Using linearization technique, the linear form of generator torque can be approximated as:

$$T_g = B_g\omega_g - T_{g,ref}. \tag{41}$$

The first-order dynamic of pitch actuator with saturation can be stated as [21, 26]:

$$\dot{\beta} = \frac{-1}{\tau}\beta + \frac{1}{\tau}\beta_{ref}. \tag{42}$$

The parameters of WECS system is taken in [17]: R=30.30 m, K_s=15.66x10^5 N/m, Bs=30.29x10^2 $N.ms/rad$, J_t=83.00x10^4 $kg.m^2$, B_g=15.99 $N.ms/rad$, J_g=5.9 $kg.m^2$, τ=100 ms. The objective of the controller is to maximize the absorbed power from wind energy by maximizing Cp. Thus, λ should be kept constant regardless of wind speed. The tracking line of maximum power point tracking (MPPT) mode of WECS is shown in Fig. 13. The power vs wind-speed characteristics (i.e., solid curves) and the maximum power point curve (i.e., dashed curve) for a pitch angle of 0^o.

The design of controller consists of two steps. First, the system is identified to create the prediction model. Second, the parameters of prediction model is used to derive the controller. The control scheme is shown in Fig. 14. In MPPT mode, the controller is to regulate the rotor speed operated at MPPT point by using the pitch control, during which generator torque is assumed in constants.

A minimum variance controller is applied for tracking control of WECS stated as follows:

$$M(t+1) = \left(\frac{1}{2}(y(t+d) - y^*(t+d))^2 + \frac{\lambda}{2}u(t)^2\right) \tag{43}$$

where λ, d are the weight of control input and delay operator. In designing of adaptive control system, the model prediction is an online step ahead prediction with d is equal to one. The controller law is derived by solving:

$$\frac{\partial M(t+1)}{\partial u} = 0 \tag{44}$$

. Hence, the control law is obtained as follows [10, 11, 29]:

$$u(t) = \frac{\hat{b}_1(t)}{\hat{b}_1^2(t) + \lambda}(\hat{b}_1(t) - \hat{b}(q^{-1}, \phi(t))q)u(t-1) + y^*(t+1) - \hat{a}(q^{-1}, \phi(t))y(t)) \tag{45}$$

The parameters of prediction model is set as controller parameter shown in (45).

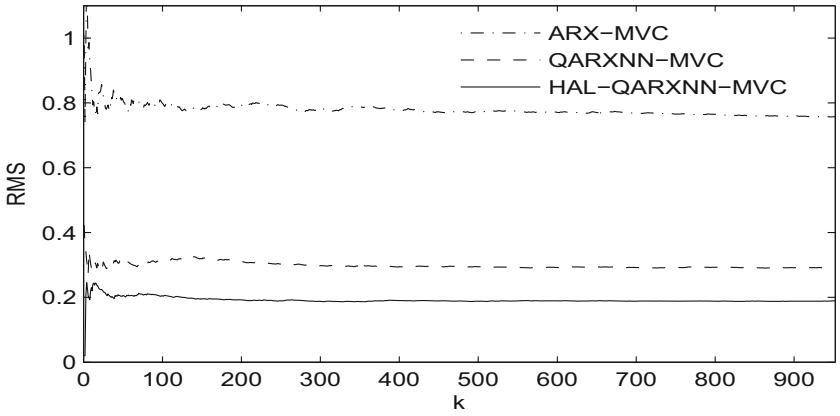

Fig. 15 The RMS performance of WECS Controllers

The performances of one step an head prediction model and control are shown in Table 3 and Fig. 15. We compare the performance of prediction model and the application of the model for control application. The HAL-QARXNN based minimum variance controller (HAL-QARXNN-MVC) is compared with Autoregressive model (ARX) based minimum variance controller (ARX-MVC) and QARXNN based minimum variance controller (QARXNN-MVC). Based on the experimental results, the proposed method gives better prediction and also better control accuracy shown in Table 3 and Fig. 15.

9 Conclusion

A hierarchical learning of linear and nonlinear sub-models with an adaptive learning has been presented. In the first step, the identification of the parameters was performed using LS algorithm. In the second step, non-linear sub-model with NN model was performed in order to improve the accuracy and convergence speed of LS estimation. Residual error of LS estimation (linear sub-model) is reduced by the training of nonlinear sub-model. A Lyapunov function based adaptive learning is proposed for the training of linear and nonlinear sub-models hierarchically. The learning rate is updated every epoch in order to guarantee that the error converges to zero. By increasing the training number, the residual error of LS sub-model can be reduced. Thus, the update of the parameters using nonlinear sub-model will converge into the true parameters. The results of theoretical analysis are proved by the results of simulation. We demonstrated the proposed method with several techniques developed previously. Based on the results of experiments, the proposed HAL-QARXNN model gives better prediction accuracy and also is able to improve control accuracy.

References

1. Algreer M, Armstrong M, Giaouris D (2012) Active online system identification of switch mode DC DC power converter based on efficient recursive DCD-IIR adaptive filter. IEEE Trans Power Elect 27(11):4425–4435
2. Barreto GA, Souza LGM (2016) Novel approaches for parameter estimation of local linear models for dynamical system identification. Appl Intell 44(1):149–165
3. Benaouda D, Murtagh F, Starckc JL, Renaudd O (2006) Wavelet-based nonlinear multiscale decomposition model for electricity load forecasting. IEEE Trans Neural Netw 70:139–154
4. Cardona DAB, Nedjah N, Mourellea LM (2017) Online phoneme recognition using multi-layer perceptron networks combined with recurrent non-linear autoregressive neural networks with exogenous inputs. Neurocomputing 265:78–90
5. Chatterjee S, Nigam S, Singh JB, Upadhyaya LN (2012) Software fault prediction using nonlinear autoregressive with exogenous inputs (narx) network. Appl Intell 37(1):121–129
6. Chen J, Fang J, Liu W, Tang T, Yang C (2018) clmf: A fine-grained and portable alternating least squares algorithm for parallel matrix factorization. Future Generation Computer Systems
7. Cheng Y, Wang L, Hu J (2012) Identification of Quasi-ARX neurofuzzy model with an SVR and GA approach. IEICE Trans Fundamentals E.95-A(5):876–883
8. Feng CB, Zheng WX (1991) Robust identification of stochastic linear systems with correlated noise. IEE Procedings-D 138(5):484–492
9. Fogel E (1981) A fundamental approach to the convergence analysis of least squares algorithms. IEEE Trans Auto Control AC-26(3):646–655
10. Hu J, Kumamaru K, Hirasawa K (2001) A Quasi-ARMAX approach to modelling of non-linear systems. Int J Control 74(18):1754–1766
11. Jami'in M, Sutrisno I, Hu J (2015) Maximum power tracking control for a wind energy conversion system based on a Quasi-ARX neural network model. IEEJ Trans Elect Electron Eng 10(4):368–375
12. Jami'in MA, Sutrisno I, Hu J (2012) Lyapunov learning algorithm for quasi-ARX neural network to identification of nonlinear dynamical system. In: Proceedings of IEEE international conference on systems, man, and cybernetics (Seoul), pp 3141–3146

13. Jami'in MA, Sutrisno I, Hu J, Mariun NB, Marhaban MH (2016) Quasi-arx neural network based adaptive predictive control for nonlinear systems. IEEJ Trans Elect Electron Eng 11(1):83–90
14. Jami'in MA, Yuyun JE (2017) Hierarchical algorithms of quasi-linear arx neural networks for identification of nonlinear systems. Eng Lett 25(3):321–328
15. Jiang J, Doraiswami R (1987) Convergence analysis of least-squares identification algorithm for unstable systems. IEE Proceedings 134(5):301–308
16. Jonas S, Zhang Q, Ljung L, Benveniste A, Delyon B, Glorennec PY, Hjalmarsson H, Juditsky A (1995) Nonlinear black-box modeling in system identification: a unified overview. Automatica 31(12):1691–1724
17. Kamal E, Aitouche A, Ghorbani R, Bayart M (2012) Robust fuzzy fault-tolerant control of wind energy conversion systems subject to sensor faults. IEEE Trans Sust Energy 3(2):231–241
18. Liu X, Wang J, Zheng W (2011) Convergence analysis of refined instrumental variable method for continuous-time system identification. IET Control Theory Appl 5(7):868–877
19. Ljung L (1977) On positive real transfer functions and the convergence of some recursive schemes. IEEE Trans Automatic Control AC-22(4):539–551
20. McElveen JK, Lee KR, Bennett JE (1992) Identification of multivariable linear systems from input/output measurements. IEEE Trans Ind Electr 39(3):189–193
21. Muhando E, Senjyu T, Yona A, Kinjo H, Funabashi T (2007) Disturbance rejection by dual pitch control and self-tuning regulator for wind turbine generator parametric uncertainty compensation. IET Control Theory Appl 1:1431–1440
22. Nassiri-Toussi K, Ren W (1994) On the convergence of least squares estimates in white noise. IEEE Trans Automatic Control 39(2):364–368
23. Peng H, Wu J, Inoussa G, Deng Q, Nakano K (2009) Nonlinear system modeling and predictive control using the rbf nets-based quasi-linear arx model. Control Eng Pract 17:59–66
24. Purwanto EC, Logeswaran R (2012) An enhanced hybrid method for time series prediction using linear and neural network models. Appl Intell 37(4):511–519
25. Rao AK, fang Huang Y, Dasgupta S (1990) ARMA parameter estimation using a novel recursive estimation algorithm with selective updating. IEEE Trans Acoustic Speech, Signal Process 38(3):447–457
26. Soliman M, Malik O, Westwick D (2011) Multiple model multiple-input multiple-output predictive control for variable speed variable pitch wind energy conversion systems. IET Renew Power Gener 5(2):124–136
27. Toivonen HT, Tötterman S, Åkesson B (2007) Identification of state-dependent parameter models with support vector regression. Int J Control 80(9):1454–1470
28. Wang H, Song G (2014) Innovative narx recurrent neural network model for ultra-thin shape memory alloy wire. Neurocomputing 134:289–295
29. Wang L, Cheng Y, Hu J (2010) A quasi-ARX neural network with switching mechanism to adaptive control of nonlinear systems. SICE Journal of Control Measurement, and System Integration 3(4):246–252
30. Yang Y, Wu QJ (2016) Multilayer extreme learning machine with subnetwork nodes for representation learning. IEEE Trans Cybern 46(11):2570–2583