Springer

Journal of
Mechanical
Science and
Technology

Journal of Mechanical Science and Technology 22 (2008) 1287~1298

www.springerlink.com/content/1738-494x

Comparative study of modeling and identification of the pneumatic artificial muscle (PAM) manipulator using recurrent neural networks

Kyoung Kwan Ahn* and Ho Pham Huy Anh

School of Mechanical and Automotive Engineering, University of Ulsan, Korea

(Manuscript Received May 8, 2007; Revised March 29, 2008; Accepted April 17, 2008)

Abstract

The paper deals with the PAM manipulator modeling and identification based on autoregressive recurrent neural networks. For the first time, the most powerful types of neural-network-based nonlinear autoregressive models, namely, NNARMAX, NNOE and NNARX models, will be applied comparatively to the PAM manipulator identification. Furthermore, the evaluation of different nonlinear neural network auto-regressive models of the PAM manipulator with different number of neurons in hidden layer is completely discussed. On this basis, the merits of each identified model of the highly nonlinear PAM manipulator have been analyzed and compared. The results show that the nonlinear NNARX model yields better performance and higher accuracy than the other nonlinear NNARMAX and NNOE model schemes. These results can be applied to model and identify not only the PAM manipulator but also to control other nonlinear and time-varying industrial systems.

Keywords: Pneumatic artificial muscle (PAM) manipulator; Autoregressive recurrent neural networks; NNARMAX model; NNARX model; NNOE model; Modeling and identification

1. Introduction

PAM actuators have been used in various precision robotic tasks as well as in human exoskeletons for enhancement of strength and mobility [1]. Due to their highly nonlinear and time-varying parameter nature, PAM manipulator modeling always presents a challenging problem that has been approached via many methodologies as follows. In [2], an antagonistic PAM pair that actuates a leg-like swinging pendulum was modeled and simulated in the lab In [3], a back-stepping controller using fuzzy model to determine valve status was designed for a simulation of a PAM hanging vertically in the lab. In [4], a fuzzy model reference learning controller was designed for a single PAM hanging vertically actuating a mass in the lab. In [5], the simulation considers PAM be modeled individually in both bicep and tricep positions. Recently, in [6], the authors applied a modified genetic algorithm for optimizing parameters of linear ARX model of the PAM manipulator. In [7], the authors successfully identified the PAM manipulator based on nonlinear neural NNARX model. All these results prove that up to now, a more efficient model for the PAM manipulator is needed, which is utilized not only in simulation but also in control of such highly nonlinear systems like PAM manipulator.

The problem of structure identification for highly nonlinear systems has received much attention in recent years, in particular for polynomial NARX, NARMAX and NOE models. Several methods and algorithms have been introduced with the aim of finding the best structure in a given model family without having to perform a parameter estimation for all possible structures [8-12]. In this paper, auto-regressive recurrent neural networks are applied to model and identify the PAM manipulator system. A PAM manipulator test system was designed. The contribution focuses on carrying experimental modeling and ana-

*Corresponding author. Tel.: +82 52 259 2282, Fax.: +82 52 259 1680
E-mail address: kkahn@ulsan.ac.kr
DOI 10.1007/s12206-008-0416-7

lyzing from gathered results the advantages of PAM manipulator identification using neural networks with different auto-regressive structures and different hidden layer nodes as well. Modeling results of the complex dynamic systems such as PAM manipulator show that the newly proposed nonlinear model presented in this paper can be applied in online control with better dynamic property, strong robustness and suitability for the identification of various plants, including linear and nonlinear processes without regard to the greatly changing external environments.

This paper is organized as follows: Section 2 gives a description of basic characteristics of PAM. In section 3, the procedure of modeling and identification of the PAM manipulator based on auto-regressive recurrent neural network is presented. Section 4 shows the hardware setup for the PAM manipulator identification. Section 5 presents the results of modeling and identification of the PAM manipulator based on auto-regressive recurrent neural network. Section 6 is the conclusion.

2. Basic characteristics of pneumatic artificial muscle (PAM)

The basic mechanical characteristics of PAM can be depicted with the equation below:

$$F = P[a(1-h)^2 - b] \tag{1}$$

where F is contraction force, P is the internal pressure (bar), h is contraction rate. a and b are constants related to the structure of PAM. This highly nonlinear feature is shown in Fig.1 extracted from [13] representing the relationship of the contraction rate h and the exerting force F of a PAM.

Eq. (1) indicates that F, P and h are three critical variables to determine the characteristics of PAM. When P is constant, F is nonlinear with respect to h.

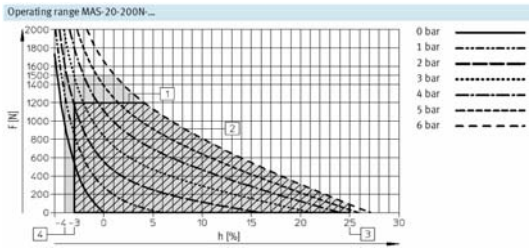

Fig. 1. h -F relationships of PAM (extracted from (FESTO, 2005)).

Meanwhile, due to the friction and wire resistance between rubber tubes and mesh shell, PAM has the characteristics of lag. Fig. 1 illustrates the ε-F relationships of artificial muscle. In this figure, the curves tell us that PAM is substantially a time-varying and nonlinear system. Thus, in order to overcome the disadvantages of nonlinearity and lag characteristics of PAM, an autoregressive recurrent neural network model is applied for identifying the PAM manipulator system.

3. Nonlinear autoregressive neural-networks-based pam manipulator identification

3.1 Description of nonlinear autoregressive model

The nomenclature of a nonlinear dynamical model is based on the terminology used to categorize linear input-output models. The linear empirical model structures can be summarized by the general family [14].

$$A(q)y(k) = \frac{B(q)}{F(q)}u(k) + \frac{C(q)}{D(q)}e(k) \tag{2}$$

where, q denotes the shift operator. For instance, *A (q)* is a polynomial in q^{-1}. This model can be given in a "pseudo-linear" regression form

$$\hat{y} = Q^T X(k)$$

where the regressors, *i.e.,* the components of *X (k)* can be given by:

- *u (k-i),* i=1,…n_b, control signals (associated with the B polynomial)
- *y (k-i),* i=1,…n_a, measured process outputs (associated with the A polynomial)
- $\hat{y}(k-i)$, simulated outputs from past *u (k)* (associated with the F polynomial)
- $e(k-i) = y(k-i) - \hat{y}(k-i)$, prediction errors (associated with the C polynomial)
- $e_u(k-i) = y(k-i) - y_u(\hat{k}-i)$, simulation errors (associated with the D polynomial)

Based on these regressors, different types of model structures can be constructed. Equation (2) can be called as the Box-Jenkins (BJ) model (A=1), the ARMAX model (F=D=1), the output-error (OE) model (A=C=D=1) and the ARX model (F=C=D=1). From this nomenclature of linear models, similar nonlinear models can be constructed as follows:

1) NNARX, Nonlinear AutoRegressive with eX-

ogenous input models, which used regressors as:

$$X(k)=\left[y(k-1),...,y(k-n_a),u(k-1),...,u(k-n_b)\right]$$

2) NNOE, Nonlinear Output Error Models, which used regressors as

$$X(k)=\left[\hat{y}(k-1),...,\hat{y}(k-n_a),u(k-1),...,u(k-n_b)\right]$$

3) NNARMAX, Nonlinear AutoRegressive Moving Average with eXogenous input models, where

$$X(k)=\left[y(k-1),...,y(k-n_a),u(k-1),...,u(k-n_b),\varepsilon_u(k-1),...,\varepsilon_u(k-n_e)\right]$$

On the soft computing and system identification of nonlinear systems, the NNARX model is called a series parallel model, while the NNOE is referred to as a parallel model. The NNARMAX and NNOE models are recurrent models, because they use the estimated output that constitutes a feedback. This makes the identification of these models difficult. Because the NNARX model structure is non-recursive, its parameters are easy to estimate. To present a dynamic system, after making some moderate assumptions, it has been proved that any nonlinear, discrete, time-invariant system can always be represented by an NNARX model [15].

3.2 Description of multilayer perceptron network system and BP learning algorithm

The multilayer perceptron (or MLP) network is probably the most-often considered member of the neural network family, mainly because of its ability to model simple as well as very complex functional relationships. This has been proven through a large number of practical applications. A fully connected two layer feed-forward MLP-network with 3 inputs, 2 hidden units (also called "nodes" or "neurons"), and 2 outputs units is shown in Fig. 2(a).

The class of MLP-networks considered in this paper is furthermore confined to those having only one hidden layer with sigmoid activating functions (*f, F*) used in hidden and output layer, respectively. The predictive output value is derived from Fig. 2(a):

$$\hat{y}_i(w,W)=F_i\left(\sum_{j=1}^{q}W_{ij}h_j(w)+W_{i0}\right)$$
$$=F_i\left(\sum_{j=1}^{q}W_{ij}\cdot f_j\left(\sum_{l=1}^{m}w_{jl}z_l+w_{j0}\right)+W_{i0}\right) \quad (3)$$

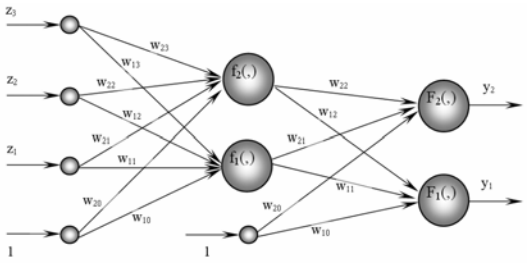

Fig. 2(a). Structure of feed-forward MLP-network

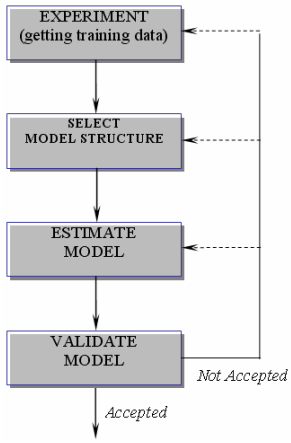

Fig. 2(b). Procedure of PAM manipulator modeling and identification.

The weights (specified by the vector θ, or alternatively by the matrices $\boldsymbol{w}$ and $\boldsymbol{W}$) are the adjustable parameters of the network, and they are determined from a set of *examples* through the process called *training*. The examples, or the training data as they are usually called, are a set of inputs, *u(t)*, and corresponding desired outputs, *y(t)*.

Specify the training set by:

$$Z^N=\left\{\left[u(t),y(t)\right]\middle|t=1,...,N\right\} \quad (4)$$

The objective of training is then to determine a mapping from the set of training data to the set of possible weights: $Z^N\rightarrow\hat{\theta}$ so that the network will produce predictions $\hat{y}(t)$, which in some sense are "*closest*" to the true joint angle outputs *y(t)* of PAM manipulator.

The prediction error approach, which is the strategy applied here, is based on the introduction of a measure of closeness in terms of a mean sum of squared error (MSSE) criterion:

$$E_N\left(\theta,Z^N\right)=\frac{1}{2N}\sum_{t=1}^{N}\left[y(t)-\hat{y}(t|\theta)\right]^T\left[y(t)-\hat{y}(t|\theta)\right] \quad (5)$$

Based on back-propagation (BP) training algorithms, the weighting value is calculated as follows:

$$W(k+1)=W(k)-\lambda\frac{\partial E(W(k))}{\partial W(k)} \quad (6)$$

with k is k^{th} iterative step of calculation and λ is learning rate which is often chosen as a small constant value.

Concretely, the weights W_{ij} and w_{jl} of weighting vector θ of recurrent neural model are then updated as:

$$\begin{aligned} &W_{ij}(k+1)=W_{ij}(k)+\Delta W_{ij}(k+1)\\ &\Delta W_{ij}(k+1)=\lambda.\delta_i.O_j\\ &\delta_i=\hat{y}_i(1-\hat{y}_i)(y_i-\hat{y}_i) \end{aligned} \quad (7)$$

with δ_i is search direction value of i^{th} neuron of output layer (i= *[1 → m]*); O_j is the output value of j^{th} neuron of hidden layer (j= *[1 → q]*); y_i and $\hat{y}_i$ are truly real output and predicted output of i^{th} neuron of output layer (i= *[1 → m]*), and

$$\begin{aligned} &w_{jl}(k+1)=w_{jl}(k)+\Delta w_{jl}(k+1)\\ &\Delta w_{jl}(k+1)=\lambda.\delta_j.u_l\\ &\delta_j=O_j(1-O_j)\sum_{i=1}^{m}\delta_i W_{ij} \end{aligned} \quad (8)$$

in which δ_j is search direction value of j^{th} neuron of hidden layer (j= *[1 → q]*); O_j is the output value of j^{th} neuron of hidden layer (j= *[1 → q]*); u_l is input of l^{th} neuron of input layer (l= *[1 → n]*).

These results of Eqs. (7) and (8) are demonstrated as follows in the case of sigmoid being the activating function of the hidden and output layers:

Consider in the case of the output layer:

Error to be minimized: $$E=\frac{1}{2}\sum_{i=1}^{m}(\hat{y}_i-y_i)^2 \quad (9)$$

Using chain rule method, we have:

$$\frac{\partial E}{\partial W_{ij}}=\frac{\partial E}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial S_i}\frac{\partial S_i}{\partial W_{ij}} \quad (10)$$

From Eq. (9), the following equation is derived.

$$\frac{\partial E}{\partial \hat{y}_i}=(\hat{y}_i-y_i) \quad (11)$$

With $S_i=\sum_{j=1}^{q}W_{ij}.O_j+bias_i$ as sum calculation at i^{th} node of output layer and $\hat{y}_i=\frac{1}{1+e^{-S_i}}$, it gives

$$\begin{aligned} \frac{\partial \hat{y}_i}{\partial S_i}&=\frac{e^{-S_i}+1-1}{\left(1+e^{-S_i}\right)^2}=\frac{1}{\left(1+e^{-S_i}\right)}\left(1+\frac{-1}{1+e^{-S_i}}\right)\\ &=\hat{y}_i(1-\hat{y}_i) \end{aligned} \quad (12)$$

$$\frac{\partial S_i}{\partial W_{ij}}=O_j \quad (13)$$

Replacing (11), (12), (13) to (10) and then putting all to (6), the following equation is derived.

$$\begin{aligned} &W_{ij}(k+1)=W_{ij}(k)+\Delta W_{ij}(k+1)\\ &\Delta W_{ij}(k+1)=\lambda.\delta_i.O_j\\ &\delta_i=\hat{y}_i(1-\hat{y}_i)(y_i-\hat{y}_i) \end{aligned} \quad (14)$$

Eq. (7) for updating the weighting values of output layer has been demonstrated.

The same way can be applied to demonstrate Eq. (8) for updating the weights of the hidden layer.

3.3 Identification of PAM manipulator using autoregressive neural networks model

The procedure executed to identify the PAM manipulator using autoregressive neural networks (RNN) model consists of four basic steps (Fig. 2(b)).

In order to validate the models we apply the measure of closeness between the predicted outputs and the measured outputs, which is called the *Error Index (EI)*. It is defined as the normalized mean square error of the residual (MSSE) given by

$$V_N\left(\theta,Z^N\right)=\frac{1}{2N}\sum_{t=1}^{N}\left[y(t)-\hat{y}(t|\theta)\right]^T\left[y(t)-\hat{y}(t|\theta)\right] \quad (15)$$

4. Experimental setup of PAM manipulator

An experimental system of modeling and identification of the PAM Manipulator based on autoregressive neural networks model is illustrated in Fig. 3.

Table 1 presents the configuration of the hardware set-up installed from Fig. 3 to model and identify the autoregressive neural networks model of the prototype PAM manipulator.

The hardware includes an IBM compatible PC (Pentium 1.7GHz) which sends the PRBS voltage signal *u(t)* to control the proportional valve (FESTO, MPYE-5-1/8HF-710B), through a D/A board (ADVANTECH, PCI 1720 card) which changes a digital signal from a PC to analog voltage *u(t)*. The rotating torque is generated by the pneumatic pressure difference supplied from an air-compressor between the antagonistic artificial muscles. Consequently, the joint of the PAM manipulator will be rotated. The joint angle, θ[deg], is detected by a rotary encoder (METRONIX, H40-8-3600ZO) and fed back to the computer through an 32-bit counter board (COMPUTING MEASUREMENT, PCI QUAD-4 card) which changes digital pulse signals to joint angle value *y(t)*. The pneumatic line is conducted under a pressure of 5[bar] and the software is coded in C program language. Experimental system of the prototype PAM manipulator is shown in Fig. 4.

Table 1. Lists of the experimental hardware set-up.

No.	Name	Model name	Company
1	Proportional valve	MPYE-5-1/8HF-710 B	FESTO
2	Pneumatic artificial muscle	MAS-10-N-220-AA-MCFK	FESTO
3	D/A board	PCI 1720	ADVANTECH
4	Counter board	PCI QUAD-4	COMPUTING MEASUREMENT
5	Rotary encoder	H40-8-3600ZO	METRONIX

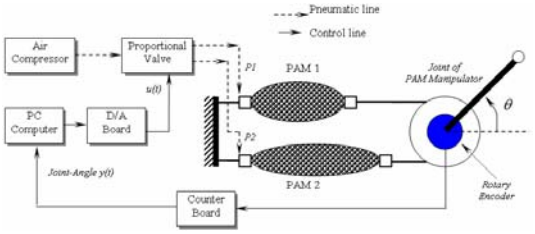

Fig. 3. Block diagram for obtaining PRBS training data of the 1-Link PAM manipulator.

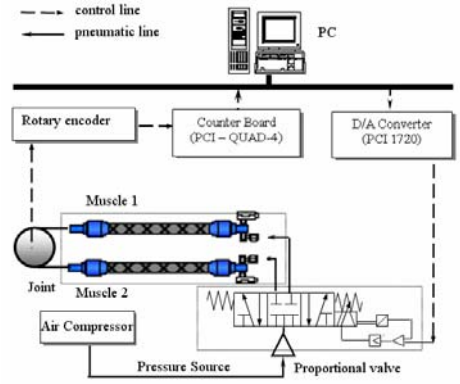

Fig. 4. Experimental system of the prototype PAM manipulator.

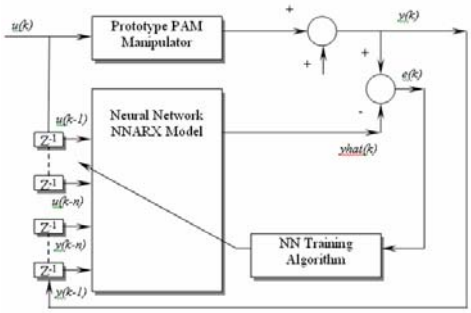

Fig. 5(a). Block diagram of the PAM manipulator autoregressive neural networks NNARX model identification

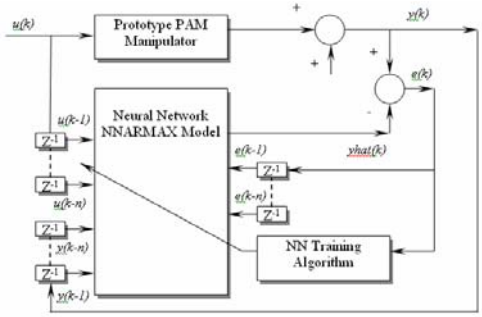

Fig. 5(b). Block diagram of the PAM manipulator autoregressive neural NNARMAX model identification

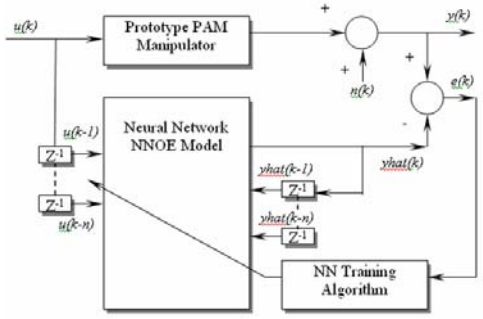

Fig. 5(c). Block diagram of the PAM manipulator autoregressive neural networks NNOE model identification.

To obtain the experimental data that describe the underlying intrinsic features of the PAM manipulator, Figs. 5(a), 5(b), and 5(c) represent an experimental diagram for modeling and identifying the neural network NNARX model, NNARMAX model and NNOE model of the prototype PAM manipulator.

A series of experimental frequency responses by a pseudo-random binary signal (PRBS) is performed. A PRBS is easily derived as a random ordered maximum length sequence (*m* sequence) of logic one and zero, emanating from a specially configured *m*-stage linear feedback shift register, which repeats after a characteristic length $L=2^m-1$. PRBS is deployed in the identification process with register length 8 and the bit magnitude is set to $\pm 50[\%]$, which corresponds to the control voltage U input [v]. Fig. 6(a) presents the PRBS input applied to the tested PAM manipulator and the responding joint angle output collected from it. This experimental input-output data is used for training and validating the resulting nonlinear neural NNARMAX, NNOE and NNARX model, respectively.

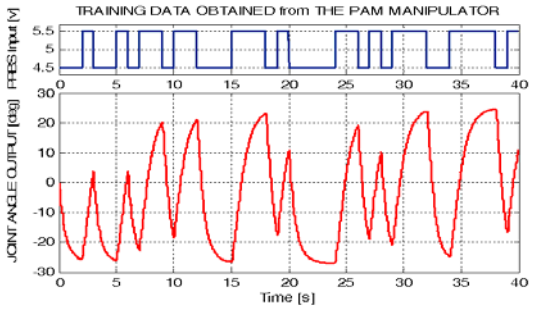

Fig. 6(a). Training data obtained from the prototype PAM manipulator

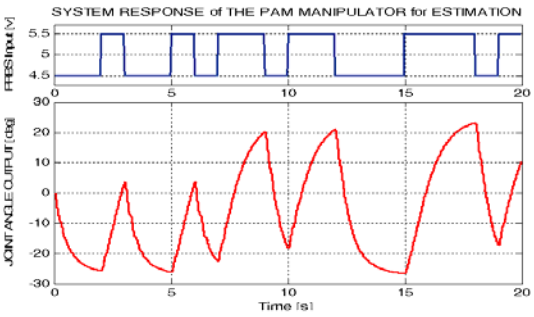

Fig. 6(b). System response of the PAM manipulator for estimation

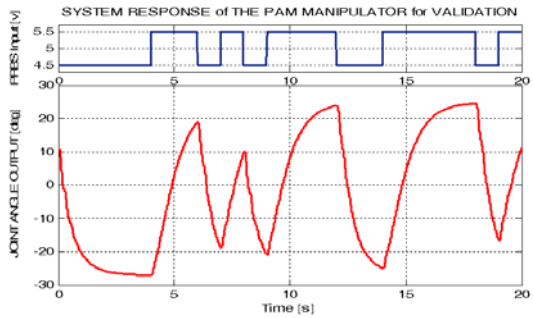

Fig. 6(c). System response of the PAM manipulator for validation.

PRBS input during the first 20 seconds and the corresponding PAM manipulator joint angle output will be used for training (Fig. 6(b)), while PRBS input during the consecutive 20 seconds and the corresponding PAM manipulator joint angle output as well will be used for validation of the derived models (Fig. 6(c)).

5. Results of modeling and identification of the pam manipulator based on recurrent neural networks (RNN) model

For comparison of the performance of three typical recurrent neural network models (NNARX, NNARMAX and NNOE), the following tests are performed. Two key parameters will be exploited to modify in these tests: the number of neurons of hidden layer (N_h) and the regressor vector (n_a, n_b) of the identified recurrent neural network model.

The 1st test carries on the modeling and identification of the prototype PAM manipulator based on recurrent neural network model with the same configuration (n_a=2, n_b=2) of regressor vector for three different typical identified NN Models (NNARMAX, NNOE and NNARX Model). The hidden layer of all three NN models composes N_h=5 nodes; the training algorithm uses Levenberg-Marquardt (LM) training algorithm; the number of iterations in training process is 100; and the tangent hyperbolic function is used for training hidden nodes and linear functions for the output node.

The comparison of the results concerning the structure of each type of neural network model, the convergence of fitness values and the validation results that represent the performance in comparison between the real PAM manipulator output and predictive NN Model output are shown in the following Figs. (7-9). The final values of weights of derived recurrent neural network models (NNARMAX, NNOE and NNARX Model) are tabulated in Tables 2,

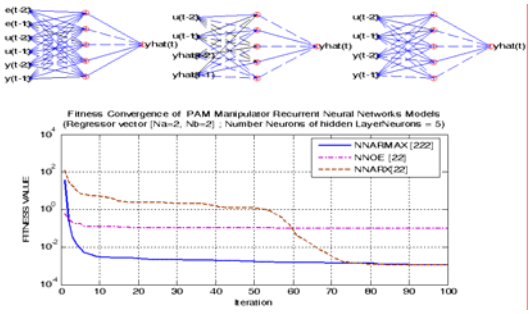

Fig. 7. Fitness comparison of PAM manipulator recurrent NN models-(n_a=n_b=2, hidden layer neurons=5).

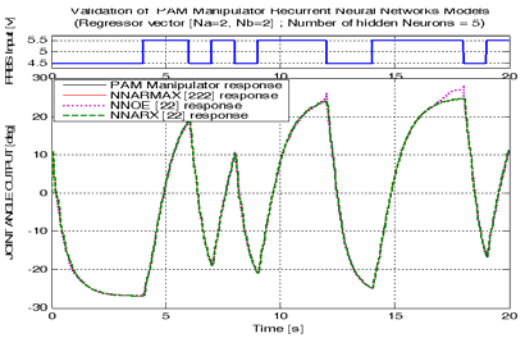

Fig. 8(a). Comparison of the Performance of PAM manipulator RNN Models-(n_a=n_b=2, hidden layer neurons=5).

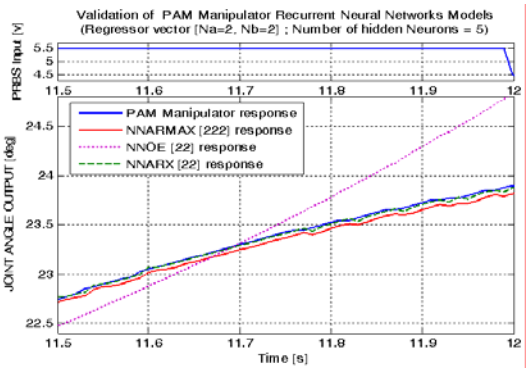

Fig. 8(b). Comparison of the Performance of PAM manipulator RNN Models-(n_a=n_b=2, hidden layer neurons=5).-zooming (11.5-12) [s].

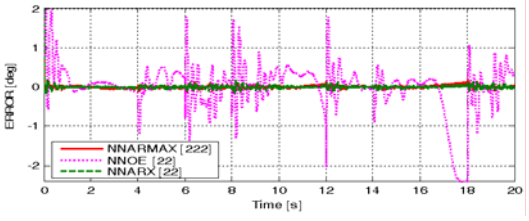

Fig. 9. Error index comparison of PAM manipulator recurrent NN models-(n_a=n_b=2, hidden layer neurons=5).

Table 2. The resultant weights of NNARMAX [222]-Number of Neurons of Hidden Layer=5.

	w_{ji} – weights of Input Layer						w_{j0} – weights of Bias Input Layer	W_{ki} - weights of Hidden layer	W_{k0} - weight of Bias Hidden layer
i / *j*	1	2	3	4	5	6	0	***k=1***	
1	1.5933	1.1423	-0.077533	-0.098761	0.66239	0.30528	0.29006	0.54993	
2	0.04827	-0.019418	-0.0011369	1.3063	-0.031017	0.0089411	-6.4596	29.049	
3	-1.1775	1.179	0.27982	-0.18217	-3.1622	-1.4819	-0.79763	-0.37933	
4	-0.037537	0.0062298	-0.0046332	1.6171	0.038212	0.0058295	-8.1336	-24.836	
5	-1.3992	-1.4091	0.035701	0.042099	0.1799	0.044579	0.20814	0.52862	
0									-2.3971

Table 3. The resultant weights of NNOE [22]-Number of Neurons of Hidden Layer=5.

	w_{ji} – weights of Input Layer				w_{ji} – weights of Bias Input Layer	W_{ki} - weights of Hidden layer	W_{k0} - weight of Bias Hidden layer
i / *j*	1	2	3	4	0	***k=1***	
1	-0.017735	-0.0053161	0.03395	-0.061219	0.73502	-33.388	
2	0.29148	-2.278	-3.483	-5.3051	0.063533	0.011513	
3	-0.19592	5.3639	-5.0743	-4.6359	0.798	-0.034125	
4	0.044934	-0.02249	0.015395	-0.021403	0.8468	36.01	
5	8.7972	-9.3037	2.4705	0.34974	-3.84	-0.11255	
0							-6.3391

Table 4. The resultant weights of NNARX [22]-number of neurons of hidden layer=5.

	w_{ji} – weights of Input Layer				w_{j0} – weights of Bias Input Layer	W_{kj} – weights of Hidden layer	W_{k0} – weight of Bias Hidden layer
i / *j*	1	2	3	4	0	***k=1***	
1	-0.10642	0.046371	-0.00064929	-0.012132	-0.50756	-11.046	
2	0.10574	-0.043032	0.0006769	0.0015434	1.889	11.946	
3	-0.14947	0.070857	0.00056791	-0.0037848	0.55018	-5.3275	
4	0.71378	0.68853	1.958	1.4743	0.83636	-0.022563	
5	0.12626	-0.061261	0.0012602	0.0042962	-1.6276	12.762	
0							-2.7706

3 and 4, respectively.

These figures indicate that both of the nonlinear NNARX and NNARMAX models excellently adapt the nonlinear joint of the PAM manipulator. The training LM algorithm forces the NNARX and NNARMAX models to converge quickly to the desired recurrent NN model. Furthermore, when apply-

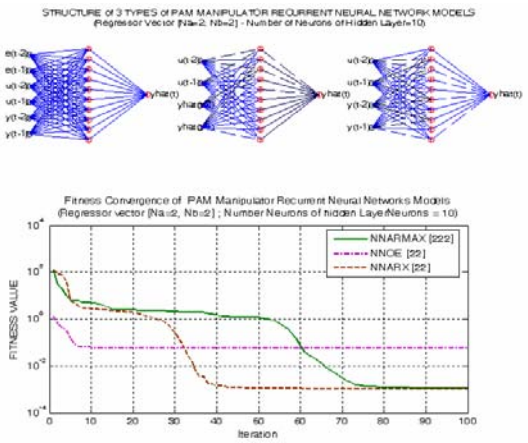

Fig. 10. Fitness comparison of PAM manipulator recurrent NN models-(n_a= n_b=2, hidden layer neurons= 10).

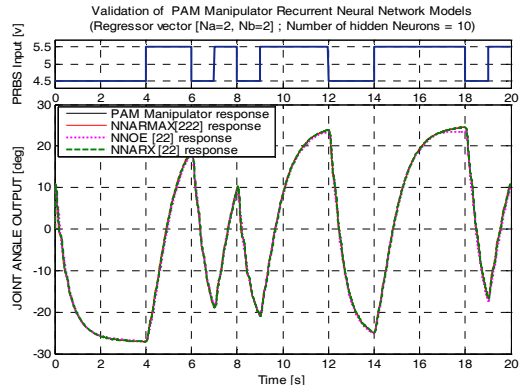

Fig. 11(a). Comparison of performance of PAM manipulator RNN models-(n_a=n_b=2, hidden layer neurons=10).

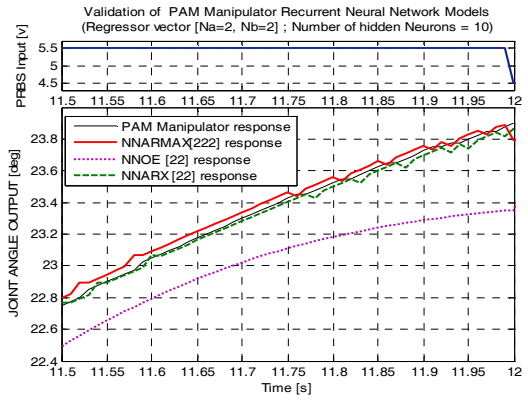

Fig. 11(b). Comparison of performance of PAM manipulator RNN models-(n_a=n_b=2, hidden layer neurons=10). –zooming (11.5-12)[s].

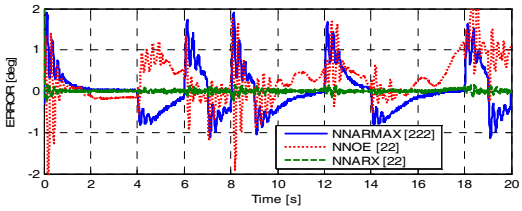

Fig. 12. Error index comparison of PAM manipulator recurrent NN models-(n_a= n_b=2, hidden layer neurons=10).

ing the recurrent neural network with only one hidden layer and regression vector n_0 being 4 (n_A=2; n_B=2), it is adequate to choose $N_{h\ of}$ about 5 to model perfectly the highly nonlinear PAM manipulator system. The error of the NNARX model proves quite superior ($\leq \pm 0.1$ [°]) in comparison with the error of NNARMAX model ($\leq \pm 0.2$ [°]) and NNOE model ($\leq \pm 2$ [°]) respectively.

The 2nd test continues the modeling and identification of the PAM manipulator based on recurrent neural network model with the same configuration (n_a=2, n_b=2) for three different typical NN Models (NNARMAX, NNOE and NNARX Model). It keeps all testing parameters like the first test except the hidden layer of all three NN models is composed of 10 neurons.

The comparison of the results concerning the structure of each NN Model, the convergence of fitness values and the performance of each autoregressive NN Model (NNARMAX, NNOE and NNARX model, respectively) are shown in following Figs.10-12. The desired values of weights of identified recurrent neural network models (NNARMAX, NNOE and NNARX Model) are tabulated in Tables 5-7, respectively.

These figures show that all three NN models with hidden nodes=10 seem to be no more performing than recurrent NN models with hidden nodes=5, except in the case of the NNOE model with its error decreased slightly from the range ± 2 [°] down to error only ± 1.5 [°]. But this advantage will also increase the complexity of the NN model with hidden nodes=10, and the time cost of iteration calculation is considerable too. Especially, the NNARMAX model shows its intrinsic unstable feature in performance with its error increased considerably from the range ± 0.2 [°] up to error ± 1.5 [°] (see Fig. 12). Finally, the NNARX model always presents its stability and robustness as well in performance with its excellent error about the range ± 0.1 [°].

Table 5. The resultant weights of NNARMAX [222]-Number of Neurons of Hidden Layer=10.

	w_{ji} – weights of Input Layer						w_{j0} – weights of Bias Input Layer	W_{ki} - weights of Hidden layer	W_{k0} - weight of Bias Hidden layer
i / *j*	1	2	3	4	5	6	0	k = 1	
1	0.72663	0.13377	0.012863	0.58692	0.15361	0.2731	0.64082	0.39324	
2	0.41195	0.20713	0.38397	0.057581	0.67564	0.25477	0.19089	0.59153	
3	0.74457	0.6072	0.68312	0.36757	0.69921	0.8656	0.84387	0.11975	
4	0.26795	0.62989	0.092842	0.63145	0.72751	0.23235	0.1739	0.038129	
5	0.43992	0.37048	0.035338	0.71763	0.47838	0.80487	0.17079	0.4586	
6	0.93338	0.57515	0.6124	0.69267	0.55484	0.9084	0.9943	0.86987	
7	0.68333	0.45142	0.60854	0.084079	0.12105	0.23189	0.43979	0.93424	
8	0.21256	0.0438	0.0157	0.45436	0.45075	0.23931	0.34005	0.26445	
9	0.83924	0.02718	0.01635	0.44183	0.71588	0.049754	0.31422	0.1603	
10	0.62878	0.3126	0.19007	0.35325	0.89284	0.078384	0.36508	0.87286	
0									0.23788

Table 6. The resultant weights of NNOE [22]-Number of Neurons of Hidden Layer=10.

	w_{ji} – weights of Input Layer				w_{j0} – weights of Bias Input Layer	W_{ki} - weights of Hidden layer	W_{k0} - weight of Bias Hidden layer
i / *j*	1	2	3	4	0	***k=1***	
1	-0.69267	-0.36791	0.2824	-0.15658	0.71697	4.2827	
2	0.13322	0.77958	-0.16679	0.24017	-1.5771	6.4621	
3	0.095898	-0.048686	-0.025256	0.027581	-1.3461	17.112	
4	0.064064	-0.011111	0.029743	-0.00489	-0.29273	10.051	
5	0.077398	-0.02563	-0.0088531	0.029734	0.92877	13.434	
6	0.022348	-0.10823	0.050773	-0.022695	-3.2899	-6.3698	
7	0.85714	-1.6286	0.061402	-0.41538	2.8898	2.1808	
8	-6.3943	6.0506	-2.1807	-2.2514	2.0539	-2.9559	
9	2.1908	-0.98621	0.58154	1.1495	1.287	0.059167	
10	-0.42182	1.0473	-2.7954	-2.0131	0.21064	-1.5541	
0							-4.8173

The 3rd test carries on the modeling and identification of the PAM manipulator based on recurrent neural network model with the enhanced configuration of the regressor vector with (n_a=3, n_b=3) for three different typical NN Models (NNARMAX, NNOE and NNARX Model). It keeps all testing parameters like the first test, in which the number of hidden layers of all three NN models is composed of 5 neurons.

The comparison of the results concerning the structure of each NN Model, the convergence of fitness values and the performance in comparison between the real PAM manipulator output and predictive NN Model output are shown in following Figs.13-15. The desired values of weights of the resultant resulting recurrent neural network models (NNARMAX, NNOE and NNARX Model) are tabulated in Tables 8-10, respectively.

Table 7. The resultant weights of NNARX [22]-Number of Neurons of Hidden Layer=10.

	w_{ji} – weights of Input Layer				w_{j0} – weights of Bias Input Layer	W_{ki} - weights of Hidden layer	W_{k0} - weight of Bias Hidden layer
i / *j*	1	2	3	4	0	***k=1***	
1	-0.72704	0.81172	0.0023689	0.1321	-0.79062	-2.2253	
2	0.12761	-0.059445	-0.0052366	0.0081257	1.9943	11.926	
3	-9.3554	5.2622	1.452	-0.38085	0.69454	1.346	
4	-0.57795	0.71604	0.06393	0.13883	-0.72567	0.014649	
5	1.9306	-4.2412	2.3251	2.6809	-0.75147	-0.017308	
6	0.092419	-0.0185	-0.0024852	0.018127	0.69591	9.1417	
7	-0.12897	0.065599	-0.0017745	-0.0034485	1.4443	-14.49	
8	0.92971	-3.4956	-2.3707	-1.6243	-0.96512	-0.026785	
9	-0.74562	1.1762	6.0616	-4.0983	-2.1745	0.040718	
10	-0.011813	0.11366	-0.0081912	0.031332	-0.70877	5.0436	
0							-2.5828

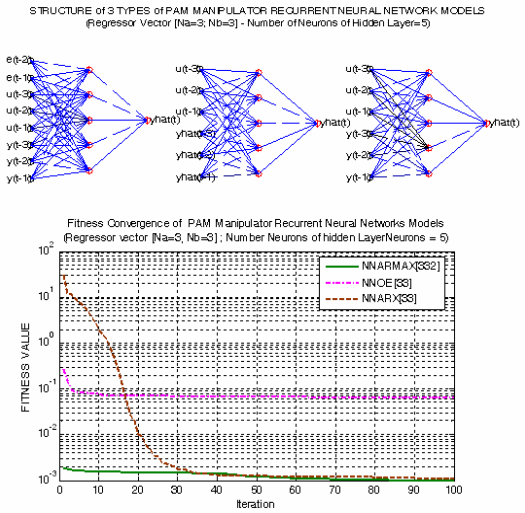

Fig. 13. Fitness comparison of PAM manipulator recurrent NN models - (n_a= n_b=3, hidden layer neurons=5).

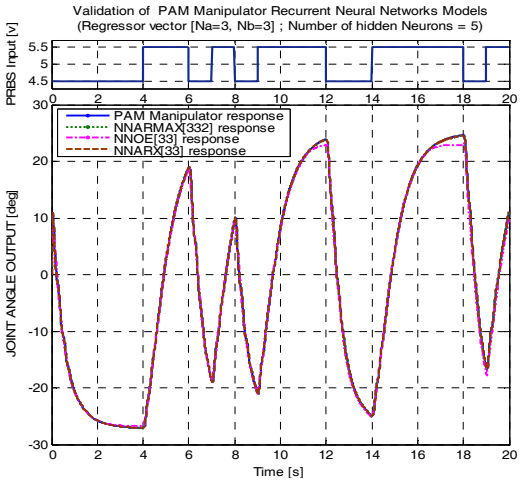

Fig. 14(a). Comparison of performance of PAM manipulator RNN models-(n_a= n_b=3, hidden neurons=5).

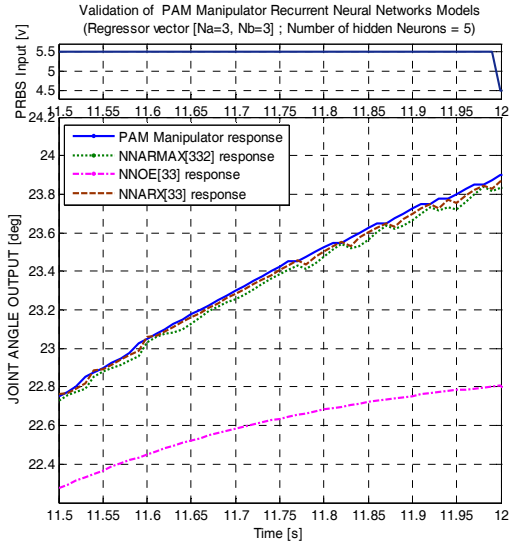

Fig. 14(b). Comparison of performance of PAM manipulator RNN models-(n_a= n_b=3, hidden neurons=5). –zooming (11.5-12)[s].

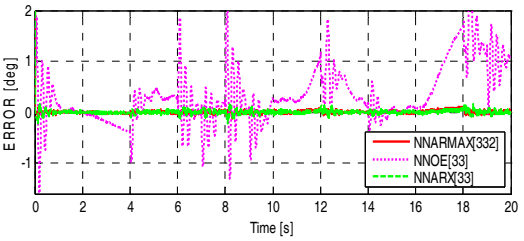

Fig. 15. Error index comparison of PAM manipulator recurrent NN models-(n_a= n_b=3, hidden layer neurons=5).

Table 8. The resultant weights of NNARMAX [332]-Number of Neurons of Hidden Layer=5.

	w_{ji} – weights of Input Layer								w_{j0} – weights of Bias Input Layer	W_{ik} - weights of Hidden layer	W_{k0} - weight of Bias Hidden layer
i / *j*	1	2	3	4	5	6	7	8	0	*k=1*	
1	0.073235	-0.065546	0.019716	0.00051002	-0.000716	0.000737	-0.021021	0.019427	-0.72404	31.271	
2	-0.080461	0.074298	-0.023716	-0.00083916	0.001191	-0.000988	0.032481	-0.018953	-1.0658	-27.1	
3	0.73051	-0.16857	-1.0695	1.8832	1.9522	1.8642	0.006482	-0.006229	-0.2027	0.3705	
4	-0.27002	0.12482	0.18328	0.20163	0.2622	0.37468	0.004534	-0.057232	0.12989	-2.496	
5	-0.17366	0.21145	-0.083757	-0.0082187	0.005681	-0.00637	0.29929	-0.10371	-0.20329	-3.784	
0											-0.866

Table 9. The resultant weights of NNOE [33]-Number of Neurons of Hidden Layer=5.

	w_{ji} – weights of Input Layer						w_{j0} – weights of Bias Input Layer	W_{ik} - weights of Hidden layer	W_{k0} - weight of Bias Hidden layer
i / *j*	1	2	3	4	5	6	0	*k=1*	
1	-1.5763	-0.56234	-0.90904	-0.21587	0.10277	-0.37957	-0.16692	0.0507	
2	-0.07697	0.17284	-0.0508559	-0.036657	0.088525	0.059495	-0.48761	12.988	
3	-0.0834	0.055679	-0.018797	-0.0057083	0.011438	0.010923	-1.4926	-17.04	
4	-0.15215	0.1962	-0.088276	-0.0055714	0.037663	0.0008309	0.99926	-17.807	
5	0.62579	0.12947	-0.79262	0.54398	-0.1079	0.033254	0.04395	1.3883	
0									-2.845

Table 10. The resultant weights of NNARX [33]-Number of Neurons of Hidden Layer=5.

	w_{ji} – weights of Input Layer						w_{j0} – weights of Bias Input Layer	W_{ik} – weights of Hidden layer	W_{k0} – weight of Bias Hidden layer
i / *j*	1	2	3	4	5	6	0	*k=1*	
1	-1.2199	-0.023401	1.4807	-0.21033	-0.17886	-0.55612	0.82492	0.079542	
2	-0.81913	-2.1201	-3.537	3.145	3.2615	3.2	0.25012	-0.023128	
3	0.042426	-0.020974	-0.0011577	0.00069379	0.00099087	0.0014491	-0.60445	39.236	
4	-0.92069	-0.084156	1.3435	-0.0030165	-0.0028313	0.12685	0.68571	0.02887	
5	0.023545	0.0095334	-0.010946	5.2208e-005	-0.00025608	0.0021564	0.79322	33.926	
0									-1.7687

These figures show that increasing the size of the regression vector n_0 being 6 (n_A=*3; n_B=3*) will only result in slightly better performance for both the NNARMAX and NNOE models in comparison with NNARMAX and NNOE models having (n_A=*2; n_B=2*). Furthermore, this attempt would highly increase the complexity of the identified NN model and the time cost of iteration calculation would be enor-

mous too. The error of the NNARX model always proves quite well and unchanged ($\leq \pm 0.1$ [°]) in comparison with the error of NNARMAX model ($\leq \pm 0.2$ [°]) and NNOE model ($\leq \pm 1.5$ [°]), which seem always worse than the NNARX model. These results determine the advantage of stability and robustness as well of the nonlinear neural network NNARX model.

Important conclusive remarks can be derived. First, the nonlinear NNARX model always yields better performance, more stability and higher accuracy than the other nonlinear NNARMAX and NNOE model schemes. Second, increasing the number of neurons of the hidden layer from 5 to 10 will only improve the performance of the NNOE model; meanwhile, the NNARMAX model proves worse than NARMAX with the number of neurons of the hidden layer equal 5. The perfect NNARX model with hidden layer 10 nodes proved the same quality. Third, these results show that the NNARMAX and NNOE models with regression vector n_0 being 6 ($n_A=3$; $n_B=3$) seem quite not better than in the case of the regression vector n_0 being 4 ($n_A=2$; $n_B=2$), whose quality is always worse than the corresponding NNARX model. Furthermore, this disadvantage will also be duplicated because of the complexity of such NNARMAX and NNOE models as well and because of the time cost of iteration calculation, which is considerable too. Finally, all three autoregressive neural networks models, namely, NNARX-NNARMAX and NNOE demonstrate their superb performance in comparison with conventional MLPNN model and linear ARX model as well (see [6]).

Consequently, the proposed auto-regressive neural networks models presented in this study, especially the NNARX model, can be applied in adaptive online control, predictive control with better dynamic property and strong robustness. Furthermore, such NN models prove quite suitable for the identification of various plants, including linear and nonlinear processes without regard to greatly changing external environments.

6. Conclusions

In the paper, model-based identification is discussed with respect to neural networks as universal approximators yielding the possibility of modeling nonlinear autoregressive models with the exogenous variable. Neural networks utilized in modeling and identification of the PAM manipulator in this paper have overcome successfully the nonlinear characteristics of the prototype PAM manipulator system, and the resulting recurrent neural network models would surely enhance the control performance of the PAM manipulator, due to the extraordinary capacity in learning nonlinear characteristics. The results show that the nonlinear NNARX model yields better performance and higher accuracy than the other NNARMAX and NNOE model schemes. Finally, three proposed recurrent NN models (NNARX-NNARMAX and NNOE) can be applied to model, identify and control not only of the PAM manipulator but also other nonlinear and time-varied parametric industrial systems.

Acknowledgment

This research was partially supported by University of Ulsan.

Nomenclature

δ_j : Search direction value of j^{th} neuron of hidden layer *(j=[1→q])*

δ_i : Search direction value of i^{th} neuron of output layer *(i=[1→ m])*

λ : Learning rate

θ : The weighting set (=*W*)

E : Error to be minimized

E_N : Summed error of batch training mode with N input-output samples

F_i : Activation function of i^{th} neuron of the output layer

f_j : Activation function of j^{th} neuron of the hidden layer

K : Number of steps used to accumulate the error values

m : Number of neurons of output layer

N : Number of input-output training samples

n : Number of neurons of input layer

n_a : The order of output $y(z^{-1})$

n_b : The order of input $u(z^{-1})$

n_k : The time delay (in this paper, $n_k = T = 1$)

q : Number of neurons of hidden layer

r : Ratio between predicted error and current error

O_j : The j^{th} output from the hidden layer *(j=[1→q])*

u_l : The l^{th} input to the input layer *(l=[1→n])*

W_{ij} : Weight from the j^{th} neuron in the hidden layer to the i^{th} neuron of the output layer

w_{jl} : Weight from the l^{th} neuron in the input layer to the j^{th} neuron of the hidden layer

y_i : The i^{th} output from the output layer *(i=[1→m])*

$\hat{y}_i$: The i^{th} predicted output from the output layer *(i=[1→m])*

z_l : The l^{th} output from the input layer (in this paper, $u_l = z_l$)

Z^N : The training set with N input-output samples

References

[1] C. P. Chou and B. Hannaford, Measurement and Modeling of McKibben pneumatic artificial muscle, *IEEE Trans. on Robotics and Automation,* 12 (1) (1996) 90-102.

[2] R. Q. van der Linde, Design, analysis, and control of a low power joint for walking robot by phasic activation of McKibben muscle, *IEEE Trans. on Robotics and Automation,* 15 (4) (1999) 599-604.

[3] P. Carbonell, Z. P. Jiang and D. Repperger, A fuzzy back-stepping controller for a pneumatic muscle actuator system, Proceedings IEEE Int. Symp. Intelligent Control, Mexico City, Mexico, (2001) 353-358.

[4] S. W. Chan, J. Lilly, D. W. Repperger and J. E. Berlin, Fuzzy PD+I learning control for a pneumatic muscle, Proc. of 2003 IEEE Int. Conf. Fuzzy Systems, St. Louis, MO, USA. (2003) 278-283.

[5] J. Lilly, Adaptive tracking for pneumatic muscle actuators in bicep and tricep configurations, *IEEE Trans. Neural Syst. Rehab. Eng.,* 11 (3) (2005) 56-63.

[6] K. K. Ahn and H. P. H. Anh, System modeling and identification of the two-link pneumatic artificial muscle (PAM) manipulator optimized with genetic algorithm, Proc. 2006 of IEEE-ICASE Int. Conf., Busan, Korea, (2006) 4744-4749.

[7] K. K. Ahn and H. P. H. Anh, A new approach of modeling and identification of the pneumatic artificial muscle (PAM) manipulator based on recurrent neural network, *In Proceedings IMechE, Part I: Journal of Systems and Control Engineering,* 221 (8) (2007) 1101-1122.

[8] L. Piroddi and W. Spinelli, A pruning method for the identification of polynomial NARMAX models, Proc. of the 13th IFAC SYSID, (2003) 1108-1113.

[9] L. Piroddi and W. Spinelli, An identification algorithm for polynomial NARX models based on simulation error minimization, *Int. J. of Control* 76 (17) (2003) 283-295.

[10] A. Leva and L. Piroddi, NARX-based technique for the modeling of magneto-rheological damping devices, *Smart Materials and Structures,* 11 (1) (2002) 79-88.

[11] M. Lovera, L. Piroddi and W. Spinelli, A Two-stage Algorithm for Structure Identification of Polynomial NARX models, Proc. of the 2006 American Control Conference, Minnesota, USA, (2006) 2387-2393.

[12] S. Lecoeuche and S. Lalot, NNOE-Based Prediction of the daily performance of solar collectors, *International Communications in Heat and Mass Transfer*, 32 (5) (2005) 603-611.

[13] FESTO Pneumatic Artificial Muscle (PAM) Manual, (2004).

[14] J. Sjoberg, Q. Zhang and L. Ljung, Nonlinear Black-box Modeling in System Identification: a unified overview, *Automatic Journal,* 3 (1995) 1691-1724.

[15] K. Vojislav, Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models, The MIT Press, Cambridge, London, (2001).