Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), Vol. 3
Lipo Wang, Jagath C. Rajapakse, Kunihiko Fukushima, Soo-Young Lee, and Xin Yao (Editors)

PARAMETER ESTIMATION OF ARX/NARX MODEL: A NEURAL NETWORK BASED METHOD

Dan Wang, Kai-Yew Lum and Guanghong Yang

Temasek Laboratories, National University of Singapore

ABSTRACT

In this paper, we consider the parameter-estimation problem of time domain models in the form of ARX or NARX. Instead of Least-Square based methods and the Maximum Likelihood method, the parameters of the time domain model will be estimated from test data using feed-forward neural networks. In the applications of neural networks, commonly used activation functions for function approximation are nonlinear transfer functions (e.g., Log-sigmoid function, Hyperbolic tangent function and polynomial function) as well as linear transfer function. In our study, we investigate the equivalence of linear/nonlinear Autoregressive exogenous (ARX/NARX) models with feed-forward neural network models, in which the activation function of hidden neuron is one of these most frequently used functions. The coefficients of ARX/NARX models are given by neural network weight values for different activation functions. We show that for pure linear systems, ARX models can be obtained from feed-forward neural networks with commonly used hidden neuron activation functions such as log-sigmoid function, hyperbolic tangent function, polynomial function, exponential function as well as pure linear function. For nonlinear systems, ARX and NARX models can be obtained from neural networks with polynomial or exponential hidden neuron activation functions to describe the linear contribution and nonlinear contribution respectively.

1. INTRODUCTION

Due to their ability to approximate any continuous functions to a desired degree of accuracy, neural networks are frequently used as a tool for identification of nonlinear systems. The NN approach does not require any assumptions to be made about the structure of identified system. It is possible to identify a model without any *prior* knowledge [17]. On the other hand, a neural-network model reflects only the input-output behavior of the system and cannot give any insight into a physical mechanism that generates the observed output, and the model cannot be used for further analysis. In this paper, we consider the parameter-estimation problem of time-domain models in the form of ARX/NARX using neural networks. ARX/NARX models are commonly used in the system identification area. We are motivated by the prospective use of ARX/NARX model in the flutter predication of aircraft wings [1][2][3]. Inherently, flutter is a nonlinear phenomenon that can be described by a NARX model. An ARX model can be easily extracted from the respective NARX model and can be analyzed for flutter predication.

Our objective is that, instead of Least-Square based methods and the Maximum Likelihood method, the parameters of the time-domain model will be estimated from test data using a feed-forward neural network. A study by Levin and Narendra has shown that a nonlinear system can be identified by the use of multiplayer feed-forward neural networks [4]. Pham and Liu [9] and Lamy and et al [5] presented some results on neural identification of linear systems. Chon and Cohen [6] showed the equivalence of linear or nonlinear autoregressive moving-average (ARMA/NARMA) models and neural network models by demonstrating that the parameters of ARMA/NARMA models can be obtained from an analysis of a neural network model trained on input-output data. In their study, a feed-forward neural network in which the activation function of hidden units is a polynomial function is used. In our study, we first show the equivalence of linear/nonlinear Autoregressive exogenous (ARX/NARX) models with feed-forward neural network models in a general case. Using Taylor's theorem, we expand the activation function into power series form. Thus the method given by Chon and Cohen [6] is generalized to any continuous activation functions. Next, more specifically, we find the solutions for some most frequently used activation functions. In neural-network applications, commonly used activation functions for function approximation are nonlinear transfer functions (e.g., Log-sigmoid function, Hyperbolic tangent function and polynomial function) as well as linear transfer functions. The coefficients of ARX/NARX models are given by neural-network weights for these activation functions. Simulation examples are presented to illustrate the effectiveness of our proposed approach. The parameters obtained by this method with different activation functions are compared with the actual values used in simulations.

The rest of this paper is organized as follows. Section 2 establishes the equivalence of ARX/NARX models with feed-forward neural network models; Section 3 illustrates the effectiveness of our proposed approach by simulation examples with different activation functions. Finally, Section 4 concludes this paper.

2. SYSTEM MODEL AND PARAMETER ESTIMATION

2.1 System model

For generality, we consider a nonlinear dynamic system represented by the following nonlinear autoregressive exogenous (NARX) model [7]:

$$\begin{aligned} y(n) &= \sum_{i=0}^{P} a(i)u(n-i) + \sum_{j=1}^{Q} b(i)y(n-j) \\ &+ \sum_{i=0}^{P}\sum_{j=0}^{P} a(i,j)u(n-i)u(n-j) \\ &+ \sum_{i=1}^{Q}\sum_{j=1}^{Q} b(i,j)y(n-i)y(n-j) \\ &+ \sum_{i=0}^{P}\sum_{j=1}^{Q} c(i,j)u(n-i)y(n-j) + \cdots + e(n) \end{aligned} \quad (1)$$

where y(n) is the system output signal; u(n) is the input signal; e(n) is the error; P and Q represent the model order of the exogenous (linear and nonlinear) and the autoregressive terms, respectively; a(i) and a(i, j) are the coefficients of linear and nonlinear exogenous terms; b(i) and b(i, j) are the coefficients of the linear and nonlinear autoregressive terms; c(i, j) are the coefficients of the nonlinear cross terms. If all the coefficients of the nonlinear terms are zero, equation (1) will give a linear ARX model.

Define

$$u = [u(n) \quad u(n-1) \quad \cdots \quad u(n-P)]^T \quad (2)$$

$$y = [y(n-1) \quad y(n-2) \quad \cdots \quad y(n-Q)]^T \quad (3)$$

$$a = [a(1) \quad a(2) \quad \cdots \quad a(P)]^T \quad (4)$$

$$b = [b(1) \quad b(2) \quad \cdots \quad b(Q)]^T \quad (5)$$

$$A = \begin{bmatrix} a(0,0) & a(0,1) & \cdots & a(0,P) \\ a(1,0) & a(1,1) & \cdots & a(1,P) \\ \vdots & \vdots & \vdots & \vdots \\ a(P,1) & a(P,2) & \cdots & a(P,P) \end{bmatrix} \quad (6)$$

$$B = \begin{bmatrix} b(0,0) & b(0,1) & \cdots & b(0,Q) \\ b(1,0) & b(1,1) & \cdots & b(1,Q) \\ \vdots & \vdots & \vdots & \vdots \\ b(Q,1) & b(Q,2) & \cdots & b(Q,Q) \end{bmatrix} \quad (7)$$

$$C = \begin{bmatrix} c(0,0) & c(0,1) & \cdots & c(0,Q) \\ c(1,0) & c(1,1) & \cdots & c(1,Q) \\ \vdots & \vdots & \vdots & \vdots \\ c(P,1) & c(P,2) & \cdots & c(P,Q) \end{bmatrix} \quad (8)$$

Then equation (1) may be expressed as follows:

$$y(n) = a^T u + b^T y + u^T Au + y^T By + u^T Cy + \cdots + e(n)$$
(9)

2.2 Solution for the general case

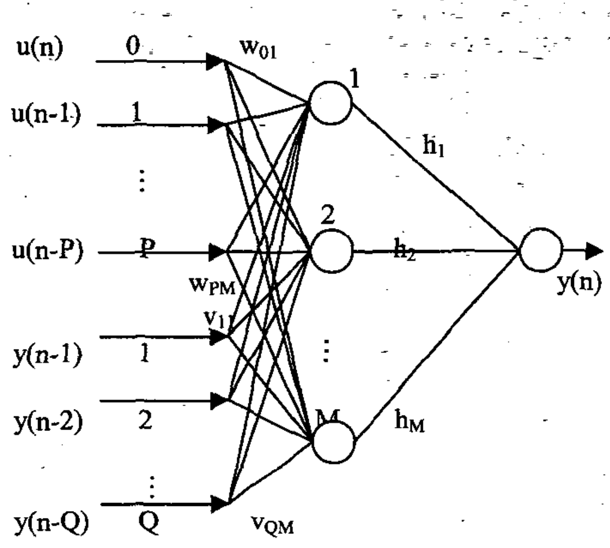

Figure 1 Feedforward neural network

Fig. 1 shows a three-layer neural network that defines an input-output mapping as follows:

$$y(n) = H^T G(x) + e(n) \quad (10)$$

where H is a vector comprising all the output weights of the neural network and G is a vector of activation functions of hidden neurons. They are defined respectively as follows:

$$H = [\, h_1 \quad h_2 \quad \cdots \quad h_M \,]^T,$$

$$G(x) = [\, g_1(x_1) \quad g_2(x_2) \quad \cdots \quad g_M(x_M) \,]^T \quad (11)$$

where x_i is the weighted sum of inputs to the i-th hidden neuron:

$$x_i = w_i^T u + v_i^T y \quad , \quad i=1, \ldots, M \quad (12)$$

Here w_i and v_i are weight vectors of the neural network and are defined as follows respectively:

$$w_i = \begin{bmatrix} w_{1i} & w_{2i} & \cdots & w_{Pi} \end{bmatrix}^T,$$
$$v_i = \begin{bmatrix} v_{1i} & v_{2i} & \cdots & v_{Pi} \end{bmatrix}^T, \quad i=1, \ldots, M \qquad (13)$$

In general, by Taylor's theorem, any continuous functions can be expanded into a Taylor's series. So we assume that the activation functions $g_i(x_i)$ of the hidden neurons have been expanded into the power series. Firstly, we consider the general case. That is:

$$g_i(x_i) = g_{0i} + g_{1i}x_i + g_{2i}x_i^2 + \cdots + g_{ni}x_i^n + \cdots \quad , \quad i=1, \ldots, M \qquad (14)$$

Substituting (11), (12), (13) and (14) into (10), we get the following expression:

$$\begin{aligned} y(n) &= h_1 g_{01} + h_2 g_{02} + h_3 g_{03} + \cdots + h_M g_{0M} \\ &+ \sum_{j=0}^{P} (h_1 g_{11} w_{j1} + h_2 g_{12} w_{j2} + \cdots + h_M g_{1M} w_{jM}) u(n-j) \\ &+ \sum_{j=1}^{Q} (h_1 g_{11} v_{j1} + h_2 g_{12} v_{j2} + \cdots + h_M g_{1M} v_{jM}) y(n-j) \\ &+ \sum_{j=0}^{P} \sum_{k=0}^{P} (h_1 g_{11} w_{j1} w_{k1} + \cdots + h_M g_{2M} w_{jM} w_{kM}) u(n-j) \\ & u(n-k) + \sum_{j=1}^{Q} \sum_{k=1}^{Q} (h_1 g_{21} v_{j1} v_{k1} + \cdots + h_M g_{2M} v_{jM} v_{kM}) \\ & y(n-j) y(n-k) + 2 \sum_{j=0}^{P} \sum_{k=1}^{Q} (h_1 g_{21} w_{j1} v_{k1} + \cdots \\ &+ h_M g_{2M} w_{jM} v_{kM}) u(n-1) y(n-k) + \cdots + e(n) \\ &= H^T G_0 + G_1^T \begin{bmatrix} h_1 & 0 & \cdots & 0 \\ 0 & h_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_M \end{bmatrix} W^T u \\ &+ G_1^T \begin{bmatrix} h_1 & 0 & \cdots & 0 \\ 0 & h_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_M \end{bmatrix} V^T y \\ &+ u^T (h_1 g_{21} w_1 w_1^T + \cdots + h_M g_{2M} w_M w_M^T) u \\ &+ y^T (h_1 g_{21} v_1 v_1^T + \cdots + h_M g_{2M} v_M v_M^T) y \\ &+ 2u^T (h_1 g_{21} w_1 v_1^T + \cdots + h_M g_{2M} w_M v_M^T) y + \cdots + e(n) \end{aligned}$$

(15)

where

$$G_0 = [g_{01} \quad g_{02} \quad \cdots \quad g_{0M}]^T,$$
$$G_1 = [g_{11} \quad g_{12} \quad \cdots \quad g_{1M}]^T \qquad (16)$$
$$W = [w_1 \quad w_2 \quad \cdots \quad w_M],$$

$$V = [v_1 \quad v_2 \quad \cdots \quad v_M] \qquad (17)$$

According to the well-known universal approximation theorem [8], with sufficient number of hidden neurons, a feed-forward neural network can approximate any smooth nonlinear function on a compact set with arbitrarily small errors. That is, after a training process, equation (15) can represent the nonlinear system given by (1) with an arbitrarily small error. Thus we can establish the equivalence of equation (15) with equation (9).

$$a^T = G_1^T \begin{bmatrix} h_1 & 0 & \cdots & 0 \\ 0 & h_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_M \end{bmatrix} W^T,$$

$$b^T = G_1^T \begin{bmatrix} h_1 & 0 & \cdots & 0 \\ 0 & h_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & h_M \end{bmatrix} V^T \qquad (18)$$

$$A = h_1 g_{21} w_1 w_1^T + h_2 g_{22} w_2 w_2^T + \cdots + h_M g_{2M} w_M w_M^T$$
(19)
$$B = h_1 g_{21} v_1 v_1^T + h_2 g_{22} v_2 v_2^T + \cdots + h_M g_{2M} v_M v_M^T$$
(20)
$$C = 2(h_1 g_{21} w_1 v_1^T + h_2 g_{22} w_2 v_2^T + \cdots + h_M g_{2M} w_M v_M^T)$$
(21)

Remark 1: The coefficients of the third order or higher order terms can be obtained similarly.

Remark 2: By assuming that the activation functions have been expanded into Taylor series form, the solution is applicable to many activation functions. Thus the method given by Chon and Cohen [6] is generalized.

2.3 Solutions for frequently used activation functions

In this section, we will further consider the equivalence of ARX/NARX model with the given neural network model for some commonly used activation functions. We shall derive the corresponding solutions of the NARX (or ARX) model coefficients.

Polynomial function

Firstly, we consider the following polynomial function as the hidden neurons' activation function:

$$p(x) = x + x^2 + \cdots + x^n + \cdots \qquad (22)$$

Using the results we obtained in above section, the coefficients of the ARX/NARX model can be given by:

$$a = WH, \quad b = VH \qquad (23)$$
$$A = h_1 w_1 w_1^T + h_2 w_2 w_2^T + \cdots + h_M w_M w_M^T \qquad (24)$$
$$B = h_1 v_1 v_1^T + h_2 v_2 v_2^T + \cdots + h_M v_M v_M^T \qquad (25)$$
$$C = 2(h_1 w_1 v_1^T + h_2 w_2 v_2^T + \cdots + h_M w_M v_M^T) \qquad (26)$$

Exponential function

If the exponential function $\exp(x) = e^x$ is used as the hidden neuron's activation function, we can express the function as the following Taylor series expansion form:

$$\exp(x) = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \cdots \tag{27}$$

The coefficients of the ARX/NARX model can be given by the following calculations:

$$a = WH, \quad b = VH \tag{28}$$

$$A = \frac{1}{2}(h_1 w_1 w_1^T + h_2 w_2 w_2^T + \cdots + h_M w_M w_M^T) \tag{29}$$

$$B = \frac{1}{2}(h_1 v_1 v_1^T + h_2 v_2 v_2^T + \cdots + h_M v_M v_M^T) \tag{30}$$

$$C = h_1 w_1 v_1^T + h_2 w_2 v_2^T + \cdots + h_M w_M v_M^T \tag{31}$$

Log-sigmoid function

One of the most commonly used activation functions for function approximation is log-sigmoid function as given by follows:

$$\log sig(x) = \frac{1}{1 + \exp(-x)} \tag{32}$$

By Taylor's theorem, the log-sigmoid function can be expanded into the following power series form:

$$\log sig(x) = \frac{1}{2} + \frac{1}{4}x - \frac{1}{8}\frac{x^3}{3!} + \cdots \tag{33}$$

Thus using the results given in above section, when log-sigmoid function is chosen as the hidden neuron activation function, it is easy to check that the solutions for the coefficients of the given ARX/NARX model are:

$$a = \frac{1}{4}WH, \quad b = \frac{1}{4}VH \tag{34}$$

$$A = [0], \quad B = [0], \quad C = [0] \tag{35}$$

Hyperbolic tangent function

The following hyperbolic tangent function is also very frequently used as hidden neuron activation functions for function approximation:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{36}$$

when $x^2 \langle \frac{\pi^2}{4}$, the hyperbolic tangent function can be expressed in the following Taylor's series form:

$$\tanh(x) = x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + \cdots \tag{37}$$

Then the coefficients of ARX/NARX model can be derived:

$$a = WH, \quad b = VH \tag{38}$$

$$A = [0], \quad B = [0], \quad C = [0] \tag{39}$$

Pure linear function

The pure linear function $g(x) = x$ is an important activation function. Sometimes it is used as an activation function to approximate a linear system. Obviously, a neural network cannot be used to approximate a nonlinear function if the hidden neuron activation functions are pure linear functions. When it is used for a linear system, the coefficients of the corresponding linear ARX model can be found as follows:

$$a = WH, \quad b = VH \tag{40}$$

Remark 3: The Taylor series expansion forms of both Log-sigmoid function and Hyperbolic tangent function are functions of only odd degree terms. If a real system includes some even degree terms, the coefficients of NARX (or ARX) model cannot be obtained with Log-sigmoid or Hyperbolic tangent hidden neuron activation functions by proposed method. So it is recommended that not to use these two functions for nonlinear systems. Obviously, pure linear activation function cannot approximate any nonlinear functions, so it can only be used when the system is linear.

3. EXAMPLES

In this section, two examples are used to illustrate the effectiveness of our proposed approach. Various neural network models are trained on simulated data to identify coefficients of ARX/NARX models.

In the first example, we generate 500 data points by a linear ARX model. The ARX model is given as follows:

$$y(n) = 0.8u(n) - 0.5u(n-1) - 0.15u(n-2) + 0.3y(n-1) - 0.2y(n-2) + 0.6y(n-3) \tag{41}$$

Then, using Neural Network Toolbox, we trained five neural network models. Their hidden neurons' activation functions are second-order polynomial function, exponential function, log-sigmoid function, hyperbolic tangent function and pure linear function respectively. The Levenberg-Marquardt algorithm is used to train the neural network models. After training neural networks on the 500 data points, the coefficients of ARX model are found according to the results in Section 2.3. The identified coefficients are given in Table 1. It can be seen that, for linear systems, excellent parameter estimation results can be achieved from any of the five neural network models. It is worth mentioning that the neural networks with second-order polynomial activation function and pure linear activation function converge very fast for the example.

The second example is a nonlinear system. The following NARX model is utilized to generate 500 data points that is used to train neural network models:

$$y(n) = 0.5u(n) - 0.7u(n-2) + 0.25y(n-1) - 0.16y(n-3) - 0.15u^2(n-1) + 0.28u(n-1)y(n-1) + 0.2y^2(n-2) \quad (42)$$

As mentioned in Remark 3, for nonlinear systems with even degree terms, only polynomial and exponential activation functions can be used in neural network models for parameter estimation. In this example, two neural networks, with polynomial and exponential hidden neuron activation function respectively, are trained on the 500 data points using the Levenberg-Marquardt algorithm. The coefficients estimated from neural network models are given in Table 2. The coefficients obtained from polynomial neural network are exactly same as their true values. The ones from exponential neural network have small errors.

4. CONCLUSIONS

In this paper, we propose a neural-network based system identification method. Neural networks are trained to model nonlinear systems and, then, ARX/NARX models can be obtained from the weights of neural networks. We have shown that, for pure linear systems, ARX models can be obtained by commonly used feed-forward neural networks such as log-sigmoid function, hyperbolic tangent function, polynomial function, exponential function as well as pure linear function. For nonlinear systems, ARX or NARX models can be obtained from neural networks with polynomial or exponential hidden neuron activation functions to describe the linear contribution and nonlinear contribution respectively. The effectiveness of our proposed approach is illustrated by examples.

5. REFERENCES

[1] G. Dimitradis and J. E. Cooper, "Flutter prediction from flight flutter test data", *Journal of Aircraft*, vol. 38, No. 2, pp. 355-367, 2001.

[2] J. K. Pinkelman and S. M. Batill, "Total least-square criteria in parameter identification for flight flutter testing," *Journal of Guidance, Control, and Dynamics*, vol. 20, No.1, pp. 74-80, 1997.

[3] Hiroshi Torii and Yuji Matsuzaki, "Flutter margin evaluation for discrete-time systems," *Journal of Aircraft*, vol. 38, No. 1, pp. 42-47, 2001.

[4] A. U. Levin and K. S. Narendra, "Identification using feedforward neural networks," *Neural Computation*, Vol. 7, pp. 349-357, 1995.

[5] D. Lamy, M. Decotte, and P. Borne, "Neural identification of linear systems," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 559 -565, Vol.1,1992.

[6] Ki H. Chon and Richard J. Cohen, "Linear and nonlinear ARMA model parameter estimation using an artificial neural network," *IEEE Transactions on biomedical engineering*, Vol. 44, No. 3, pp. 168-174, 1997.

[7] Lennart Ljung, *System identification: theory for the user*, Prentice Hall, 1999.

[8] K. Hornik, "Approximation capabilities of multiplayer feed-forward networks," *Neural networks*, Vol. 4, pp. 251-257, 1991.

[9] D. T. Pham and X. Liu, "Neural networks for discrete dynamic system identification," *J. of Syst. Eng.*, Vol. 1, pp. 51-60, 1991.

[10] Lind R. and Brenner M., "Incorporating flight data into a robust aeroelastic model," *Journal of Aircraft*, Vol. 35, No. 3, 1998.

[11] Nelles M., *Nonlinear system identification*, Springer, 2001.

[12] Johansson, R., *System modeling and identification*, Prenice Hall, 1993.

[13] Enso Ikonen and Kaddour Najim, *Advaced process identification and control*, Marcel Dekker, Inc., 2002.

[14] Eric Walter and Luc Pronzato, *Identification of parametric models from experimental data*, Springer, 1997.

[15] Jer-Nan Juang, *Applied system identification*, Prentice Hall, 1994.

[16] Graham Goodwin, *Model identification and adaptive control*, Springer, 2001.

[17] Korbicz, J. and Janczak, A., "*A neural network approach to identification of structural systems,*" Proceedings of the IEEE International Symposium on Industrial Electronics, Vol.1, pp. 98-103, 1996.

Table 1 Parameter estimated for example 1

Parameters	a_0	a_1	a_2	b_1	b_2	b_3
True Values	0.8000	-0.5000	-0.1500	0.3000	-0.2000	0.6000
Polynomial	0.8000	-0.5000	-0.1500	0.3000	-0.2000	0.6000
Exp	0.8000	-0.5000	-0.1500	0.3000	-0.2000	0.6000
Log-sigmoid	0.8000	-0.5000	-0.1500	0.3000	-0.2000	0.6000
Tanh	0.8000	-0.5000	-0.1500	0.3000	-0.2000	0.6000
Linear	0.8000	-0.5000	-0.1500	0.3000	-0.2000	0.6000

Table 2 Parameters estimated for example 2

Parameters	a_0	a_1	a_2	b_1	b_2	b_3	a(2,2)	b(2,2)	c(2,1)
True Values	.5000	.0000	-.7000	.2500	.0000	-.1600	-.1500	.2000	.2800
Polynomial	.5000	.0000	-.7000	.2500	.0000	-.1600	-.1500	.2000	.2800
Exp	.5003	.0005	-.7001	.2500	.0008	-.1599	-.1509	.2009	.2809