Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 1, 63–68

TIME-VARYING TIME-DELAY ESTIMATION FOR NONLINEAR SYSTEMS USING NEURAL NETWORKS

YONGHONG TAN*

* Laboratory of Intelligent Systems and Control Engineering
Guilin University of Electronic Technology
541004 Guilin, China
e-mail: tany@gliet.edu.cn

Nonlinear dynamic processes with time-varying time delays can often be encountered in industry. Time-delay estimation for nonlinear dynamic systems with time-varying time delays is an important issue for system identification. In order to estimate the dynamics of a process, a dynamic neural network with an external recurrent structure is applied in the modeling procedure. In the case where a delay is time varying, a useful way is to develop on-line time-delay estimation mechanisms to track the time-delay variation. In this paper, two schemes called direct and indirect time-delay estimators are proposed. The indirect time-delay estimator considers the procedure of time-delay estimation as a nonlinear programming problem. On the other hand, the direct time-delay estimation scheme applies a neural network to construct a time-delay estimator to track the time-varying time-delay. Finally, a numerical example is considered for testing the proposed methods.

Keywords: modelling, time delay, nonlinear systems, neural networks, estimation

1. Introduction

Many industrial systems involve time delays. Therefore, the identification of time delays is one of the most important issues in system modelling and identification. Some literature can be found on time-delay estimation. Reed *et al.* (1981) applied the LMS algorithm to locate the cross-correlation function in order to estimate time delay between input/output signals. Teng and Sirisena (1988) proposed an approach to extend the order of the numerator polynomial function for time-delay estimation. Lim and Macleod (1995) proposed an adaptive time-delay tracking method for the IIR filter. Shor and Messer (1997) developed a statistical method for time-delay estimation in a non-Gaussian process. Balestrino *et al.* (1998) proposed a strategy for steady-state time-delay estimation. Ching *et al.* (1999) applied wavelets to time-delay estimation. However, almost all of these approaches are only available for linear systems. It is well known that most industrial systems contain not only time delays but also nonlinearities. Hence, if the non-linearity of a process is significant, it will be necessary to develop approaches for the modelling of nonlinear processes with time delays.

During the recent decade, neural networks have been proved to be useful for system modelling and function approximation. In this paper, a dynamic neural network where the input layer has an external recurrent connection with the output of the network is used to model the dynamic nonlinear process. Time delays in some industrial processes may be varying in time. For example, the inlet flow rate being the manipulated variable of a continuous stirred tank reactor may change in time. Thus this causes variations in the manipulated time delay. In this case, on-line time-delay estimation is necessary if the effect of those variations cannot be ignored.

In this paper, two neural network based methods for the modelling of a class of nonlinear processes with time delays are proposed. The first one is called the indirect time-delay estimation method. In this method, the criterion is minimized with respect to the estimated time delay that is contained in the neural network based model used for the identification of the non-linearity and the dynamic behaviour of the process. The indirect method can be considered as solving a nonlinear programming problem. On the other hand, the second scheme is called the direct time-delay estimator formed by a neural network. In order to model a time delay, a neural network is applied. To evaluate the proposed time-delay estimation schemes, a numerical example is given for comparison.

2. Indirect Estimation Method

Suppose that the process under consideration is described by a mapping $f : \mathbb{R}^{n+m} \to \mathbb{R}$, i.e.

$$y_k = f(Y_{k-1}, U_{k-\tau_k}), \tag{1}$$

where $f(\cdot) \in C^2$, $Y_{k-1} = [y_{k-1}, \ldots, y_{k-n}]^T \in \mathbb{R}^n$ and $U_{k-\tau_k} = [u_{k-\tau_k-1}, u_{k-\tau_k-2}, \ldots, u_{k-\tau_k-m}]^T \in \mathbb{R}^m$ are respectively the output and input vectors, and τ_k is a time delay. A neural network based model used to describe the process is of the form

$$\hat{y}_k = {W^2}^T S(x), \tag{2}$$

where $\hat{y}_k$ denotes the output of the neural model with time delay, $W^2 = [w_1^2, w_2^2, \ldots, w_h^2]^T \in \mathbb{R}^h$ is the weight vector connecting the outputs of the hidden layer and the output of the network, $S(x) = [s(x_1), \ldots, s(x_h)]^T \in \mathbb{R}^h$ is the output vector of the hidden layer,

$$s(x_i) = \frac{1 - e^{-x_i}}{1 + e^{-x_i}}$$

is the sigmoid function, and the inputs of the sigmoid function in the hidden layer are expressed as

$$x_i = \sum_{j=1}^{n_a} w_{ij}^1 \hat{y}_{k-j} + \sum_{j=1}^{n_b} w_{i,n_a+j}^1 u_{k-\hat{\tau}_k-j}, \quad i = 1, \ldots, h, \tag{3}$$

where $\hat{\tau}_k$ is the estimate of the time delay, n_a and n_b are respectively the lags of the output and input of the neural model, and w_{ij}^1 represents the weights connecting inputs with hidden nodes. The introduction of the auto-regression of the model output into the network can be useful to simulate the process dynamics.

Consider the case where the delay is time varying. It is supposed that the time delay can be split into integer and fractional parts, i.e.

$$\hat{\tau}_k = \hat{d}_k + \delta\hat{\tau}_k, \tag{4}$$

where $\hat{d}_k$ is the integer part of the delay whilst $\delta\hat{\tau}_k$ denotes the fractional part of the delay, which is constrained to lie within the range of one sample period, i.e. $(2, 3)$ (Lim and Macleod, 1995). Suppose that the change in the time delay is much slower than the sampling rate so the time delay can be considered constant during one sample period. In order to identify the time delay, the fractional part of the time delay is considered as one of the parameters to be estimated. Then, the integer part of the time delay can be deduced from the estimated result of the fractional part. For the estimation of the fractional part of time delay, the gradient of the output of the neural model with respect to $\delta\hat{\tau}$ should be calculated. From (2) and (3), it follows that

$$\frac{\partial \hat{y}_k}{\partial \hat{\tau}} = \frac{\partial \hat{y}_k}{\partial \delta\hat{\tau}} = \sum_{i=1}^{h} w_i^2 s'(x_i) \Big(\sum_{j=1}^{n_a} w_{ij}^1 \frac{\partial \hat{y}_{k-j}}{\partial \delta\hat{\tau}} + \sum_{j=1}^{n_b} w_{i,n_a+j}^1 \frac{\partial u_{k-\hat{\tau}_k-j}}{\partial \delta\hat{\tau}} \Big), \tag{5}$$

where

$$s'(x) = \frac{\mathrm{d}s(x)}{\mathrm{d}x} = 0.5(1 - s(x)^2).$$

In (5), the effect of the recurrent connection to the gradient was considered. Using a first-order interpolation, we can estimate $\partial u_{k-\hat{\tau}_k-j}/\partial\delta\hat{\tau}$. The Taylor series expansion leads to

$$\begin{aligned} u_{k-\hat{\tau}_k} &\approx u_{k-\hat{d}_k-1} + \delta\hat{\tau} \frac{u_{k-\hat{d}_k} - u_{k-\hat{d}_k-1}}{k - \hat{d}_k - (k - \hat{d}_k - 1)} \\ &= u_{k-\hat{d}_k-1} + \delta\hat{\tau}(u_{k-\hat{d}_k} - u_{k-\hat{d}_k-1}). \end{aligned} \tag{6}$$

Hence the gradient of $u_{k-\hat{\tau}_k-j}$ with respect to $\delta\hat{\tau}$ can be expressed as

$$\frac{\partial u_{k-\hat{\tau}_k-j}}{\partial \delta\hat{\tau}} = u_{k-\hat{d}_k-j} - u_{k-\hat{d}_k-j-1}. \tag{7}$$

Moreover, the gradient of the output of the neural model with respect to the weights is given by

$$\frac{\partial \hat{y}_k}{\partial w_i^2} = s(x_i), \quad i = 1, \ldots, h \tag{8}$$

and

$$\frac{\partial \hat{y}_k}{\partial w_{ij}^1} = \sum_{i=1}^{h} w_i^2 s'(x_i) \Big(\sum_{j=1}^{n_a} w_{ij}^1 \frac{\partial \hat{y}_{k-j}}{\partial w_{ij}^1} + \hat{y}_{k-j} \Big), \quad j = 1, \ldots, n_a, \tag{9}$$

as well as

$$\frac{\partial \hat{y}_k}{\partial w_{ij}^1} = \sum_{i=1}^{h} w_i^2 s'(x_i) \Big(\sum_{j=1}^{n_a} w_{ij}^1 \frac{\partial \hat{y}_{k-j}}{\partial w_{ij}^1} + u_{k-\hat{\tau}_k-j} \Big), \quad j = n_a + 1, \ldots, n_b. \tag{10}$$

Define the index

$$Q = 0.5e_k^2 = 0.5(y_k - \hat{y}_k)^2, \tag{11}$$

and the parameter matrices, i.e.

$$\begin{aligned} \theta &= [w_1^2, \ldots, w_h^2, \delta\hat{\tau}]^T, \\ \omega &= [w_{ij}^1]_{(h)\times(n_a+n_b+q)}^T. \end{aligned} \tag{12}$$

Then the estimates of these matrices will respectively be

$$\begin{aligned} \theta_k &= \theta_{k-1} - \lambda_1 \frac{\partial Q}{\partial \theta_{k-1}}, \\ \omega_k &= \omega_{k-1} - \lambda_2 \frac{\partial Q}{\partial \omega_{k-1}}, \end{aligned} \tag{13}$$

where $\lambda_i > 0$, $i = 1, 2$ are the optimizing step-sizes. If a second-order optimization algorithm, e.g. a modified

Levenberg-Marquardt method, is applied, the update of matrices θ and ω becomes

$$\psi(k) = \psi(k-1) - \lambda[H_e + \alpha I]^{-1}\frac{\partial Q}{\partial \psi} + \beta\big(\psi(k-1) - \psi(k-2)\big), \tag{14}$$

where Ψ is a generalized parameter of the neural network, H_e is the Hessian matrix, i.e.

$$H_e = \Big(\frac{\partial \hat{y}_k}{\partial \psi}\Big)\Big(\frac{\partial \hat{y}_k}{\partial \psi}\Big)^T_{\Psi=\theta} = \Big(\frac{\partial \hat{y}_k}{\partial \theta}\Big)\Big(\frac{\partial \hat{y}_k}{\partial \theta}\Big)^T,$$

or

$$H_e = \Big(\frac{\partial \hat{y}_k}{\partial \psi}\Big)\Big(\frac{\partial \hat{y}_k}{\partial \psi}\Big)^T_{\Psi=\omega} = \Big(\frac{\partial \hat{y}_k}{\partial \omega}\Big)\Big(\frac{\partial \hat{y}_k}{\partial \omega}\Big)^T,$$

$\alpha > 0$ is the adjustable factor within $(0, \infty)$. At the beginning, α is set as a rather large value to ensure the positive definiteness of the approximation to the Hessian. In this case, the algorithm becomes the steepest descent method. Then α should be decreased towards zero at each successful iteration. If α becomes zero, the Gauss-Newton algorithm is obtained. Moreover, α has the stabilization capability provided that the algorithm converges to a saddle point. In this situation the Hessian matrix approaches zero, and $\alpha > 0$ will improve the numerical stability. In order to increase the possibility to escape from local minima, a momentum term is embedded into this algorithm and $\beta > 0$ is the momentum factor.

When the estimated $\delta\hat{\tau}_k$ is obtained, both the fractional and integer parts will be updated in accordance with (Lim and Macleod, 1995):

$$\begin{cases} \hat{d}_{k+1} = \hat{d}_k - 1, \\ \delta\hat{\tau}_{k+1} = \delta\hat{\tau}_k + 1 \end{cases} \tag{15a}$$

for $\delta\hat{\tau}_k \in (-\infty, k+\eta]$, or

$$\begin{cases} \hat{d}_{k+1} = \hat{d}_k + 1, \\ \delta\hat{\tau}_{k+1} = \delta\hat{\tau}_k - 1 \end{cases} \tag{15b}$$

for $\delta\hat{\tau}_k \in [k+1+\eta, \infty)$, and

$$\begin{cases} \hat{d}_{k+1} = \hat{d}_k, \\ \delta\hat{\tau}_{k+1} = \delta\hat{\tau}_k \end{cases} \tag{16}$$

for $\delta\hat{\tau}_k \in (k+\eta, k+1+\eta)$, where $0 < \eta < 1$ is a very small number. Since the indirect time-delay estimation is a procedure of on-line nonlinear programming, a high computational load as involved.

3. Direct Estimation Method

The procedure of indirect time-delay estimation illustrated in the foregoing section is considered as a problem of nonlinear programming. In this section, the so-called direct time-delay estimation approach will be proposed. A dynamic neural network will be constructed directly for time-delay estimation. The performance of the estimator depends on the specification of the weights V, the orders of the inputs and the number of the hidden nodes of the network. In this section, time-delay estimation is formulated as a procedure of system identification. Assume that the time delay can be separated as integer and fractional parts as well, i.e.

$$\hat{\tau}_k = \hat{d}_k + \delta\hat{\tau}_k, \tag{17}$$

where $\hat{d}_k$ is the integer part of the time delay whilst $\delta\hat{\tau}_k$ denotes the fractional part of the time delay.

In order to estimate the time delay, the estimator of the fractional part of the time-delay is proposed as follows:

$$\delta\hat{\tau}_k = g(V, I_k), \tag{18}$$

where $g(\cdot) \in C^2$ implements the mapping g: $\mathbb{R}^q \to \mathbb{R}$, where $q = q_1 + q_2, q_1$ and q_2 are respectively the orders of the sequence of the fractional part of the time delay $\{\delta\hat{\tau}k\}$ and the sequence of the differences between the system and model outputs $\{e_k\}$; V is the weight matrix, and I_k is the input vector of the time-delay neural estimator of the following form:

$$I_k = \big[\delta\hat{\tau}_{k-1},\ \ldots,\ \delta\hat{\tau}_{k-q_1},\ e_{k-1},\ \ldots,\ e_{k-q_2}\big]^T,$$

where $e_k = y_k - \hat{y}_k$. Formula (18) can be realized using a neural network, i.e.

$$\delta\hat{\tau}_k = \sum_{i=1}^{H} v_i^2 s(z_i) = \sum_{i=1}^{H} v_i^2 s\Big(\sum_{j=1}^{q} v_{ij}^1 I_{k-j}\Big), \tag{19}$$

where H is the number of the hidden neurons of the neural estimator, v_i^2 represents the weight connecting the output of the i-th hidden neuron with the output of the neural network, whilst v_{ij}^1 denotes the weight assigned to the connection between the j-th input of the network and the input of the i-th hidden neuron.

To determine the weights of the neural network which is used for the modelling of the fractional time delay, the derivatives of $\delta\hat{\tau}$ with respect to v_i^2 and v_{ij}^1 are respectively calculated by

$$\frac{\partial \delta\hat{\tau}_k}{\partial v_i^2} = s(z_i), \quad i = 1, \ldots, H, \tag{20}$$

and

$$\frac{\partial \delta\hat{\tau}_k}{\partial v_{ij}^1} = \sum_{i=1}^{H} v_i^2 s'(z_i)\Big(I_{k-j} + \sum_{j=1}^{q_1} v_{ij}^1 \frac{\partial \delta\hat{\tau}_{k-j}}{\partial v_{ij}}\Big), \quad j = 1, \ldots, q. \tag{21}$$

The gradient of $\hat{y}_k$ with respect to the weights of the neural network is determined based on (8)–(10). The weights are adjusted using the modified Levenberg-Marquadt method expressed in (14). The corresponding parameter matrices of the neural network are

$$\theta = [v_1^2, \ldots, v_H^2, \delta\hat{\tau}]^T, \quad \omega = [v_{ij}^1]_{H\times q}^T.$$

Then, based on the estimated result of the fractional part of the time delay, both the integer and fractional parts of the time-delay are adjusted separately using (15a), (15b) and (16).

Usually, the orders of the input variables are specified based on empirical knowledge, and then the number of the hidden nodes of the neural network depends on the criterion minimized with respect to the weight matrix V_H which denotes the weight matrix related to the number of the hidden neurons of the neural estimator, i.e.

$$J(H, V_H) = \sum_{t=1}^{N} \left[y(t) - \hat{y}(t, \hat{d} + \delta\hat{\tau})\right]^2. \quad (22)$$

Define

$$J(H) = \min_{V_H} \sum_{t=1}^{N} \left[y(t) - \hat{y}(t, \hat{d} + g(V_H, I))\right]^2, \quad (23)$$

where N is the number of epochs for optimization. Based on the approximation theory of multilayer feedforward neural networks (Cybenko, 1989), it is known that there will be

$$J(H) \geq J(H+1). \quad (24)$$

Theoretically, while increasing H, there exists an optimal number of the hidden neurons of the time-delay neural estimator, i.e. $H = H^*$, which may lead to

$$J(H-1) \geq J(H^*) \approx J(H^*+1). \quad (25)$$

In the following, the comparison of the on-line computational costs between the indirect and direct time-delay estimation schemes will be given. In order to simplify the comparison, it is assumed that only the steepest descent algorithm is applied to the training procedure of neural networks, as well as the time-delay estimation procedure. Tables 1 and 2 demonstrate the computational operations required for both the approaches.

Table 1. On-line computational load of the indirect approach.

Multiplications	$6(n_a + n_b) + 5h + 4$
Additions	$5(n_a + n_b) + 5h + 3$
Nonlinear computations	$2h + 1$

Table 2. On-line computational load of the direct approach.

Multiplications	$6(n_a + n_b) + 6h + 9$ $+(h+6)(q_1 + q_2 + H)$
Additions	$5(n_a + n_b) + 6h + 3$ $+(h+2)(q_1 + q_2 + H)$
Nonlinear computations	$2h + 2H + 1$

It can be seen that the on-line computational load of the direct time-delay estimation will certainly be heavier than that of the indirect method. However, the direct neural time-delay estimator can be trained either on-line or off-line. Then the training procedure can be stopped if the neural network has been trained well. In this case, the estimator can be implemented with much fewer on-line computational efforts and can be used for fast processes as well.

4. Numerical Example

A numerical example will be used to show the performances of the proposed time-delay estimation approaches. Suppose that the nonlinear process with time delay is

$$y_k = \frac{y_{k-1} + 0.01}{1 + y_{k-1}^2 + y_{k-2}^2} + 0.5u_{k-\tau},$$

where the time delay constitutes a continuous time-varying function of the form

$$\tau = \begin{cases} 0.005t + 2, & t < 300, \\ 17, & 300 \leq t < 400, \\ 17 - 0.005(t - 400), & t \geq 400. \end{cases}$$

Suppose that the sampling period is 0.1 s and a neural network with the SISO architecture and five hidden nodes is used for system modelling. The input signal of the form

$$u_k = \sin(k/20) + 2\sin(k/10) + \cos(k/5) + \cos(k/2) + 2\sin(k)$$

is used to stimulate the process. Both the direct and indirect algorithms for time-delay estimation are applied to the modelling procedure. Figure 1 illustrates the relation between the number of the hidden nodes of the neural network based time-delay estimator and the accuracy of the modelling approximation. It is shown that an increase in the number of hidden nodes will decrease the mean squared error of the modelling approximation. However, when the number of hidden nodes is greater than a certain

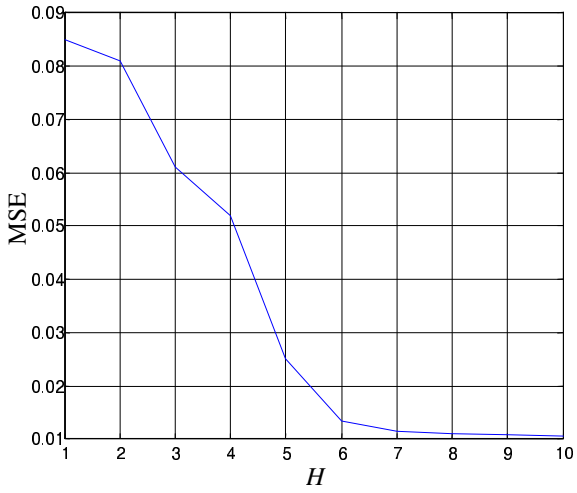

Fig. 1. Relation between the MSE and the number of hidden nodes.

value (here this value is 7), the mean squared error will not be reduced obviously. In this case, we chose seven hidden nodes to construct the neural time-delay estimator. In this network, the input vector is of the form

$$I = \left[\delta\hat{\tau}_{k-1},\ \delta\hat{\tau}_{k-2},\ \delta\hat{\tau}_{k-3},\ e_{k-1}, e_{k-2},\ e_{k-3}, 1\right]^T.$$

Figure 2 shows the result of time-delay estimation using the direct method. For the indirect time-delay estimation method, the parameters for the Levenberg-Marquadt algorithm are chosen as $\lambda = 0.025$ and $\beta = 0.75$. The initial value of the adjustable factor is $\alpha = 0.15$. The corresponding time-delay estimation result is illustrated in Fig. 3. The estimation errors are shown in Tables 3 and 4.

Table 3. Estimation errors of the indirect method.

Mean squared error	0.9715
Maximum error	4.4950

Table 4. Estimation errors of the direct method.

Mean squared error	0.1077
Maximum error	3.8622

From the presented results, it follows that the direct approach for time-delay estimation is better than the indirect method. Obviously, the direct method results in a much smaller residual for the estimate. The on-line computational cost for the direct method is, however, much more expensive than that of the indirect approach. If the neural time-delay estimator is trained well, then the training mechanism can be terminated. In this case, the well-trained neural estimator can be implemented with high speed and substantially reduced computational cost.

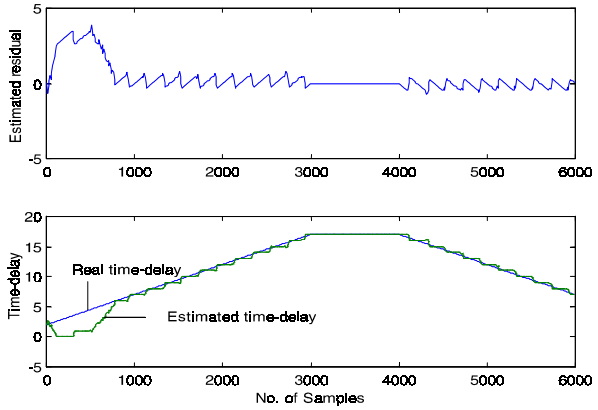

Fig. 2. Time-delay estimation using the direct method.

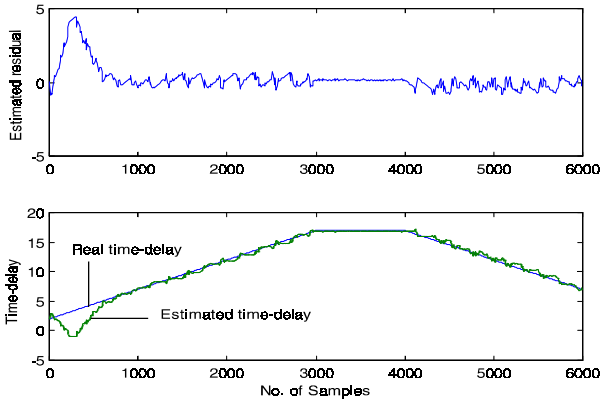

Fig. 3. Time-delay estimation using the indirect method.

5. Conclusions

In this paper, the neural network based direct and indirect time-delay estimation methods for nonlinear dynamic systems with time-varying time delays are proposed. The proposed indirect approach, based upon a neural model with time delay to simulate a given nonlinear dynamic system with time delay, can be considered as an on-line nonlinear programming procedure. On the other hand, the direct method for time-delay estimation employs a neural network based time-delay estimator to identify the time delay directly. For the computational cost, the direct method is obviously less efficient than the indirect method if on-line training is implemented. However, if the training procedure is finished, the well-trained estimator will have a much lighter computational load than the indirect method since it does not require any on-line optimizing computation in this case. In order to simplify the procedure of time-delay estimation, the technique of splitting the time delay into the integer and fractional parts was applied. The presented numerical example shows that both

of the proposed methods can be used to estimate time delays for nonlinear systems. The direct method produced, however, better estimation results.

Acknowledgement

This research was partly supported by research funds granted by the Natural Science Foundation of China (NSFC grant no. 50265001), Natural Science Foundation of Guangxi, China (GXNSF grant no. 0009005) and the Fund of Key Research Projects granted by the Ministry of Education of China. The author would also like to express his thanks to the anonymous reviewer for his valuable comments and suggestions.

References

Balestrino A., Verona F. and Landi A. (1988): *On-line process estimation by ANNs and Smith controller design*. — IEE Proc., Pt. D. Contr. Theory Appl., Vol. 145, No. 2, pp. 231–235.

Ching P.C., So H.C. and Wu S.Q. (1999): *On wavelet denoising and its applications to time delay estimation*. — IEEE Trans. Signal Process., Vol. 47, No. 10, pp. 2879–2882.

Cybenko G. (1989): *Approximation by superposition of a sigmoidal function*. — Math. Contr. Signal Syst., Vol. 2, No. 4, pp. 303–314.

Lim T.J. and Macleod M.D. (1995): *Adaptive algorithm for joint time-delay estimation and IIR filtering*. — IEEE Trans. Signal Process., Vol. 43, No. 4, pp. 841–851.

Reed F., Feintuch P. and Bershad N. (1981): *Time-delay estimation using the LMS adaptive filter-static behavior; dynamic behavior*. — IEEE Trans. Acoust. Speech Signal Process., Vol. 29, No. 3, pp. 561–576.

Shor G. and Messer H. (1997): *Performance evaluation of time-delay estimation in non-Gaussian conditions*. — Proc. IEEE Signal Processing Workshop *Higher-Order Statistics*, Banff, Canada, pp. 20–24.

Teng F.C. and Sirisena H.R. (1988): *Self-tuning PID controllers for dead time processes*. — IEEE Trans. Ind. Electron., Vol. 35, No. 1, pp. 119–125.

Received: 10 February 2003
Revised: 16 April 2003
Re-revised: 26 January 2004