On-line Identification Based on Neural Networks using of Levenberg-Marquardt Method and Back-propagation Algorithm

PIVOŇKA PETR, DOHNAL JIŘÍ
Department of Control and Instrumentation
Brno University of Technology, Faculty of Electrical Engineering and Communication
Božetěchova 2, 612 66 Brno
CZECH REPUBLIC

Abstract: - In order to be able to control a system by means of an automatically adaptable controller, we have to adjust its parameters to the changes taking place in the system. For this purpose a number of methods have been developed. A frequently used identification method is the least-squares method, but we are limited by the choice of the suitable sampling period. The neural network seems to be a desirable solution because of its adaptation characteristics. Among the most used learning algorithms belong the Levenberg- Marquardt and back-propagation.

Key-Words: Levenberg- Marquardt, Back-propagation, Neural network, Training set

1 Introduction

Process identification is of great significance in adaptive control. Properly identified parameters of a dynamic system make it possible to design a suitable controller. The most frequently used method is the least squares method. Its advantage is the fast determination of the sought-for parameters, however, we are limited by the choice of the suitable sampling period T. Its limit value, when controlling a real system via A/D and D/A converters, is approximately a tenth of the dominant time constant of the process. Making sampling period even shorter leads to unrealistic estimates of the state. If the parameters identified in this way were used for calculating the settings of the adaptive controller this would result in a worse controlling action than when employing a traditional PID controller (or discrete PID with derivative component filtering), in particular when the controlling action is to rectify a fault. Such situations require that different methods of continuous identification are applied. One of the possible approaches is employing neural networks [1].

2 Neural Approach

A formal neuron has n real inputs $x_1,\ldots,x_n$. The inputs are evaluated using corresponding real synaptic weights $w_1,\ldots,w_n$ defining their „throughput“. A neuron transforms input data into output data based on the transfer function. Individual neurons can be arranged to form a neural network – the neurons are interconnected so that a neuron output is an input to multiple neurons. The number of neurons and their interconnections in the network determine the neural network architecture. The so-called feed-forward networks are used in control technology to implement controllers or, for example, to identify process parameters. In a linear model of a process it seems beneficial to use only a single neuron with a linear transfer function for identification.

The advantageous and distinguishing feature of neural networks is their ability to learn. The network in the adaptive mode abstracts and generalizes the function character in the process of learning from training patterns. The learning algorithm is an optimization method capable of finding weight coefficients and thresholds for a given neural network and a training set. There are a number of learning algorithms. Those that are used most frequently are the back-propagation algorithm and the Marquardt-Levenberg algorithm.

2.1 Back-propagation Algorithm (BP)

This algorithm is based on minimizing the error of the neural network output compared to the required output. The required function is specified by the training set (a sequence of input / required network output pairs). The error of network E relative to the training set is defined as the sum of the partial errors of network E_k relative to the individual training patterns and depends on network configuration w

$$E = \sum_{k=1}^{p} E_k \qquad (1)$$

The partial error of network E_k relative to the k-training pattern is proportional to the product of the squares of the deviations of the actual values of network y_j outputs for the input of k-training pattern x_k from the corresponding required values of outputs for pattern d_j

$$E_k = \frac{1}{2}\sum_{j\in Y}\left(y_j - d_{kj}\right)^2 \qquad (2)$$

where Y is the set of output neurons

The adaptation of weights (in time $t = 0$ the configuration weights $w^{(0)}$ are set randomly close to zero) takes place in discreet time steps corresponding to the training cycles. The new configuration in time $t > 0$ is calculated as follows

$$w_{ji}(k) = w_{ji}(k-1) - a \frac{\partial E}{\partial w_{ji}} \qquad (3)$$

where $0 < \alpha < 1$ is the speed of learning [2], [5].

The speed of training is dependent on the set constant α. If a low value is set, the network weights react very slowly. On the contrary, high values cause divergence - the algorithm fails. Therefore the parameter α is set experimentally.

If the neural network is to be used as a model in an adaptive system, for real industrial process control, the divergence must be prevented. In order to avoid it, the algorithm is often modified, the parameter α can be adjusted in the progress of training in dependence on the network error E. The neural network is submitted the training set patterns. The instantaneous error $E(w(k))$, $\partial E(w(k))/\partial w(k)$, is determined, and a new weight configuration $w(k+1)$, then $E(w(k+1))$ are calculated. Now, we have to find out if the network training error was reduced. If

$$E(w(k+1)) < E(w(k)) \qquad (4)$$

is fulfilled, the new configuration of network weights is accepted, the value of parameter α is increased. Otherwise constant α is decreased and configuration $w(k+1)$ is recalculated.

Algorithm can be enhanced including momentum β. This parameter can be adapted in the same way as the training speed α.

2.2 Levenberg-Marquardt Algorithm (LM)

This algorithm is a variant of the Gauss-Newton optimization method. As in back-propagation we look for the minimum of the function

$$E = \frac{1}{2}\sum_{k=1}^{p} e_k^2 = \frac{1}{2}\|\varepsilon\|^2 \qquad (5)$$

where p is a number of available patterns, ε is a error vector (difference between the actual and the desired value of the network output for the individual patterns).
By introducing the Jacobi's matrix

$$J = \begin{pmatrix} \frac{\partial e_1}{\partial w_1} & \cdots & \frac{\partial e_1}{\partial w_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial e_p}{\partial w_1} & \cdots & \frac{\partial e_p}{\partial w_n} \end{pmatrix} \qquad (6)$$

it is possible to estimate the error development using the Taylor polynomial of the 1st degree

$$E(k+1) = \frac{1}{2}\|\varepsilon(k) + J(w(k+1) - w(k))\|^2 \qquad (7)$$

By minimizing using $w(k+1)$ (where w is the weights vector) and introducing parameter λ an equation is obtained for calculating the network weights

$$w(k+1) = w(k) - (J^T J + \lambda I)^{-1} J^T \varepsilon(k) \qquad (8)$$

For a large value of parameter λ the algorithm approaches the gradient method, for small λ it becomes the Newton method.

Parameter λ is modified based on the development of error function E. Should the step cause a reduction of E, we accept it. Otherwise we change parameter λ, reset the original value and recalculate $w(k+1)$ [3].

2.3 Comparison of Methods

In the simple gradient method we proceed in the direction of tangent vector $\partial E/\partial w$ downwards by a α. For a sufficiently small α we obtain configuration $w(k)$ for which the error function is smaller than for the original configuration $w(k-1)$. We therefore proceed by layers $E = \text{const.}$ towards the minimum. Obviously the algorithm converges towards the optimum slowly but the greatest disadvantage seems to be the possibility of stopping at a stationary point – at the location of local minimum E. The network error is not reduced any longer and the optimum configuration w^* is not obtained. There are modifications available for making the back-propagation algorithm more robust (by introducing momentum, learning constant modifications, etc.).
Levenberg-Marquardt algorithm places a quadratic function around the point and proceeds towards its minimum. The sought - for optimum w^* can be established with great accuracy using fewer steps, although at the expense of increased calculation complexity. In one step a much greater number of calculation operations need to be made compared to a single step in back-propagation.

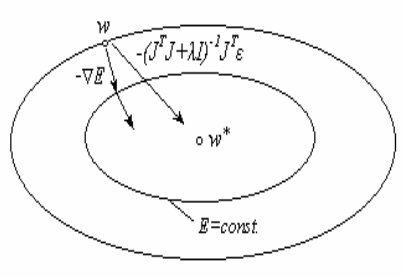

Fig. 1 Method of establishing optimum w* using the LM and BP methods

3 ARX Regression Model

In most cases (and in adaptive control in particular) the parameters that are to be determined are those of a linear regression model ARX which models the system output based on the equation

$$y(k) = -\sum_{i=1}^{n} a_i y(k-i) + \sum_{i=1}^{m} b_i u(k-i) \qquad (9)$$

or

$$A(z^{-1})y = B(z^{-1})u \qquad (10)$$

where u – excitation signal, y – system output.

The equation can also be expressed in the form of vectors

$$y(k) = \Theta^T(k)\varphi(k-1) \qquad (11)$$

where $\Theta = [a_1, a_2, ..., a_n, b_1, b_2, ..., b_m]$ is the vector of the parameters of the investigated model and $\varphi(k\text{-}1)$ is the vector of the data [4].

4 Simulation Experiment

The individual algorithms were applied in the MATLAB software environment. The learning process has been tested on several transfer functions.

Examples

- identification of the transfer function

$$G_A(s) = \frac{2}{(10s+1)(s+1)} \qquad (12)$$

which is expressed in the discreet area for sampling period $T = 0.3$ s as

$$G_A(z) = \frac{0.008081z^{-1} + 0.007239z^{-2}}{1 - 1.71z^{-1} + 0.7189z^{-2}} \qquad (13)$$

After the identification process is finished the transfer function can be drawn up. Figure 3 shows the system and model response for input $u(t) = 1$, figures 4,5 curves of identified parameters in time.

- identification of the transfer function (formula in discreet area, sampling period T = 0.3 s)

$$G_1(z) = \frac{0.009948z^{-1} + 0.008779z^{-2}}{1 - 1.669z^{-1} + 0.6873z^{-2}} \qquad (14)$$

change of system happened in time $t = 50$ s using block switch to

$$G_2(z) = \frac{0.001747z^{-1} + 0.001695z^{-2}}{1 - 1.912z^{-1} + 0.9139z^{-2}} \qquad (15)$$

Identification block scheme is in the figure 2. Figures 6,7 show curves of identified parameters in time.

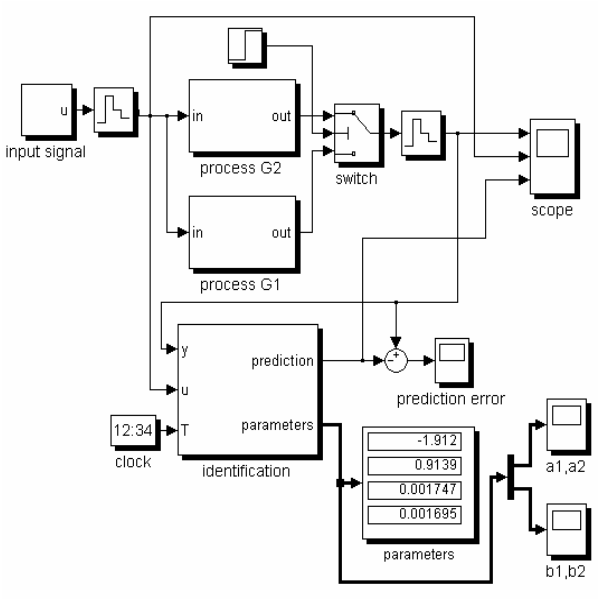

Fig. 2 Scheme of on-line identification of the system

During the experiment, the random signal with Gaussian distribution has been used as an input to the system. A mean value equals to zero and a variance equals to five (a standard normal distribution).

The standard learning algorithm has been extended by the batch learning method. The input signal $u(k)$ and output system response

$$y(k) = \Theta^T x \qquad (16)$$

(where $\Theta^T = [b_1, b_2, a_1, a_2]$ is the vector of the parameters model) are sampled in sampling period T = 0.3 s. A vector

$$x^T = [u(k-1), u(k-2), -y(k-1), -y(k-2)] \qquad (17)$$

is used as a net input during the learning process and according to the current system response $y(k)$ we minimize function performance (1). A training set consists of fifteen training pairs x^T, d. Until the training set is not full, the training patterns have been added to the training set. It is necessary to eliminate out-of-date training patterns and add current data to ensure continuous identification.

The initial parameter set-up for LM algorithm is: $\lambda = 0.001$; for BP than $\alpha = 0.1$ and momentum $\beta = 0.8$.

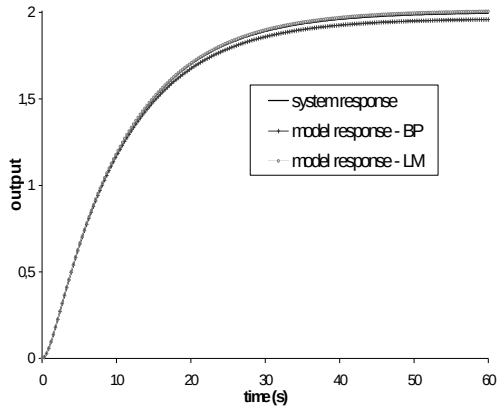

Fig. 3 Identification using the LM and BP methods

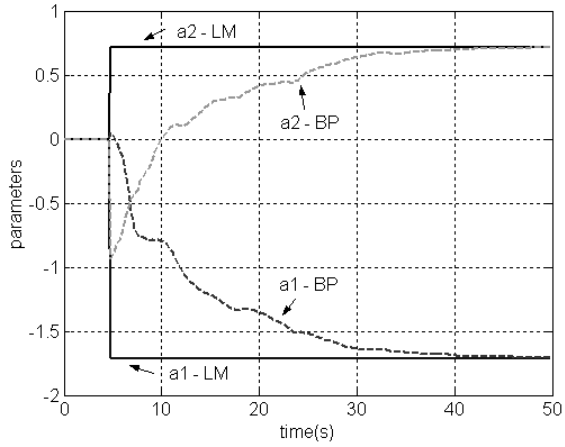

Fig. 4 Curve of parameters (system G_A)

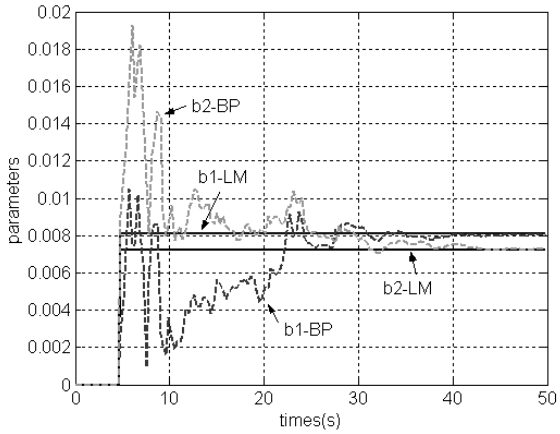

Fig. 5 Curve of parameters (system G_A)

The identification process begins in time t = 4.8 s. Data have been added to the training set in time period t = 0 to 4.8 s. Above mentioned figures present that back-propagation algorithm converges towards the optimum parameters slowly. On the other hand, Levenberg-Marquardt gives better results because coefficients have been gained almost immediately. Because of its speed and accuracy, only algorithm LM was used in further experiments.

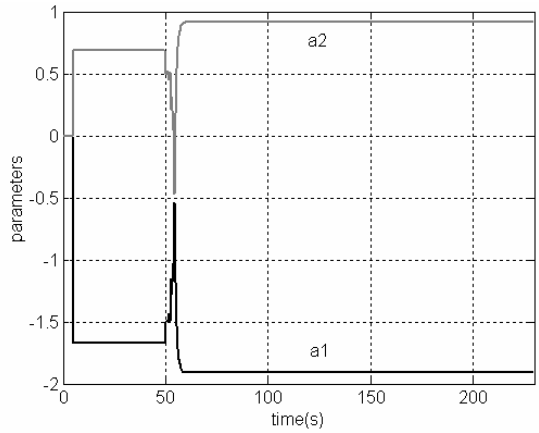

Fig. 6 Curve of parameters a_1, a_2 (LM, system $G_1 \rightarrow G_2$)

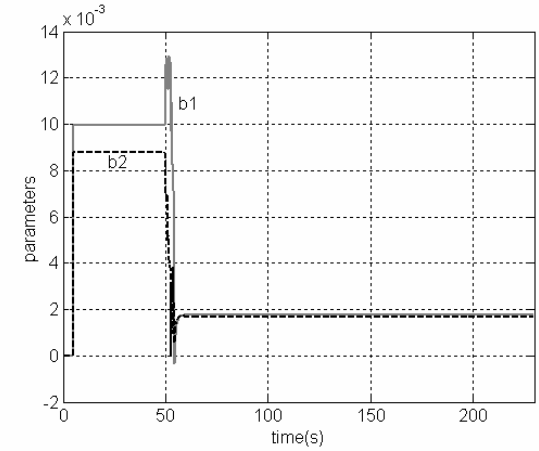

Fig. 7 Curve of parameters b_1, b_2 (LM, system $G_1 \rightarrow G_2$)

5 Physical model identification

For verification of algorithms on physical models communication between the MATLAB/Simulink environment and the programmable logic controller B&R (series 2005) was applied (using the PVI interface). PLC acts as an input/output card only. In synchronous instants the PVI client can read or record into internal variables of PLC mapped at its inputs and outputs.

When working with real systems we can measure data with only limited accuracy. It is due to the signal transition in A/D, D/A converters with the step function of parameters. The calculation accuracy is therefore not in the order of 10 or more valid digits, but a maximum of 4 valid digits. Consequently, the identified parameters are not stabilized, but oscillate, which is undesirable. One of the possible approaches is stopping the quantification in the instant when the prediction model error drops below the set limit.

The system is initiated by signal u_1 (rectangular pulses – amplitude 2,-1; period 5 s), The output of the process is measured in the sampling period 0.1 s. The batch contains 40 patterns. Initial values were $\lambda = 0.001$.

Fig.3 shows the curve of the physical model response (approximate value of time constants $T_1 = 10$ s, $T_2 = 1$ s) and the neural model to the initiation signal.

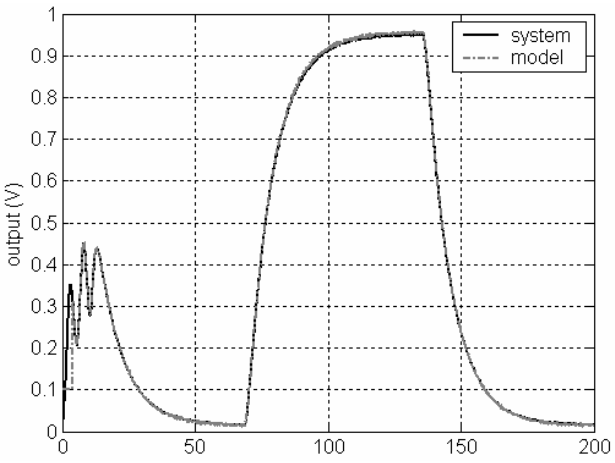

Fig. 8 Physical model identification

It was proven by experiments that an single neuron can be used even in noise-affected systems.

6 Conclusion

Neural networks are a suitable tool for process identification. They facilitate determining the required parameters in situations where the least squares method fails. This happens when the short sampling period is required (mainly for quick disturbance cancellation). Presented figures show that Levenberg-Marquardt algorithm can reach optimum parameters of system transfer function with a great accuracy when using fewer steps. The disadvantage is an increased calculation complexity. It was proved that when an single neuron is used an industrial PLC can identify a noise-affected linear model with sufficient accuracy.

Acknowledgement

The paper has been prepared with the support of the research plan CEZ: MSM 260000013

References

[1] K. Švancara, P. Pivoňka: Identification Based on Neural Networks in adaptive Optimal Controller, In *Procceedings of Zittau Fuzzy Colloquium*, pages 218-223, Zittau 2002.

[2] Šíma, J., Neruda, R.: *Theoretical issue of neural networks*. MATFYZPRESS 1996, in Czech

[3] Hristev, R.M.: *The ANN Book*, GNU Public Licence, 1998

[4] Bobál, V., Böhm, J., Prokop, R., Fessl, J.: *Practical Aspects of Self-Tuning Controllers, Algorithms and Implementation*. Brno, VUTIUM 1999, in Czech

[5] Cichocki, A., Unbehaunen, R., *Neural Networks for Optimization and Signal Processing*, John Wiley&Sons 1993