Real-Time Implementation of Adaptive Optimal Controller Based on a Pseudo-Space Model into PLC

PETR PIVOŇKA, VLASTIMIL LORENC
Department of Automatic Control and Instrumentation
Faculty of Electrical Engineering and Communication, Brno University of Technology
Kolejni 4, 612 00 Brno
CZECH REPUBLIC

Abstract: - This paper describes the design and implementation of adaptive linear optimal controller (LQ) based on a pseudo-space model into PLC. For identification of the controlled system an algorithm based on artificial neural network is used. The method of comfortable implementation of algorithms from the MATLAB/Simulink environment into PLC is shown. In the first step of implementation the algorithm is designed in the simulation environment. The second step contains verification of the designed algorithm during real process control, and finally is the algorithm moved from MATLAB/Simulink into PLC. The basic condition for realization of the described implementation method is the existence of real-time communication between MATLAB/Simulink and PLC. This article shows how real-time communication can be created using PLC B&R. The behavior of the described adaptive LQ algorithm is compared with that of adaptive PSD controller during control of the real system.

Key-Words: - Adaptive controller, Optimal controller, MATLAB, Simulink, PLC, Levenberg-Marquardt, Neural Networks, Self-Tuning

1 Introduction

The purpose of adaptive controllers is to adapt control law parameters of control law to changes of the controlled system. Many types of adaptive controllers are known. In this article the adaptive self-tuning LQ controller is described. The scheme of this controller is separated into the two main parts: identification and controller. In this work, identification based on neural network approach is used and the control algorithm is the linear quadratic optimal controller. Figure 1 shows the architecture of the self-tuning LQ controller where w denotes desired value, u denotes action value and y denotes output of the controlled system.

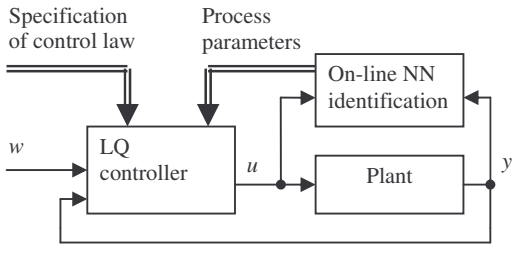

Fig. 1. Architecture of self-tuning LQ controller.

2 On-line Identification

The most important part of self-tuning systems is the on-line identification algorithm. Widely used method for identification of systems is recursive least squares method (RLS). Instead of RLS, the identification method based on neural network can be used. A very fast algorithm for training neural networks is the Levenberg-Marquardt (LM) algorithm. The main idea of on-line identification is that according to the measured input to the identified system $u(t)$ and the corresponding system output $y(t)$ we are able to find the vector of system parameters Θ.

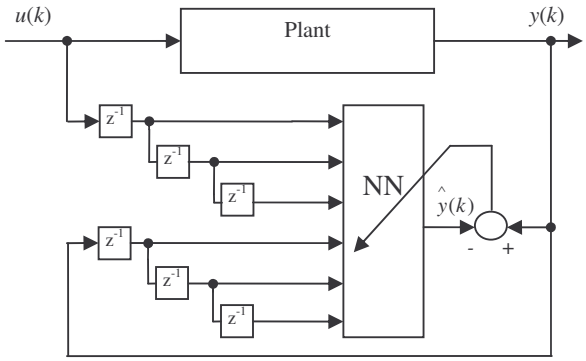

Fig. 2. The principle of identification of system using neural network.

For computing of the identified system output we can use the linear ARX model

$$F_M(z^{-1}) = \frac{b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}. \quad (1)$$

The ARX model can be written in vector form as follows

$$\hat{y}(k) = \varphi^T(k)\theta(k) \quad (2)$$

where

$$\varphi(k) = \left[u(k-1) \ \ldots \ u(k-1-m) \ -y(k-1) \ \ldots \ -y(k-1-n)\right]^T \quad (3)$$

is the vector of measured inputs and outputs and

$$\theta(k) = \left[b_1(k) \ \ldots \ b_m(k) \ a_1(k) \ \ldots \ a_n(k)\right]^T \quad (4)$$

is the vector of estimated system parameters and k denotes discrete time.

As it has been mentioned above, the Levenberg–Marquardt method can be used for training of the neural network. The new vector of parameters is in each step given by next equation.

$$\theta(k+1) = \theta(k) - \left(JJ^T + \lambda I\right)^{-1} J^T \varepsilon(k) \quad (5)$$

where J is Jacobian matrix in the form

$$J = \begin{pmatrix} \frac{\partial \varepsilon_1}{\partial w_1} & \cdots & \frac{\partial \varepsilon_1}{\partial w_j} \\ \vdots & \ddots & \vdots \\ \frac{\partial \varepsilon_p}{\partial w_1} & \cdots & \frac{\partial \varepsilon_p}{\partial w_j} \end{pmatrix}, \quad (6)$$

and ε is the vector of errors between the output of the system and the output of the model for all training patterns. The parameter p denotes the number of training patterns and j denotes the number of estimated parameters.

The reason why identification based on neural network approach was used instead of RLS is that using a neural network we can set a shorter sampling period [7]. Then the controller is able to reject errors faster.

3 Linear LQ Controller

The action value of LQ controller is solved according to the identified model (1) and according to the minimization of quadratic performance (7) in each step.

Quadratic performance is defined by

$$J = x^T(k)Qx(k) + \sum_{k=k_0+1}^{k_0+T} q_y\left(w(k) - y(k)\right)^2 + q_u\left(u(k) - u0(k)\right)^2 \quad (7)$$

where $w(k)$ denotes the desired value, $y(k)$ denotes system output and $u(k)$ is action value. Parameter $u0(k)$ is a signal equal to the desired value, which is used for elimination of offset. Parameters q_y (q_u) denote weights for output (action) value, k_0 denotes the first step while the minimization is used and $x^T(k)Qx(k)$ denotes the minimum in the last step k_0+T. When we work with the pseudo state matrix $S = [S_u, S_x, S_w, S_{u0}]$ defined by

$$S_u = \left[1 \ 0 \ 0 \ b_0 \ 0 \ \cdots \ 0\right]^T$$

$$S_w = \left[0 \ 0 \ 0 \ 0 \ \cdots \ 1 \ 0\right]^T \quad (8)$$

$$S_{u0} = \left[0 \ 0 \ 0 \ 0 \ 0 \ \cdots \ 1\right]^T$$

$$S_x = \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & 0 & \cdots & 0 \\ b_1 & \ldots & b_m & a_1 & \cdots & a_n \\ 0 & \cdots & 0 & 1 & 0 & 0 \\ \vdots & \ddots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 \end{bmatrix} \quad (9)$$

and when we introduce the pseudo state vector $z^T(k) = [x(k), w(k), u0(k)]$ and $x^T = [u(k), \ x(k\text{-}1), w(k), u0(k)] = S(k)z(k\text{-}1)$ we can rewrite the quadratic performance to the more suitable form [1, 2]

$$J = \sum_{k=k_0+1}^{k_0+T} z^T(k)Qz(k). \quad (10)$$

Using a nonstandard state vector we can build universal quadratic performance. For example, for standard penalization according to equation (7) matrix Q is defined as follows

$$Q = \begin{bmatrix} q_u & 0 & 0 & 0 & 0 & 0 & 0 & -q_u \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_y & 0 & 0 & -q_y & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -q_y & 0 & 0 & q_y & 0 \\ -q_u & 0 & 0 & 0 & 0 & 0 & 0 & q_u \end{bmatrix}. \quad (11)$$

The method for minimization in each step is known [2, 1] and is described by next equations.

$$J(k_0+T) = z^T(k_0+T-1)S^T QSz(k_0+T-1) = \\ = z^T(k_0+T-1)Hz(k_0+T-1) \quad (12)$$

The minimum of (12) is given for derivation by $u(k_0+T)$ equal to zero

$$J^*(k_0+T) = \\ = z_m^T(k_0+T-1)\left(H_{\varphi\varphi} - H_{u\varphi}^T H_{uu}^{-1} H_{u\varphi}\right) z_m^T(k_0+T-1) \quad (13)$$

where $z_m^T(k_0+T-1)$ is $z^T(k_0+T-1)$ without $u(k_0+T-1)$ and

$$H = \begin{bmatrix} H_{uu} & H_{u\varphi} \\ H_{\varphi u} & H_{\varphi\varphi} \end{bmatrix} \quad (14)$$

For next minimization step we can write

$$J^*(k_0+T-1) = \\ = z_m^T(k_0+T-1)H^* z_m^T(k_0+T-1) + \\ + z^T(k_0+T-1)Qz(k_0+T-1) \quad (15)$$

Where matrix $H^* = H_{xx} - H^T{}_{ux} - H^{-1}{}_{uu}H_{ux}$ is defined at previous step.

The algorithm given above is unsuitable for practical calculations. The reason is that matrix $H_{\varphi\varphi}$ ceased to be positively definite (semi-definite). This means that the algorithm can be unstable. The solution is LD-FIL decomposition [2]. When we work with matrix G instead of H where $H=GDG^T$ and G is a lower triangular matrix and D is a diagonal matrix, we can rewrite quadratic performance to the form

$$\left\| G\left[u(k), \varphi(k-1), w(k), u0(k)\right]^T \right\|^2 \quad (16)$$

Minimization of equation (15) at step k is

$$G_{uu}u(k) + G_{u\varphi}\varphi(k-1) = 0 \quad (17)$$

Where G_{uu} and $G_{u\varphi}$ are sub-matrices of G. Finally, the action value is in each step computed as follows

$$u(k) = -G_{uu}^{-1} G_{u\varphi}\varphi(k-1).$$

To summarize in the table below, we can see the recursive algorithm for computing of the LQ controller.

Step	Equation and Notes
1.	$H^* = H_{xx} - H^T{}_{ux} - H^{-1}{}_{uu}H_{ux}$ recursively solves lost function
2.	GDG^T LD-FIL decomposition
3.	$u(k) = -G_{uu}^{-1}G_{ux}x(k-1)$ solves action value

Table 1. Iteration algorithm of LQ controller.

4 Implementation of designed control algorithm into PLC

4.1 Introduction

The described control algorithm was created in the MATLAB/Simulink environment and then was implemented to the programmable logic controller (PLC) B&R. Design and implementation were done in three steps.

In the first step the algorithm was designed and tested only using MATLAB/Simulink. In the second step MATLAB/Simulink was interconnected with the PLC. This step allows verification properties of the designed control algorithm during control of the real system. It is important to notice that in this case we have the same possibilities for verification of the control algorithm behaviour, and we can very easily change its parameters and monitor the quality of real process regulation. In the third step the algorithm is moved from MATLAB/Simulink to the PLC. For fast and comfortable accomplishment of this step it is necessary to use a language which can be used in MATLAB/Simulink and in the PLC environment as well. In our case the language ANSI C was used.

This implementation method is faster than in the case when the control algorithm is developed directly in the environment of PLC. The reason is that MATLAB/Simulink allows more comfortable methods and tools for design, debugging and testing of algorithms. Another major advantage is that the PLC is not necessary for the first step of implementation.

4.2 Connection between MATLAB/Simulink and PLC B&R

For realization of the described implementation scheme it is imperative to establish connection between MATLAB/Simulink and PLC. This communication was carried out using Process Visualization Interface (PVI). The PVI is a file of tools and libraries for the operating system MS Windows and is provided as a component of the

B&R AUTOMATION NET software. The PVI procures a communication interface between PLC B&R and applications on PC with MS Windows [6]. As a physical network layer Ethernet or RS-232 link can be used.

The most important part of PVI is the PVI Manager. This utility provides network connection between PC and PLC. Using PVI libraries and some standard development environment (e.g. MS Visual Studio or Borland Delphi) it is possible to create user application which can, by force of PVI Manager, communicate with PLC.

In our case, the communication client was created as the s-function for MATLAB/Simulink. The s-function represents a real controlled system. That means that the s-function realizes transfer of the action value computed in MATLAB to the PLC, and from the PLC values representing the output of the real controlled system are transferred. In the PLC, a task is running which only resends these values into and from the real process using D/A and A/D converters. Next figure shows the scheme of the described communication.

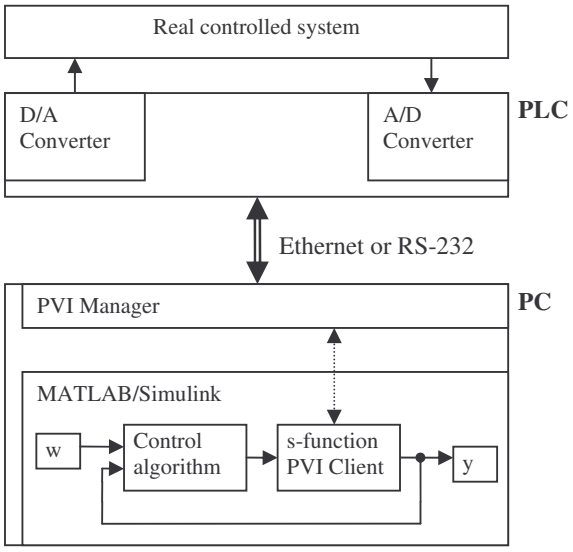

Fig. 3. The scheme of communication between MATLAB/Simulink and PLC B&R.

5 Real Process Control Results

The described LQ controller was implemented into the PLC B&R. Used as the controlled system was the laboratory model with the transfer function $F(s) = \frac{1}{(10s+1)(s+1)^2}$. In the figure 3 we can see the real process response with the described LQ control algorithm. Figure 4 shows real the process response with the self-tuning PSD controller based on the modified Ziegler Nichols method [2, 4]. In both cases identification with the neural network approach and the same setup and initial conditions were used. The sampling period was set to $T_s = 1s$.

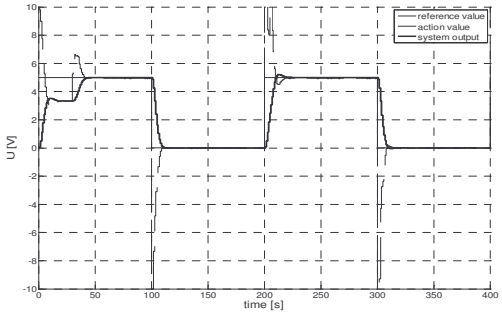

Fig. 4. Real process control using LQ controller (q_u=0.1, q_y = 1).

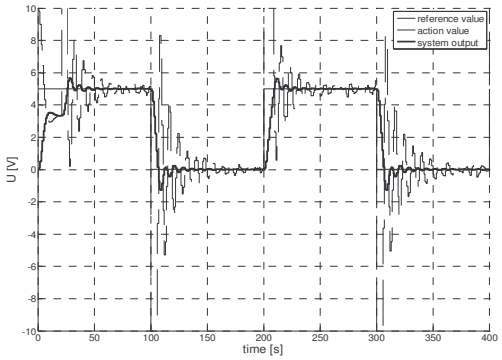

Fig. 5. Real process control using self-tuning PSD controller

In comparison of two previous figures we can see that using LQ controller was reach better results notably in case of changes of action value.

As it was shown above we can modify the criterion of LQ controller by changing of two parameters q_u and q_y. The parameter q_u influences the changes of action value and q_y is standard penalization of control error. In fact the criterion is dependent on relation of these two parameters. If we set q_y = 1, then we have only one parameter q_u for changing of criterion. The responses of LQ controller are optimal in regard of the criterion but it doesn't mean that they are optimal from the user's viewpoint. The user specifications can be very easily included to the control law using parameter q_u.

6 Conclusion

It was described design of adaptive LQ controller. This algorithm solves one iteration in each step. That means the time for computing is short and this algorithm is suitable for implementation in industrial controller. As it was shown above the properties of control law can be changed using one parameter q_u. This algorithm was implemented to the industrial controller applying of third-step implementation scheme, which allows control of real processes directly from MATLAB/Simulink environment. By using this method can be saved time for development of the control algorithm, because MATLAB/Simulink is able to give more comfortable possibilities for design and debugging of algorithm in comparison with the environment of PLC. The communication between MATLAB/Simulink and PLC B&R using PVI was described.

Acknowledgement:

The paper has been prepared as a part of the solution of Czech Science Foundation GAČR project No. 102/06/1132 Soft Computing in Control and by the Czech Ministry of Education in the frame of MSM MSM0021630529 Intelligent Systems in Automation.

References:

[1] Švancara, K., Adaptive Optimal Controller with Identification Based on Neural Networks, *PhD thesis*, FEEC BUT, Brno 2004.

[2] Bobál V., Böhm J., *Practical Aspects of Self-Tuning Controllers, Algorithms and Implementation*, VUTIUM, 1999, in Czech, ISBN 80-214-1299-2.

[3] Švancara K., Pivonka P., Closed Loop On-line Identification Based on Neural Networks in Adaptive Optimal Controller, *In Proceedings of 10th Zittau Fuzzy Colloquium*, Zittau (Germany), 2002.

[4] Kořínek, V., *Comunication MATLAB-Ethernet. Diploma thesis in Czech*, FEEC BUT, Brno 2003.

[5] Lorenc V., Self-Tuning Controllers in MATLAB - B&R Environment, *Diploma thesis in Czech*, FEEC BUT, Brno 2006.

[6] *B&R Documentation. Automation Net/PVI Online Help*, AS240Ee, 1997-2007, http://www.br-automation.com

[7] Veleba, V., The advantages of identification based on neural network approach, *In Proceedings of the 11th conference Student EEICT,* BRNO 2005.