ELSEVIER
IFAC
PUBLICATIONS
www.elsevier.com/locate/ifac

SELF-TUNING CONTROL: LABORATORY REAL - TIME EDUCATION

Vladimír Bobál, Petr Chalupa, Marek Kubalčík, Petr Dostál

Department of Control Theory, Institute of Information Technologies,
Tomas Bata University in Zlín, Nám. TGM 275, 762 72 Zlín, Czech Republic,
tel.: +420 57 603 3217, E-mail: bobal@ft.utb.cz

Abstract: The combination of the automatic control theory courses and practical implementation of the designed controller algorithms in real-time conditions is very important for training of the control engineers. This contribution presents a MATLAB-Toolbox for design, simulation verification and especially real-time implementation of single input - single output (SISO) discrete self-tuning controllers. The proposed adaptive controllers what are included into a Toolbox can be divided into three groups. The first group covers PID adaptive algorithms using traditional Ziegler-Nichols method for the setting of the controller parameters, the second group of described controllers is based on the pole placement design and the third group contains the controllers derived on the other approaches (dead-beat, minimum variance etc.). The controllers are implemented as an encapsulated SIMULINK blocks and thus allows users simple integration into existing SIMULINK schemas. The process of developing real-time applications using MATLAB Real-Time Workshop and several control courses is also presented. The MATLAB-Toolbox is very successfully used in Adaptive Control Course in education practice for design and verification of self-tuning control systems in real-time. It is suitable for design and verification of the industrial controllers, too. The MATLAB-Toolbox is available free of charge at Internet site - **http://www.utb.cz/stctool/**. *Copyright © 2003 IFAC.*

Key words: Self-tuning control; ARX model; Recursive least squares; PID control; Pole assignment; Real-time control; MATLAB–Toolbox.

1. INTRODUCTION

Adaptive control methods have been developing significantly over the past four decades. The aim of this research was to solve the problem of designing a controller for systems where the characteristics are not completely known or vary. Different approaches were proposed and utilized. One successful approach is represented by self-tuning controllers (STC). The main idea of an STC is based on the combination of a recursive identification procedure and a selected controller synthesis. Different branches of the STC differ mainly in the method of the controller design while the identification recursive least squares method (RLSM) applied to a regression (ARX) model forms a standard.

Despite intensive research activity, there is a lack of controllers which are able to cope with practical requirements. Industrial applications on a large scale are still missing, however, many realized cases were successful. It is evident that until to now application has required a control engineer skilled in specific technology as well as versed in modern control methods.

Also the aim of this contribution is to inform of potential users about Self-Tuning Controllers Simulink Library - STCSL (see Bobál and Chalupa, 2002) and to help for practical usage in the simulation and real time conditions. The individual self-tuning algorithms are introduced in the brief form in User's Guide that is attached into the STCSL. This library can be also the suitable bridge

between theory and practice by presenting some simple controller algorithms in a form acceptable for industrial users. The theoretical background describes not only these simple algorithms, but also algorithms based on polynomial solutions of controller synthesis and controllers that are derived from the use of minimization of linear quadratic criterion, are given in Bobál, *et al.* (1999a, 1999b).

All the explicit self-tuning controllers that are included into STCSL have been algorithmically modified in the form of mathematical relations or as flow diagrams so as to make them easy to program and apply. Some are original algorithms based on a modified Ziegler-Nichols criterion, others have been culled from publications and adapted to make them more accessible to the user.

2. THEORETICAL BACKGROUND

2.1 Recursive identification

The regression (ARX) model of the following form

$$y(k) = \Theta^T(k)\phi(k-1) + n(k) \qquad (1)$$

is used in the identification part of the designed controller algorithms, where

$$\Theta^T(k) = \left[a_1\ a_2\ \dots\ a_{na}\ b_1\ b_2\ \dots\ b_{nb}\right] \qquad (2)$$

is the vector of the parameter estimates and

$$\begin{aligned}\phi^T(k-1) = [&-y(k-1) - y(k-2) \dots - y(k-na)\\ &u(k-1)\,u(k-2) \dots u(k-nb)]\end{aligned} \qquad (3)$$

is the regression vector ($y(k)$ is the process output variable, $u(k)$ is the controller output variable). The non-measurable random component $n(k)$ is assumed to have zero mean value $E[n(k)] = 0$ and constant covariance (dispersion) $R = E[n^2(k)]$. The recursive least squares method for calculating of the parameter estimates (2) is utilized, where adaptation is supported by directional forgetting (Kulhavý, 1987).

2.2 Algorithms of digital PID Ziegler-Nichols controllers

The PID controllers are still widely used in industry. These types of controllers are more convenient for users owing to their simplicity of implementation, which is generally well known. Provided the controller parameters are well chosen they can control a considerable part of continuous technological processes. To get a digital version of the PID controller, it is necessary to discretize the integral and derivative component of the continuous-time controller. For discretizing the integral component we usually employ the forward rectangular method (FRM), backward rectangular method (BRM) or trapezoidal method (TRM). The derivative component is mostly replaced by the 1st order difference (two-point difference). The recurrent control algorithms which compute the actual value of the controller output $u(k)$ from the previous value $u(k\text{-}1)$ and from compensation increment seem to be suitable for practical use

$$u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) + u(k-1) \qquad (4)$$

where q_0, q_1, q_2 are the controller parameters. The advantage of algorithm (4) is matter of fact that it is not necessary to storage last input and output data in the computer storage.

It is subsequently possible to derive further variants of digital PID controllers. First group of proposed self-tuning PID controllers is based on the classical Ziegler and Nichols (1942) method. In this well-known approach the parameters of the controller are calculated from the ultimate (critical) gain K_{pu} and the ultimate period of oscillations T_u of the closed loop system. The analytical expressions for computing of these critical parameters are derived in Bobál, *et al.* (1999a, 1999b).

2.3 Algorithms of pole placement controllers

A controller based on the pole placement method in a closed feedback control loop is designed to stabilise the closed control loop whilst the characteristic polynomial should have a previously determined pole. Digital controllers is possible can be expressed in the form of a discrete transfer function

$$G_R(z) = \frac{U(z)}{E(z)} = \frac{Q(z^{-1})}{P(z^{-1})} \qquad (5)$$

Let the controlled process be given by the transfer function

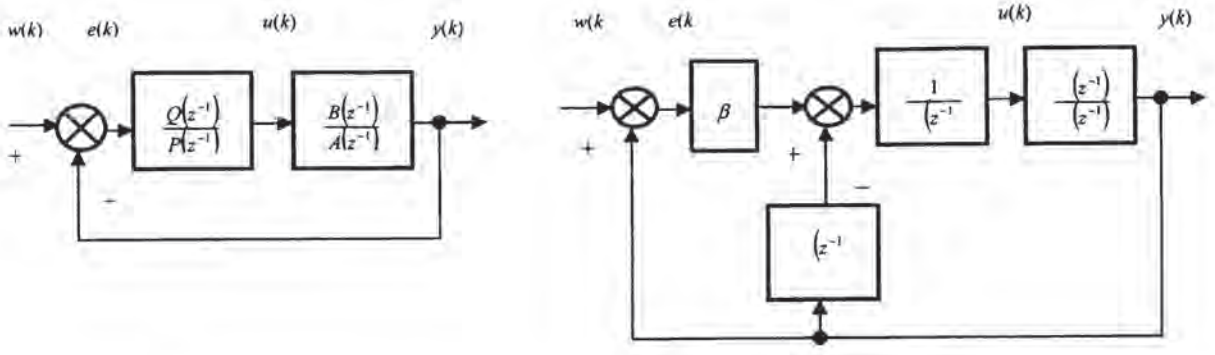
$$G_P(z) = \frac{Y(z)}{U(z)} = \frac{B(z^{-1})}{A(z^{-1})} \qquad (6)$$

with the polynomials $A(z^{-1})$, $B(z^{-1})$ for degree $n = 2$.

PID - A controller structure. This controller structure is shown in Fig. 2a). The controllers polynomial in equation have the form

$$\begin{aligned}Q(z^{-1}) &= q_0 + q_1 z^{-1} + q_2 z^{-2}\\ P(z^{-1}) &= (1 - z^{-1})(1 + \gamma z^{-1})\end{aligned} \qquad (7)$$

and then the characteristic polynomial of the closed-loop system with a PID-A controller (see Fig. 1a)) is in the form



a) PID - A controller

b) PID - B controller

Fig. 1 Block diagrams of control loops with pole placement PID controllers

$$A(z^{-1})P(z^{-1})+B(z^{-1})Q(z^{-1})=D(z^{-1}) \quad (8)$$

The pole placement of characteristic polynomial (8) determines the dynamic behaviour of the closed - loop system. The characteristic polynomial $D(z^{-1})$ can be specified by different methods. It is offen described by the dominant poles for a second order continuous model (controller PID A-1)

$$D(s)=s^2+2\xi\omega_n s+\omega_n^2 \quad (9)$$

The dominant poles are given by the desired damping factor ξ and the natural frequency ω_n of the closed-loop.

A PID-A controller with the characteristic polynomial (controller PID A-2)

$$D(z)=(z-\alpha)^2\left[z-(\alpha+j\omega)\right]\left[z-(\alpha-j\omega)\right] \quad (10)$$

is proposed in Bobál, *et al.* (1999a). Characteristic polynomial (10) has a double real pole $z_{1,2}=\alpha$ within interval $0 \le \alpha < 1$ and a pair of complex conjugated poles $z_{3,4}=\alpha \pm j\omega$, where $\alpha^2+\omega^2<1$. The parameter α influences the speed of the control-loop transient response and influences controller output changes, too. By changing parameter ω it is possible to influence the desired overshoot.

PID - B controller structure. The structure of the control loop with controller PID-B is shown in Fig. 1b). The characteristic polynomial has in this case form

$$A(z^{-1})P(z^{-1})+B(z^{-1})\left[Q(z^{-1})+\beta\right]=D(z^{-1}) \quad (11)$$

where polynomial $P(z^{-1})$ has the same form as in equation (7) and second polynomial in equation (11) is given by

$$Q(z^{-1})=(1-z^{-1})(q_0-q_2z^{-1}) \quad (12)$$

Using polznomial (9) it is possible to derived controller B-1, using equation (10) PID B-2.

3. TOOLBOX DESCRIPTION

The first modification of this toolbox used the Graphical User Interface - GUI (Bobál, *et al.*, 1999, Bobál and Böhm, 2000). The modification using GUI is depended on the individual MATLAB version. On that account the SIMULINK version of this toolbox (so called Self-Tuning Controllers Simulink Library - STCSL) has been designed.

3.1 Self-Tuning Controllers Simulink Library (STCSL)

The purpose of the STCSL was to create an environment suitable for creating and testing of self-tuning controllers. The library is available free of charge at Internet site (see Bobál and Chalupa, 2002). The library was created using MATLAB version 6.0 (Release 12), but it can be ported with some changes to lower MATLAB versions. Controllers are implemented in the library as standalone SIMULINK blocks, which allow an easy incorporation into existing simulation schemes and an easy creation of new simulation circuits. Only standard techniques of SIMULINK environment were used when creating the controller blocks and thus just basic knowledge of this environment is required for the start of a work with the library. Controllers can be implemented into simulation schemes just by the copy or drag & drop operation and their parameters are set using dialog windows. Another advantage of the used approach is a relatively easy implementation of user-defined controllers by modifying some suitable controller in the library. Nowadays the library contains 29 simple single input single output discrete self-tuning controllers, which use discrete models of second and third order for the on-line process identification. The

library package contains not only the controllers but also reference manual with simple description of the algorithm and the internal structure of each controller.

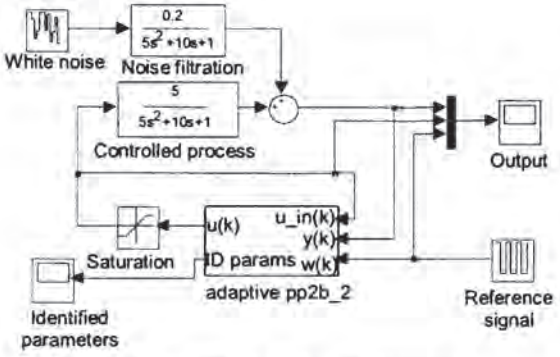


Fig. 2. Control circuit in SIMULINK environment

The typical wiring of any library controller is shown in Fig. 2. Each self-tuning controller from the library uses 3 input signals and provides 2 outputs. The inputs are the reference signal (w) and the actual output of controlled process (y). The last controller input is the current input of controlled process – the control signal (u_in). The value of this signal does not have to be the same as controller output e.g. due to the saturation of controller output. The main controller output is, of course, control signal – the input signal of the controlled process. The second controller output consists of the current parameter estimates of the controlled process model. The number of parameters this output consists of depends on the model used by on-line identification.

A scheme analogous to the scheme in Fig. 2 can be used to simulate the control process of both a discrete and a continuous controlled process with much more complicated structure. It is possible to implement processes with time variable parameters, processes described by non-linear differential equations etc.

3.2 Overview of library controllers

The Self-Tuning Controllers Library is started by opening the SIMULINK file *cntrl_lib.mdl*, which contains block schemes of all controllers. The name of the controller always corresponds with the name of the file, which provides the calculation of controller parameters and has the following structure: **xx(n)yyyy**. First two characters (**xx**) indicate controller type – **zn** stand for the controller synthesis based on Ziegler-Nichols method, **pp** represent controller with a pole placement synthesis, **mv** denote minimum variance controller, etc. The third character (**n**) in a controller name is a digit 2 or 3 corresponding to the order of the model used by on-line identification part of the controller. Following characters (**yyyy**) serve to cover other controller details. The survey of the individual controllers with the name of the m-files and requisite input parameters is introduced in Bobál *et al.* (2001). The detailed description of most of the controller algorithms is introduced in Bobál, *et al.* (1999a).

3.3 Internal controller structure

Each library controller is constructed as a mask of a subsystem, which consists of SIMULINK blocks and has inputs, outputs and parameters. Internal controller structure consists of SIMULINK blocks which provide, among others, the possibility of easy creation of a new controller by a modification of some suitable library controller. The structure of controller *pp2a_1* is presented in Fig 3 as an example. Each library controller consists of three basic parts:

- on-line identification block,
- block computing controller parameters,
- block computing controller output.

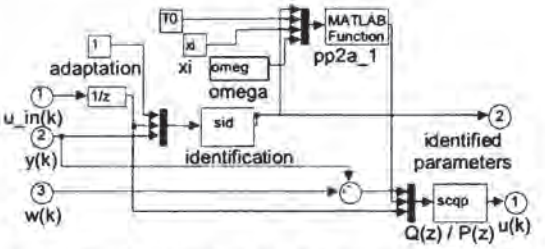


Fig. 3. Block scheme of PP2A_1 controller

3.4 Reference guide and help

Besides the files implementing the functionality of all library controllers the library package includes reference guide with a short description of all controller algorithms, of an operating with the controllers and of the internal structure of the controllers. The guide is provided in two versions:

- *pdf* format suitable for printing,
- *html* format used for the context help

Moreover the help for each function included in the library can be invoked by entering *help function_name* at MATLAB command line.

4. PRACTICAL USAGE OF STCL IN REAL - TIME APPLICATIONS

When using the library to control some real-time model or equipment the first phase consists of selecting appropriate controller and some simulations are performed to tune controller parameters. In the next phase the testing the controller using real model is performed. In the last phase the SIMULINK environment is used to generate source codes of program working outside MATLAB - SIMULINK environment. This code can be compiled, linked and loaded into industrial controllers.

Practical verification of library controllers has been performed using several laboratory models. One of them, the air heating system, is shown in Fig. 4. This system has two inputs (rotations of ventilator and a power of resistance heating) and two outputs (the flow of an air measured by the rotations of aircrew and a temperature inside the tunnel measured by resistance thermometer). The system was divided into two input-output pairs, where the first pair consists of the rotations of ventilator as an input and the flow of an air as an output. The second pair consists of the power of the resistance heating as an input and the temperature as an output. The aim is to control a two inputs two outputs system using two standalone single input single output controllers. This approach is known as the decentralized control.

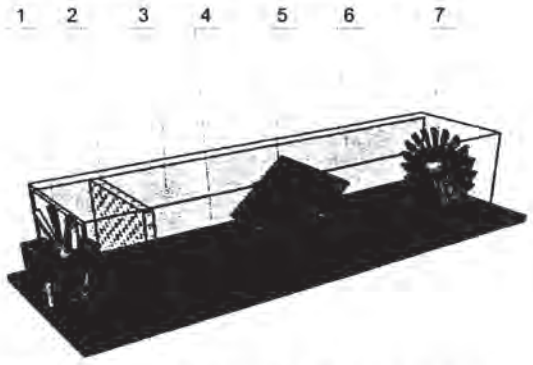


Fig. 4. Laboratory air-heating model (1- ventilator, 2-resistance heating, 3- pressure sensor, 4-resistance thermometer, 5-shutter, 6-shield, 7-air flow measurement)

The connection of the laboratory model and the SIMULINK environment has been realized through control and measurement PC card Advantech PCL-812 PG. Blocks for the reading analogue inputs and for writing to the analogue outputs on the PC card were used to communicate with the model. These blocks are implemented as s-functions written in C language, which allows low-level access to the ports of the PC computer. This mechanism allows the connection of SIMULINK and any PC compatible equipment designed to collect external data.

The sample of control course using a pole-placement controller is plot in Fig. 5, where reference signal is green, controlled signal is red and control signal is blue. The time constants of the pair ventilator-flow are relatively small when comparing with time constants of the heating-temperature pair and thus the output of the first pair becomes steady earlier. The courses demonstrate that the change of the resistance heating power does not affect the flow of an air, but the influence of ventilator rotation to the temperature is substantial. The parameters of the heating-temperature controlled system are thus changing in time and on-line identification method used should assign greater weight to newer data then to older data. Time courses of parameter estimates are presented in Fig 6. It is obvious from this picture

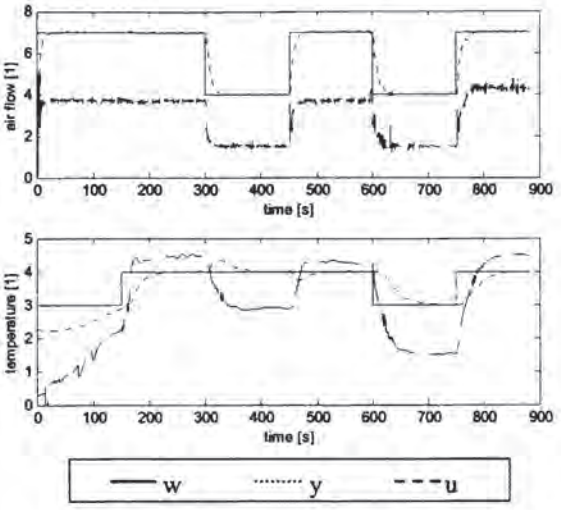


Fig. 5. Control courses using PP2B_1 controllers

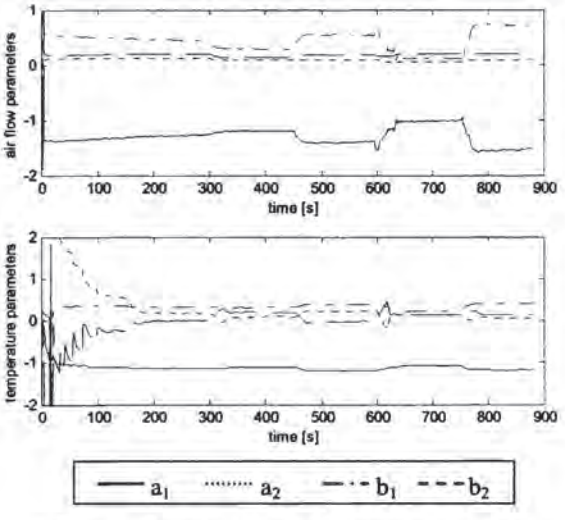


Fig. 6. Courses of model parameter estimates using PP2B_1 controllers

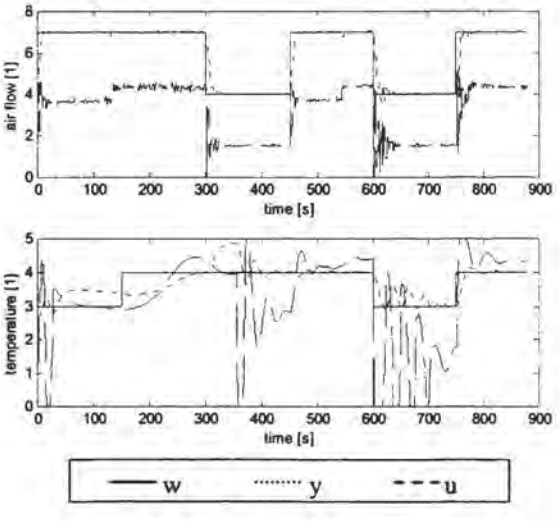


Fig. 7. Control courses using ZN2BR controllers

That the changes of parameters are greater in first phase – this is caused by inaccurate initial parameters estimates. Further changes of parameters correspond to changes of control signal what indicates the presence of a non-linearity in the system.

The control course is different when controller of another type was used as shown in Fig. 7. The course of reference signal is the same as in previous case, but the *zn2br* controllers were used. The *zn2br* controller uses Ziegler-Nichols algorithm of computation the value of control signal with the backward rectangular method of discretization integration component. In this case the oscillations of controlled signals are significantly greater and the quality of control process is lower when comparing with the pole placement controllers.

5. CREATING APPLICATIONS WITH REAL - TIME WORKSHOP

The MATLAB - SIMULINK environment can also be used to generate code to be used in controllers in industrial practice. *Real-Time Workshop*, one of the toolboxes shipped with MATLAB, allows generating of source code and programs to be used outside the MATLAB environment. The process of generating the source code is controlled by special compiler files that are interpreted by *Target Language Compiler*. These files are identified by the *.tlc* (target language compiler) extension and describe how to convert SIMULINK schemes to target language. Thereby source code is generated and after compiling and linking the resulting application is created. Applications for various microprocessors and operating systems can be created by selecting corresponding target language compiler files.

The target language compiler can create applications to be used under the Windows environment, which perform control algorithm and save results in a binary file with the structure acceptable by MATLAB. An analysis of the control process can then be performed using advantages of MATLAB functions and commands. Selecting another *.tlc* file leads to creation of a MS-DOS application or an application to be used on PC based industrial computers without a requirement of any operating system.

Many manufactures of industrial computers and controllers has created their own target language compiler files used to create applications for equipment they produce. *Real-Time Workshop* provides a relatively opened environment for the conversion of block schemes to various platforms where users can create their own target language compiler files for converting the block scheme to a source code and hence reach the compatibility with any hardware.

An application creation consists only of selecting appropriate target language compiler file, eventually setting compiler parameters and then start-up of compilation process.

6. CONCLUSIONS

The Self-Tuning Controllers Simulink Library is used in university course of adaptive control systems. Its architecture enables an easy user orientation in SIMULINK block schemes and source code of controllers' functions. The controllers provided are suitable for modification and thereby implementation of user-defined controllers. The compatibility with *Real-Time Workshop* ensures not only the possibility of laboratory testing using real time models but also the possibility of creating applications for industrial controllers.

ACKNOWLEDGEMENTS

This work was supported in part by the Grant Agency of the Czech Republic under grants No.102/02/0204, 102/00/0526 and in part by the Ministry of Education of the Czech Republic under grant MSM 281100001.

REFERENCES

Bobál, V., J. Böhm, J. Prokop and J. Fessl (1999a). *Practical Aspects of Self-Tuning Controllers: Algorithms and Implementation*. VUTIUM Press, Brno University of Technology, Brno (in Czech).

Bobál, V., J. Böhm, and R. Prokop (1999b). Practical aspects of self-tuning controllers. *International Journal of Adaptive Control and Signal Processing*, **13**, 1999, 671-690.

Bobál, V. and J. Böhm (2000). MATLAB-Toolbox for design and verification of self-tuning controllers. In: *Prep. of IFAC Symposium on Advances in Control Education*, Gold Coast, Australia, Paper No. 10S-1.

Bobál, V., J. Böhm, and P. Chalupa (2001). MATLAB-toolbox for CAD of simple self-tuning controllers. In: *Proc. 7th IFAC Workshop on Adaptation and Learning in Control and Signal Processing ALCOSP 2001*, Cernobbio-Como, Italy, 2001, 273-278.

Bobál, V. and P. Chalupa (2002). Self-Tuning Controllers Simulink Library. http: //www.utb.cz/stctool/.

Kulhavý, R. (1987). Restricted exponential forgetting in real time identification. *Automatica*, **23**, 586-600.

Ziegler, J. G. and N. B. Nichols (1942). Optimum settings for automatic controllers, *Trans. ASME* **64**, 1942, 759 – 768.