USAGE OF SELF-TUNING CONTROLLERS SIMULINK LIBRARY IN REAL TIME APPLICATIONS

Petr Chalupa, Vladimír Bobál

Department of Control Theory, Institute of Information Technologies,

Tomas Bata University in Zlin,

Abstrakt: Příspěvek představuje knihovnu samočinně se nastavujících regulátoru vytvořenou v prostředí MATLAB Simulink, jejichž návrh je založen především na metodách Ziegler-Nicholse a přirazení pólu v kombinaci s průběžnou identifikací pomocí ARX modelu. Základem příspěvku je požití regulátoru obsažených v knihovně pro řízení aplikací v reálném čase s využitím Real-Time Toolboxu. Z implementačního pohledu je stručně probrána jak možnost řízení přímo z prostředí Simulink, tak tvorba aplikací, které mohou běžet nezávisle na prostředí MATLAB Simulink.

Abstract: The contribution deals with the self-tuning controllers (STC) library designed under MATLAB Simulink environment. The design of library controllers is mainly based on the Ziegler-Nichols and pole-placement methods in combination with the on-line ARX model identification. The process of developing real-time applications using MATLAB Real-Time Workshop and several control courses of real laboratory models is discussed and presented.

Keywords: STC toolbox, MATLAB, Simulink, ARX model, on-line identification, Ziegler-Nichols method, pole-placement method.

1 SELF-TUNING CONTROLLERS

Self-tuning controllers belong by their character to the class of adaptive control systems. The main aim of the adaptive control approach is to solve the control problem in cases where the characteristics of a controlled system are unknown or time variable. The basic principle of the adaptive control system is to change the controller characteristics on base of the characteristics of control process. Self-tuning controllers use the combination of the on-line process identification on base of a selected model of the process and the controller synthesis based on knowledge of parameters of controlled process.

The goal of self-tuning controllers is to fulfil the following tasks:

- automatic tuning of digital or analogue controller,
- increase the quality of control process in cases of presence of unsteady disturbances,
- capture changes of controlled process parameters, which can be caused by various technological sources, e.g. the operating of the controlled equipment in the different modes,
- subsequent increase of the quality of the control process by adjustment of digital controller parameters.

The general task of optimal adaptive control with on-line process identification is very complicated and thus the method of **the forced separation of the identification and the control** is often used for the design of self-tuning controllers. The principle of this simplification is in the cyclic repeating of the following steps:

1. The process parameters are assumed to be known for the current control step and equal to their current estimations.
2. The control strategy for selected criterion of control quality is designed on base of previous assumption and controller output is calculated.
3. The next identification step is performed after the obtaining the new sample of the controlled variable (eventually the external measured disturbance) – the parameters of controlled process model are recomputed using a recursive identification algorithm.

2 SELF-TUNING CONTROLLERS LIBRARY

On base on monograph [1] was created a library of self-tuning controllers in Matlab Simulink environment [2]. The purpose was to create an environment suitable for creating and testing of self-tuning controllers. The library is available free of charge at internet site of Tomas Bata University in Zlin – www.utb.cz/stctool [3]. 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 work with the library. Controllers can be implemented to simulation schemes just by the copy or drag & drop operation and their parameters are set using dialog windows. Another advantage of 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. All of these controllers use discrete control laws, where controller parameters are computed by various methods. Most methods calculate controller parameters on base of the ultimate parameters of controlled process (Ziegler-Nichols approach [4]) or on base of pole placement approach. 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. A typical wiring of a library controller is shown in figure 1.

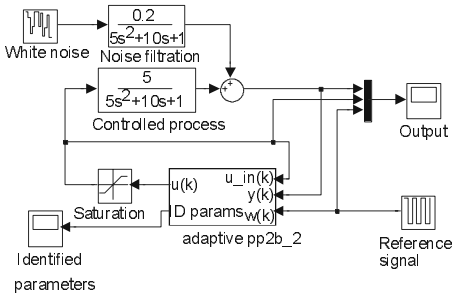

Fig. 1. Control circuit in Simulink environment

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 parameters estimations of the controlled process model. The number of parameters this output consists of depends on the model used by on-line identification: the controllers using second order model provide 4 values (a_1, a_2, b_1, b_2) and the controllers using third order model provide 6 values (a_1, a_2, a_3, b_1, b_2, b_3).

2.1 On-line process identification

The Controllers included in the library use discrete ARX model for the on-line identification on controlled process. The compact form of ARX model is described by the following equation:

$$y_k = \boldsymbol{\Theta}_k^T \cdot \boldsymbol{\Phi}_{k-1} + e_k$$
$$\boldsymbol{\Theta}_k = \left[a_1, \ldots, a_n, b_1, \ldots, b_m\right]^T$$
$$\boldsymbol{\Phi}_{k-1} = \left[-y_{k-1}, \ldots, -y_{k-n}, u_{k-1}, \ldots, u_{k-m}\right]^T$$

where $\boldsymbol{\Theta}_k$ is the vector of the parameters of the controlled process model, $\boldsymbol{\Phi}_{k-1}$ is the data vector and e_k represents the influence of an immeasurable disturbances.

The user of the library can select one of three recursive on-line identification methods offered for computation of the parameters estimation vector $\hat{\boldsymbol{\Theta}}_k$: the least squares method or one of its modifications – the least squares method with exponential forgetting and the least squares method with adaptive directional forgetting.

When using the basic least squares method, the influence of all pairs of controlled process inputs and outputs to the parameters estimations is the same. This property can be inconvenient for example when identifying the process with time-variable parameters. In this case it is better to use least square method with exponential forgetting where the influence of newer data to the parameters estimations is greater then the influence of older data. Another modification of lest squares method is the adaptive directional forgetting [5] where weight assigned to the data is changing on base of the course of input and output signals of identified process

2.2 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 29 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.

2.3 Controllers parameters

The behaviour of a controller can be influenced by changing parameters that are available for a given controller. Controller parameters can be divided in to two groups: parameters common to all controllers and controller specific parameters.

The group of common controller parameters consists of the sample time and parameters affecting on-line identification process. In addition some controllers allow to be adjusted by parameters specific just to this controller – controller specific parameters.

All controller parameters are set or changed by the standard approach of Simulink environment, that is by changing items in dialog window invoked for example by mouse double-click on the controller object. This dialog window also contains very short description of the corresponding controller and the "Help" button used to invoke the corresponding part of the hypertext reference guide.

2.4 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. As stated in the previous chapters, each controller uses 3 input signals (reference signal *w*, controlled signal *y* and control signal *u_in*) and provides 2 outputs (control signal *u* and current estimations of model 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. Each library controller consists of three basic parts:

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

From the programmer point of view each block corresponds to the stand-alone program file, which is, after the simulation start, interpreted by the Matlab environment. The blocks are implemented as a *Matlab Fcn* and an *S-Function*. The *Matlab Fcn* block performs the call of a Matlab function, which converts input data vector to the output data vector. The *S-Function* block is more universal and corresponding function can archive and use also discrete and continuous states and thus work with some sample time.

The basic version of the library uses the Matlab programming language to implement all mentioned functions. The advantage of this approach is an easy implementation of matrix and vector operations.

2.5 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, an operating with the controllers and the description of internal structure of controllers. The guide is provided in two versions:

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

Moreover the help for each function included in the library can be invoked by entering *help function_name* at Matlab command line. Then the short description of the function, its inputs and outputs is displayed into Matlab command window.

3 USAGE OF SELF-TUNING CONTROLLERS LIBRARY

When using the library to control some real-time model or equipment the 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. 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 figure 2. 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 airscrew 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 decentralised control.

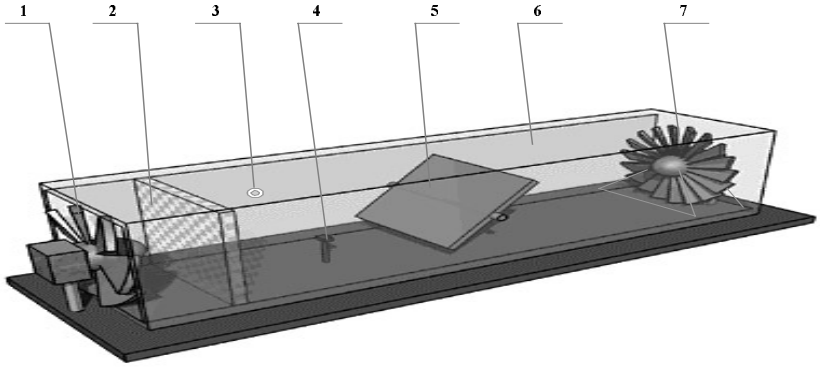

Fig. 2: Laboratory through - flow heater (1 – ventilator, 2 – electric resistance heating, 3 – pressure sensor, 4 –resistance thermometer, 5 – throttle valve, 6 – cover of tunnel. 7 – flow indicator).

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 reading analog 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.

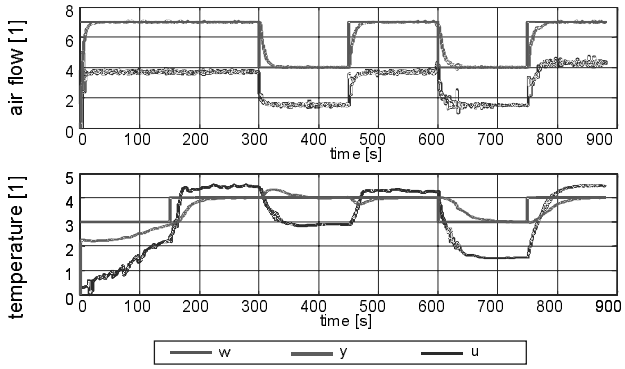

Fig. 3. Control courses using PPB_1 controller

The sample of control course using a pole-placement controllers is plot in figure 3, 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 demonstrates that the change of the resistance heating power does not affect the flow oaf 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.

4 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 any 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.

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

4.1 Compatibility of Simulink schemes

Target language files provided in standard installation of Real-Time Workshop unfortunately does not support all Simulink blocks. The set of unsupported blocks depends on a target language compiler file used, but when working with the Self-tuning Controllers Library the user encounters the problems especially with two blocks: *MATLAB Fcn* and *S-Function*. Target language compiler files does not support the *MATLAB Fcn* block and the *S-Function* block is supported only in cases where the function is written in the C language without calls into Matlab.

The standard library version *cntr_lib* was revised and the *cntrl_lib_rtw* was created to achieve full compatibility of the library and *Real-Time Workshop*. The functionality and internal structure of all controllers is the same in both versions, the difference is only in resources used to program individual parts of the controllers. The *MATLAB Fcn* blocks are used in standard version to compute controllers' parameters. These blocks were superseded by C language s-functions, which compute their outputs (controllers' parameters) only on base of their inputs (parameters estimations of controlled process model and eventually other controller-specific parameters) – these functions do not use any states. The on-line identification block and the blocks computing controller output are implemented as an s-functions that had to be rewritten to C language. The file names are in both library versions the same – the difference is in the extension: functions written in Matlab language use *.m* extension and functions written in C language use *.c* extension.

4.2 Library versions

As mentioned in the previous chapter, the Self-tuning Controllers Library is available in two versions: the standard version and the version for *Real-time Workshop*. Functionality, the number of controllers and the internal controller structure is the same in both versions – the difference is in internal program implementation of program code

The standard version uses Matlab programming language, with simple code structure, where no variable declarations are required – program interpreter assigns the type to the variable according to its usage. Another advantage of this language is very simple implementation of a matrix and vector operations. Especially on-line identification block uses matrix multiplications and transpositions and the computation of dead-beat controllers' parameters requires matrix inversion. The code written in this language is easy to read, to study and to understand and thus suitable when studying the details of controllers' implementations. This code is also suitable for writing user-defined controllers.

The *Real-time Workshop* version of the library was constructed using the C programming language. The main advantage of this version is its portability, because only functions written in C language are supported when converting block schemes to applications through the Target Language Compiler. The disadvantage resides in larger source files and thus worse readability of the code.

5 CONCLUSION

The Self-tuning controllers 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 controller 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.

6 ACKNOWLEDGMENT

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.

7 LITERATURE

[1] BOBÁL, V., BÖHM, J., PROKOP, R., FESSL, J.: *Praktické aspekty samočinně se nastavujících regulátorů: algoritmy a implementace.* Nakladatelství VUTIUM, VUT Brno, 1999, ISBN 80-214-1299-2. (in Czech)

[2] BOBÁL, V., BOHM, J., CHALUPA, P.: 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.

[3] BOBÁL, V., CHALUPA, P.: Self-Tuning Controllers Simulink Library. http://www.utb.cz/stctool/, 2002

[4] ZIEGLER, J. G., NICHOLS, N. B.: Optimum settings for automatic controllers, *Trans. ASME* 64, 1942, 759 – 768

[5] KULHAVÝ, R.: Restricted exponential forgetting in real time identification, *Automatica 23,* 1987, 586 – 600.

8 CONTACT

Vladimír Bobál, Petr Chalupa

Department of Control Theory

Institute of Information Technologies

Tomas Bata University in Zlin,

Nám. TGM 275, 762 72 Zlín, Czech Republic,

Tel.:+420 57 603 3217,

E-mail: bobal@ft.utb.cz, chalupa@ft.utb.cz