Tuning and Implementation of PID Controllers using Rapid Control Prototyping

Radu Duma, Mirela Trusca, Petru Dobra

Automatic Control Department, Technical University of Cluj–Napoca
(e-mail: radu.duma@aut.utcluj.ro, mirela.trusca@aut.utcluj.ro, dobra@aut.utcluj.ro)

Abstract: The paper presents a Rapid Control Prototyping (RCP) toolbox for Scilab/Scicos, which generates real-time C code using RTAI-Lab. The toolbox contains three main blocks: PID controller, Fractional Order PID (FOPID) controller and Least Mean Square (LMS) adaptive filter. Using the implemented toolbox two closed loop control systems using a PID and a FOPID controller are implemented. For tuning the PID controller the relay feedback tuning method and a robust experimental tuning method are used. For the FOPID controller tuning an experimental tuning method is implemented.

Keywords: rapid control prototyping, PID controller, fractional order PID controller, controller tuning, closed loop, relay feedback.

1. INTRODUCTION

Rapid Control Prototyping (RCP) refers to the use of rapid prototyping for the implementation and testing of control algorithms in a real-time environment. It is a usual application for engineers which model and simulate control systems. RCP requires two components: a Computer Aided Control System Design (CACSD) software and a dedicated hardware, capable of running tasks in real-time. Using Linux-RTAI (Bucher et al., 2004) the CPU of the PC becomes the RCP destination. An acquisition card can be used as an input/output interface with the process to be controlled

CACSD tools are extensively used to generate real-time code automatically. The graphical programming approach removes the need to write software by hand and allows the engineer to focus instead on improving functionality and performance. Complete system design is carried out within the simulation environment. An RCP toolbox has as its main feature the concept of automated code generation, which refers to the translation of the model of a control algorithm into source code. Automated code generation is considered to be the fifth generation in software evolution (Erkkinen, 2003).

We have implemented an RCP toolbox for Scilab/Scicos, which generates real-time C code for Linux-RTAI, using RTAI-Lab. The toolbox can be used for system identification and control and has the abbreviation SIC. The code generated from the blocks of the toolbox is not dependent on the acquisition card used for the input/output related functionality. In order to test the implemented toolbox two experimental PID controller tuning methods and a Fractional Order PID (FOPID) experimental tuning method are implemented. The closed loop performances obtained with the three controllers are compared. Using RCP the engineer does not have to write code by hand, but can concentrate on improving the performance and the efficiency of the control algorithm.

In the industry control loops of PI and PID type are extensively used. The reason for which the PID controller is so widely used is its simple structure which has proved to be very robust with regard to many commonly control problems. The experimental tuning methods used for the PID controller are: the relay feedback method (Astrom and Hagglund, 1984) and a robust experimental tuning method (Chen and More, 2005).

Fractional order calculus is used in a wide range of science and engineering fields, especially for fractional-order systems. Fractional controllers are a research topic also in the field of control systems (Machado, 1997; Petras et al., 1998; Podlubny, 1999). Xue et al., (2006) presented a case study for FOPID control of a DC motor with elastic shaft. Barbosa et al. (2008) investigated several types of FOPID controllers for a laboratory servo system. Feliu et al. (2008) used a robust FOPID controller for a main irrigation canal pool. All the above mentioned papers do not offer a practical implementation of the fractional order PID controller and of the closed loop control algorithm, presenting only simulation results. This paper presents a practical implementation of a FOPID controller and the control of a process implemented on an analog computer.

The structure of the paper is presented below. In section 2 some of the relevant work related to RCP is presented. Section 3 describes the implemented RCP toolbox, presents the hardware test setup used for testing the toolbox and describes the translation process from block diagram to source code. In this section the transfer function of a DC motor is defined. This transfer function is implemented on an analog computer and is the process for which the PID and FOPID controllers are tuned. In section 4 the *PID* block of the implemented toolbox is described. The equations used for the implementation of the computation function of the block are presented. In section 5 the relay feedback method is used for tuning a PID controller for the process defined in section 3, while in section 6 a robust experimental tuning method for PID controllers is presented and used for the same process. In section 7 the *FOPID* block implemented in the RCP toolbox is presented and the discrete equations used for the implementation of the computation equation of the block are described. In this section also a FOPID controller is tuned for

the process defined in section 3. Section 8 presents conclusions and final remarks.

2. RELATED WORK

In this section the most significant work related to RCP is presented.

Matlab/Simulink/Realtime-Workshop is a powerful but expensive CACSD tool. MathWork's developed toolboxes for some wildly used targets: Motorola MPC555, Infinion C166, C2000 and C6000 DSP families from Texas Instruments. Rebeschiess (1999) developed the MICROS toolbox for standard 80C166 micrcontrollers. A low cost DSP based RCP system for engineering education and research using as CACSD tool Matlab/Simulink is presented in (Hercog et al., 2006). Microchip (2009) developed a Matlab RCP toolbox for the dsPIC33 Digital Signal Controllers (DSC), while Kerhuel (2009) developed a toolbox which offers support for several Microchip microcontrollers and DSCs. Duma and Dobra implemented a RCP toolbox for the Renesas M32C87 microcontroller (Duma et al., 2010)

A free, open source, solution is based on Scilab/Scicos (Campbell et al., 2009) and Linux-RTAI, which uses the processor of a general purpose computer for executing real-time tasks. A real-time patch is applied to the standard Linux kernel. A modified version of the Scicos code generator, RTAI-Lab generates hard real-time code compatible with Linux-RTAI.

Ravn (2006) implemented an adaptive control toolbox, for Scilab/Scicos, which can be used for RCP. A target supported in Scilab/Scicos is the Microchip dsPIC DSC microcontroller ([Evidence]).

National Instruments LabVIEW graphical code can be used to develop ARM microcontroller embedded applications for the Stellaris LM3S88962 using the LabVIEW Embedded Module for ARM Microcontrollers (National Instruments, 2010).

3. RCP TOOLBOX FOR SCILAB/SCICOS AND LINUX-RTAI

The software tool chain used for the implemented RCP toolbox presented in this paper is an open-source community work, so no license is required. Scilab/Scicos is a free CACASD software developed by INRIA (Institut National de Recherche on Informatique et on Automatique) and ENPEC (Ecole Nationale des Ponts et Chaussee). RTAI and the Linux distribution are also „open-source" softwares.

An RCP system must have the capability of running tasks in real-time. The standard Linux distributions like: Fedora, Ubuntu, Debian are not real-time operating systems (OS), mainly due to: time-sharing, the lack of preemption of the kernel, virtual memory and cache problems. In order to make Linux a real-time OS a hard real-time patch, Linux-RTAI, must be applied to the standard Linux kernel. Linux-RTAI was developed at the Dipartimento di Ingeneria Aerospaziale del Politecnico di Milano. The COMEDI project provides drivers, library functions and an application programming interface to interact with signal acquisition hardware. Hundreds of devices are supported.

The SIC toolbox (Fig. 1) can be used for system control and identification and consists of three main blocks: PID controller, Fractional Order PID controller and Least Mean Square adaptive filter. The toolbox also has blocks for the generation of Pseudo Random Binary Sequences (PRBS), for dead band generation and for the generation of a constant signal.

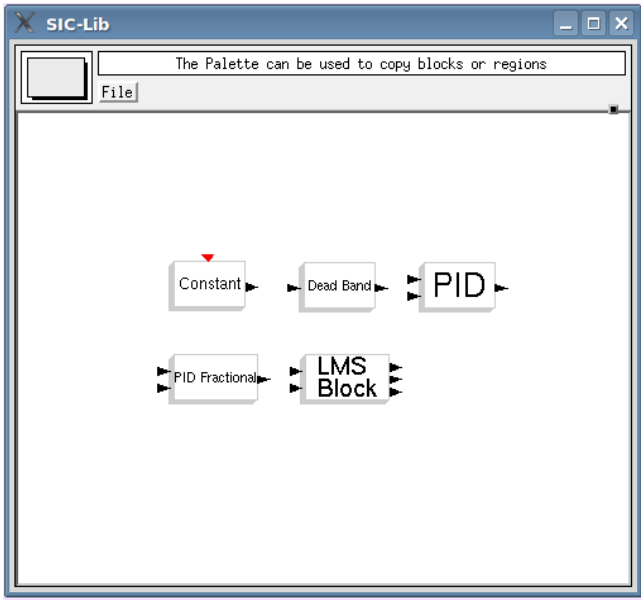

Fig. 1. The System Identification and Control RCP toolbox

A hardware test setup (Fig. 2) is implemented in order to test the RCP toolbox. The computer used is not a powerful one. It has a CPU speed of 700MHz, so hardly could it be used for other purposes. The real-time patch is applied to a Fedora 8 Linux distribution. An AT-MIO-16E-10 acquisition card from National Instruments is used. The board has to be connected on the ISA bus of the PC.

All the aspects presented above make of this RCP solution a low cost but flexible and powerful real-time environment for testing and implementing control algorithms.

Fig. 2. The hardware test setup

The control model is implemented in Scilab/Scicos. The automatic code generation is performed directly from the Scicos scheme through the *RTAI codegen* menu option. This option leads to a binary executable which is a real-time implementation of the control algorithm and which can be monitored using a virtual oscilloscope, Xrtailab. Xrtailab allows changing different parameters of the control scheme online, without a new compilation.

The bock diagram for a closed loop control algorithm is presented in Fig. 3.

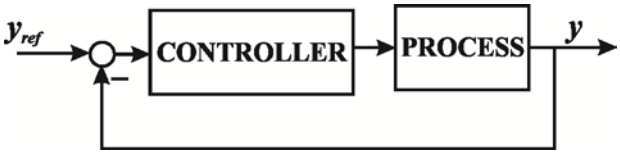

Fig. 3. Block diagram of a closed loop control system

Corresponding to the diagram presented in Fig. 3, a Scicos closed loop control model is presented in Fig. 4.

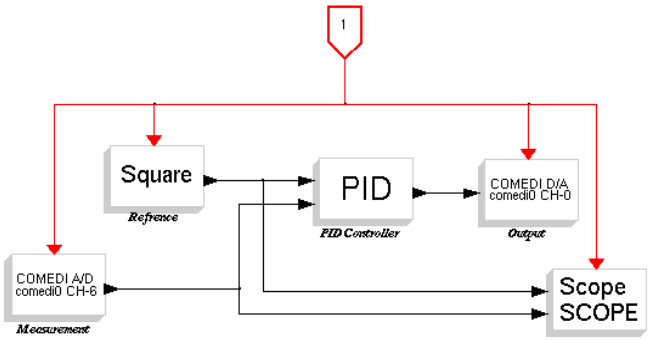

Fig. 4. Scicos model for closed loop control with PID controller

The reference rectangular signal is applied on the reference input port of the *PID* block, implemented in the toolbox presented in this paper. The *PID* block is described in section 4. The output of the process is acquired using the *Comedi A/D* Scicos block, for analog to digital conversion, and is applied on the feedback port of the *PID* block. The *PID* block computes a new value for the control signal, which is applied at the input of the controlled process using a *Comedi D/A* block. Using a *Scope* block the rectangular reference input signal and the response of the process can be visualized.

From the model implemented in Fig. 4, using the Scicos *RTAI Codegen* menu, real-time code is generated. The code runs on the processor of the PC and the acquisition card is used as an input/output interface with the process through the *Comedi A/D* and *Comedi D/A* Scicos blocks.

The generic transfer function of an electric motor, used in a positioning system, is implemented on the analog computer (Meda 43HA):

$$H(s) = \frac{1}{s(T_e s + 1)(T_m s + 1)} \tag{1}$$

where $T_e = 0.005$ s is the electrical time constant and $T_m = 0.5$ s is the mechanical time constant, which is the ration between the moment of inertia and the viscous friction coefficient J/B. The integral component of the transfer function is typical for a positioning system, being the relation from angular velocity to angular position of the rotor.

Corresponding to the block diagram presented in Fig. 3, the process used in this paper is the one defined in equation (1). For this process the two experimental PID tuning methods and the FOPID tuning method are implemented.

4. PID CONTROLLER

In this section is described the *PID* block implemented in the RCP toolbox presented in this paper. This block is used for the implementation of closed loop control Scicos model with PID algorithm (Fig. 4). The parameters of the PID controller are determined using the methods presented in sections 5 and 6.

The standard equation of a PID controller is presented below:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{2}$$

where $e(t) = y_{ref}(t) - y(t)$ is the error signal y_{ref} is the set point and y is the output of the process.

During years of research several modifications have been done to the standard PID controller structure. A pure derivative gives a very large amplification of measurement noise. The gain of the derivative must thus be limited, by low-pass filtering the derivative term, resulting in a limited gain *N* at high frequencies. *N* is typically in the range of 3-20. Only a fraction β of the reference signal acts on the proportional part, while the derivative component acts only on the output of the process, not on the error signal.

If the derivative operator $p = \frac{d}{dt}$ is introduced, equation (2) becomes:

$$u(t) = K_p\left((\beta y_{ref}(t) - y(t)) + \frac{1}{pT_i}e(t) - \frac{pT_d}{1 + pT_d/N}y(t)\right) \tag{3}$$

In order to obtain a discrete form of equation (3) the rules presented below are used.

The proportional part $P(t) = K_p(\beta y_{ref}(t) - y(t))$ requires no transformation since it is purely static.

In the case of the integral part $I(t) = K_p/T_i\int_0^t e(\tau)d\tau$ the integration can be approximated using:

$$\int_0^{t+T_s} e(\tau)d\tau = \int_0^t e(\tau)d\tau + \int_t^{t+T_s} e(\tau)d\tau \cong \int_0^t e(\tau)d\tau + T_s e(t)$$

which gives the forward time discretization rule:

$$I(t+T_s) = I(t) + K_p T_s/T_i\, e(t).$$

The derivative component:

$$\frac{T_d}{N}\frac{dD(t)}{dt} + D(t) = -K_p T_d \frac{dy(t)}{dt}$$

is approximated using backward differences:

$$\frac{T_d}{N}\frac{D(t) - D(t-T_s)}{T_s} + D(t) = -K_p T_d \frac{y(t) - y(t-T_s)}{T_s}$$

which gives the approximation:

$$D(t) = \frac{T_d}{T_d + NT_s} D(t-T_s) - \frac{K_p T_d N}{T_d + NT_s}(y(t) - y(t-T_s))$$

The structure of the PID controller with integrator anti-windup is shown in Fig. 5.

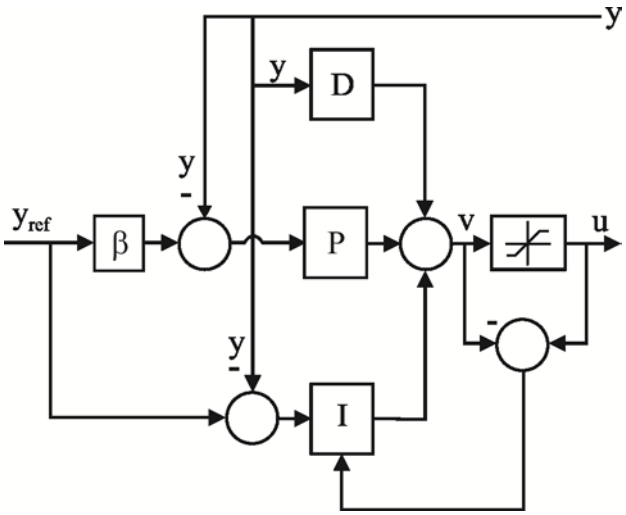

Fig. 5. The structure of the PID controller with anti-windup

Taking into consideration the above remarks, the PID algorithm can be described by the following equations:

$$\begin{aligned} P(n) &= K_p(\beta y_{ref}(n) - y(n)) \\ D(n) &= \alpha_d D(n-1) + \beta_d(y(n-1) - y(n)) \\ v(n) &= P(n) + I(n) + D(n) \\ I(n+1) &= I(n) + \beta_i(y_{ref}(n) - y(n)) + \beta_t(u(n) - v(n)) \end{aligned} \quad (4)$$

where $\alpha_d = T_d/(T_d + NT_s)$, $\beta_d = (K_p T_d N)/(T_d + NT_s)$, $\beta_i = K_p T_s / T_i$ and $\beta_t = T_s / T_t$.

In the above realations *Kp* is the proportional gain, *Ti* is the integration time, *Td* is the derivative time, *Tt* is the integral correction time and *Ts* is the sampling time.

Each Scicos basic block is defined by two functions: a computational function, normally written in C, and a function which handles the interactions with the editor. Equations (4) were used to implement the discrete PID controller computational function in C.

For the PID controller the following parameters have to be chosen: *Kp*, proportional gain; *Ti* integration time; *Td* derivative time; β fraction of control signal; *N* high frequency limiter of derivative action; u_{min} minimum saturation value; u_{max} maximum saturation value; and T_s sampling time. In Fig. 6 is presented the *Set Block properties* window for setting the parameters of the PID controller.

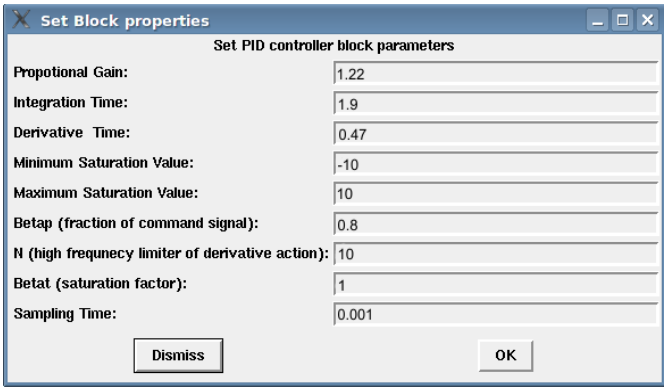

Fig. 6. Window for setting the parameters of the PID controller

5. RELAY FEEDBACK METHOD

Astrom and Hagglund (1984) proposed an extension of the Ziegler-Nichols method. In order to obtain the limit cycle the process must not be brought at the stability limit, but a nonlinear feedback of relay type is introduced in the control loop (Fig. 7). Results using the relay feedback method are presented in (Besancon-Voda, 2002).

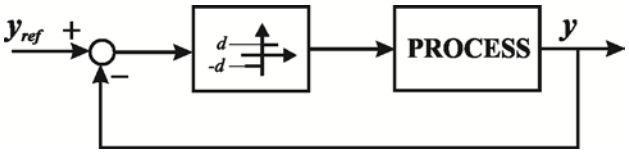

Fig. 7. Block diagram of the relay feedback method

The relay feedback causes the process to oscillate with controlled amplitude. From the limit cycle the period of oscillation, also known as ultimate period T_u is read. Based on the describing function of the relay, the relation for the ultimate gain K_u is determined:

$$K_u = \frac{4d}{\pi a} \quad (5)$$

where *a* is the amplitude of the limit cycle, and *d* is the amplitude of the output signal of the relay. In order to obtain the limit cycle the amplitude of the output signal of the relay, *d*, has to be changed. After determining the ultimate period T_u and the ultimate gain K_u based on the Ziegler-Nichols table for the ultimate period method the parmeters of the PID controller are computed. The relations for determinng the paramenters of the controller are: $K_p = 0.6K_u$, $T_i = T_u/2$ and $T_d = T_u/8$. The computed paramters are introduced in the graphical user interface presented in Fig. 6.

The Scicos model implemented for the relay feedback method is presented in Fig. 8. From this model using the Scicos *RTAI Codegen* menu, real-time code is generated. Using the *Scope* Scicos block, the reference rectangular

signal and the response of the system can be logged and visualized with the virtual Xrtailab oscilloscope (Fig. 9).

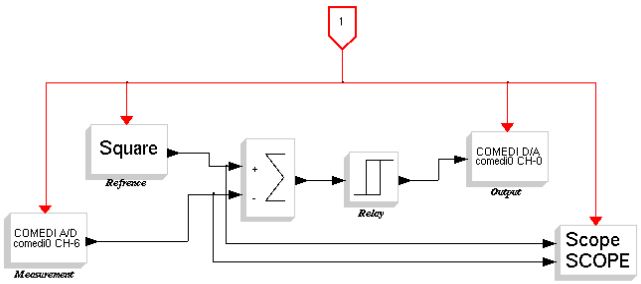

Fig. 8. Scicos diagram for relay feedback method

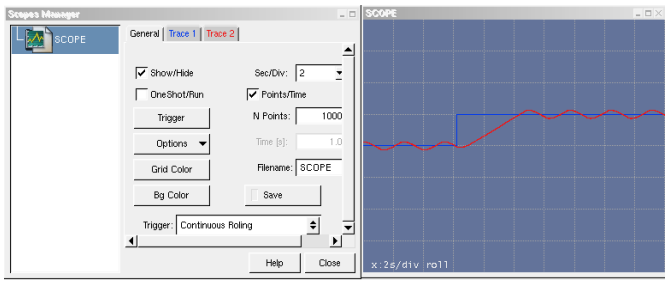

Fig. 9. The controlled oscillation of the process using the relay feedback method

From Fig. 9 it can be observed that the system oscillates both at the low and high levels of the reference rectangular input signal. The Xrtailab software has an option which allows the user to save the displayed data to a file. Using this feature the limit cycle which is the output of the process implemented on the analog computer and acquired using a *Comedi A/D* block, is stored to a file and processed in Scilab in order to compute the parameters of the PID controller. The limit cycle obtained when the reference signal is at the low level is presented in Fig. 10.

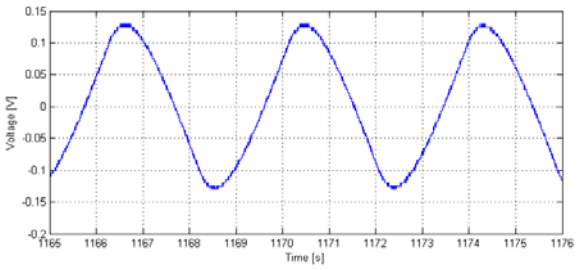

Fig. 10. Limit cycle obtained using the relay feedback method

From Fig. 10 the critical period T_u is determined and the ultimate gain K_u is computed using (5). The same procedure is applied for the limit cycle obtained when the reference signal is high. An average between the ultimate gain and the ultimate period obtained for the two limit cycles is computed.

In order to prove the correctness of the implemented algorithms for controller tuning and for closed loop control, snapshots are taken with an oscilloscope. Using the probes of the oscilloscope, signals are measured on the expansion pad of the acquisition card. For the two algorithms, the signal generated from the PC using the digital to analog converter of the acquisition card and the output of the process implemented on the analog computer are measured. In Fig. 11, corresponding to the Scicos model presented in Fig. 8, are shown in the upper part of the snapshot the output of the relay generated with the *Comedi D/A* block and in the lower part of the snapshot the obtained limit cycle, which represents the output of the process implemented on the analog computer.

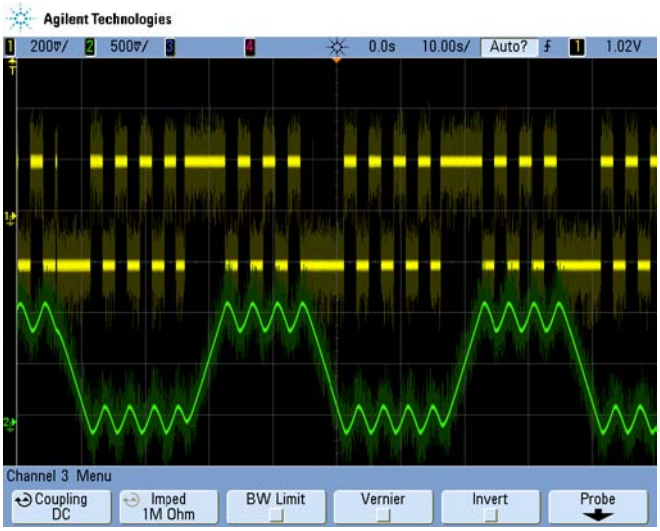

Fig. 11. The control signal (the output of the relay) and the output of the process when using the relay feedback method

Using the Ziegler Nichols relations the parameters of the controller are determined and the step response is shown in Fig. 12. The data is logged using the Xrtailab virtual oscilloscope.

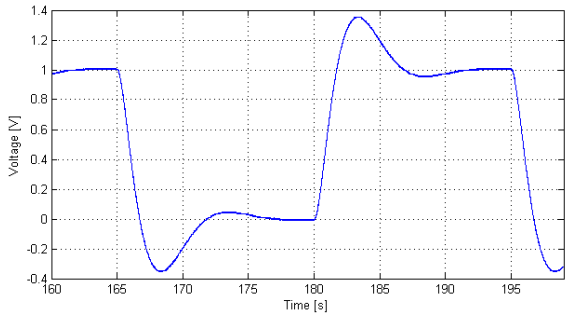

Fig. 12. The response of the closed loop system using a PID controller tuned using the relay feedback method

In Fig. 13, corresponding to the Scicos model presented in Fig. 4, are shown in the upper part of the snapshot the output of the PID controller which represents the control signal and in the lower part of the snapshot the closed loop response of the process to a rectangular reference signal.

The response time of the closed loop system is 12 seconds and the overshoot is 35%. The performances obtained with the extended Ziegler-Nichols method are acceptable for the presented example.

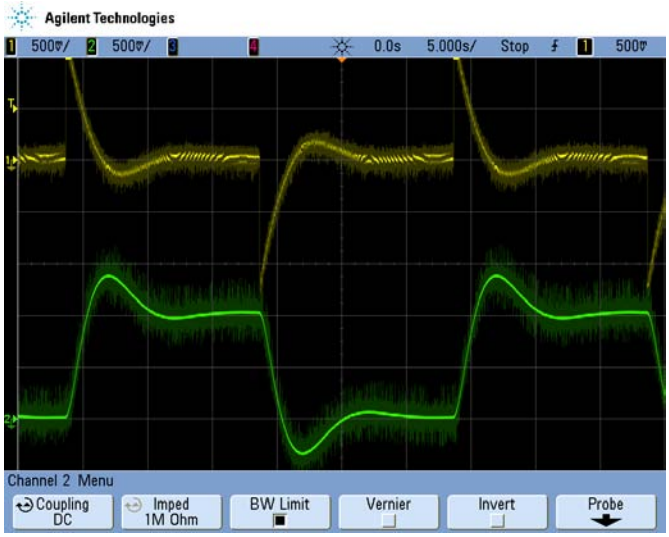

Fig. 13. The control signal and the output of the process for a closed loop system with a PID controller tuned with the relay feedback method

6. ROBUST EXPERIMENTAL TUNING METHOD

Chen and More (2005) proposed for controller tuning a robust tuning method. The method is based on the condition that the phase Bode plot at a specified frequency ω_c (referred to as "tangent frequency") at the point where sensitivity circle touches Nyquist curve is locally flat which implies that the system will be more robust to gain variations. Their paper presents a new tuning rule which gives a new relationship between T_i and T_d instead of the equation $T_i = 4T_d$, proposed in the modified Ziegler-Nichols methods ((Hagglund and Astrom, 1995), (Hang et al., 1991)).

The PID controller determined using this tuning method yields a faster closed loop system only if a higher frequency of oscillation (that in the case of the previous tuning method) can be determined. Also this frequency has to be invariant with respect to the introduced dead time.

After the initial selection of the tangent frequency ω_c and of the tangent phase γ_m using an iterative algorithm the phase $\angle P(j\omega_c)$ and the module $|P(j\omega_c)|$ of the process can be estimated. The derivative of the phase of the open loop system $s_p(\omega_c)$ can be approximated by Bode's Integral (Karimi et al., 2002):

$$s_p(\omega_c) \approx \angle P(j\omega_c) + \frac{2}{\pi}[\ln|K_g| - \ln|P(j\omega_c)|] \tag{6}$$

where $|K_g| = P(0)$ is the static gain of the process.

The equations for determining the parameters of the PID controller, as described in Chen and More (2005), are presented below:

$$K_p = \frac{\cos(\gamma_m)}{\left|P(j\omega_c)\sqrt{1+\tan^2(\gamma_m - \angle P(j\omega_c))}\right|}$$

$$T_i = \frac{-2}{\omega_c\left[s_p(\omega_c) + \tan\left(\hat{\Phi}\right) + \tan^2\left(\hat{\Phi}\right)s_p(\omega_c)\right]} \tag{7}$$

$$Td = \frac{-T_i\omega_c + 2s_p(\omega_c) + \sqrt{\Delta}}{2s_p(\omega_c)\omega_c^2 T_i}$$

where: $\hat{\Phi} = \gamma_m - \angle P(j\omega_c)$ and

$$\Delta = T_i^2\omega_c^2 - 8s_p(\omega_c)T_i\omega_c - 4T_i^2\omega_c^2 s_p^2(\omega_c)$$

In order to estimate the phase and the module of the process at the desired frequency a closed loop system with relay and time delay has to be implemented (Fig. 14).

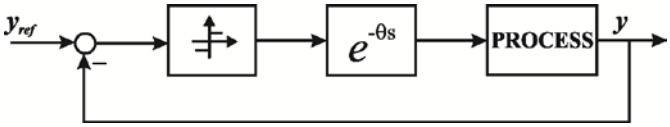

Fig. 14. Block diagram of a system with relay and dead time

The iterative algorithm described in detail in Chen and More (2005) is implemented in order to tune the parameters of the PID controller. The frequency of oscillation caused by the relay can be changed by introducing an artificial dead time on the direct path. Starting with the selected tangent frequency and using an iterative method, the dead time corresponding to the desired frequency is determined.

The selection of the value for ω_c depends on the dynamic of the process for which the controller must be tuned. For the majority of the processes there is an interval for the selection of ω_c in order to realize the flat phase condition. If this interval is not known the initial selection for ω_c can be the cuttoff frequency. In order to tune a controller for the process defined in (1) are considered $\omega_c = 2.42$ rad/s and $\gamma_m = 45^\circ$.

The desired frequency is computed using the relation: $\omega_n = 2\pi/T_u$, where T_u represent the ultimate period, which is determined from the limit cycle. The module of the process is estimated using the relation: $|P(j\omega_n)| = 1/K_u$, where K_u is the ultimate gain determined using (5). The phase is computed using the relation $\angle P(j\omega_n) = -\pi + \omega_n\theta$, where θ represents the dead time for which the desired frequency is obtained.

In order to determine the parameters of the controller, the Scicos model presented in Fig. 15, corresponding to the block diagram presented in Fig. 14 is implemented. The output of the process is acquired using a *Comedi A/D* Scicos block. The difference between the reference signal and the output of the process is applied at the input port of a relay with hysteresis

Scicos block. After the relay a *Variable delay* Scicos block is used for the implementation of the dead time.

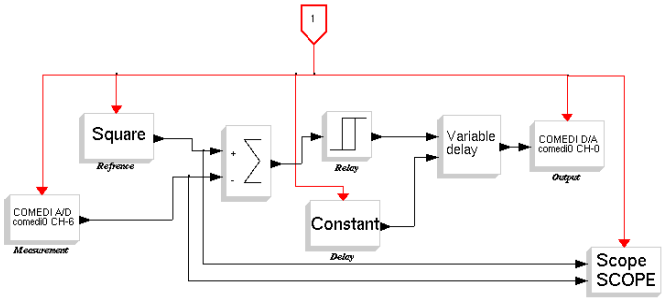

Fig. 15. Scicos model for the closed loop system with relay and dead time

The rectangular reference signal and the output of the process are logged and visualized with the virtual oscilloscope Xrtailab (Fig. 16).

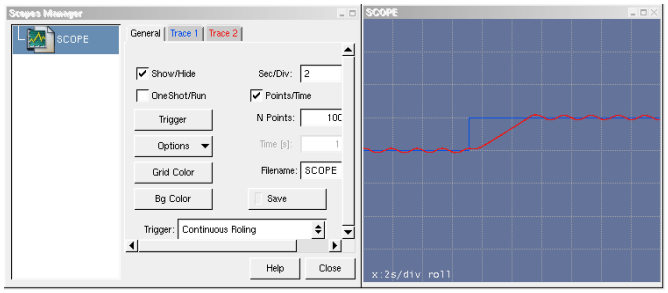

Fig. 16. The controlled oscillation of the closed loop system with relay and dead time

As in the case of the relay feedback method an average between the ultimate gain and the ultimate period obtained for the two limit cycles are computed and the average values are used for computing the parameters of the PID controller. The limit cycle corresponding to the desired frequency, obtained for a low reference signal, is presented in Fig. 17.

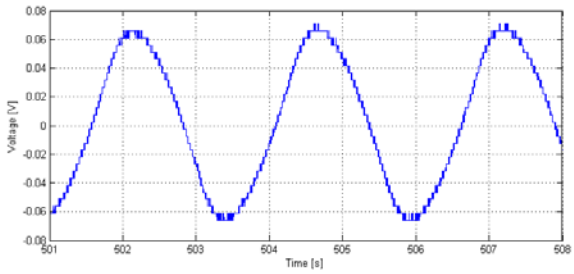

Fig. 17. Limit cycle obtained using the closed loop system with relay and dead time

As in the case of the relay feedback method, in order to prove the correctness of the implemented method the signal generated from the PC using the digital to analog converter of the acquisition card and the output of the process are captured using an oscilloscope. In Fig. 18, corresponding to the Scicos model presented in Fig. 15, are shown in the upper part of the snapshot the output of the relay generated with the *Comedi D/A* block and in the lower part of the snapshot the limit cycle corresponding to the desired frequency, which represents the output of the process implemented on the analog computer.

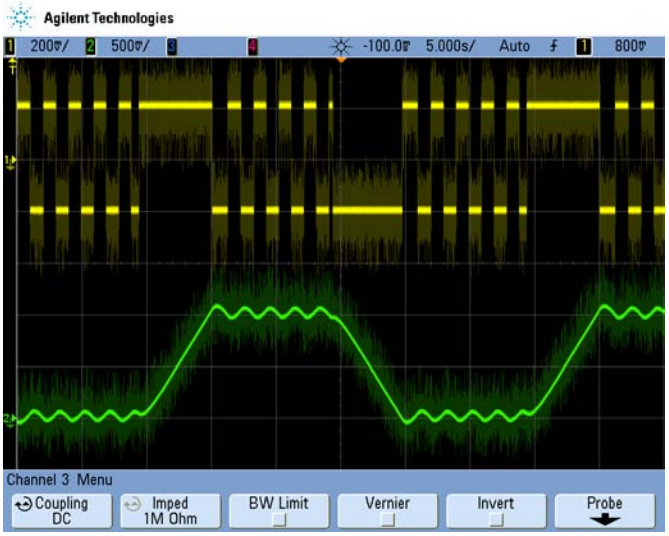

Fig. 18. The control signal (the output of the relay) and the output of the process for a closed loop system with relay and dead time

From the obtained limit cycle the phase and the module of the process are estimated and s_p is computed using (6). The parameters of the controller are computed using (7). The response of the closed loop system with the determined PID controller, at a rectangular reference signal is presented in Fig. 19.

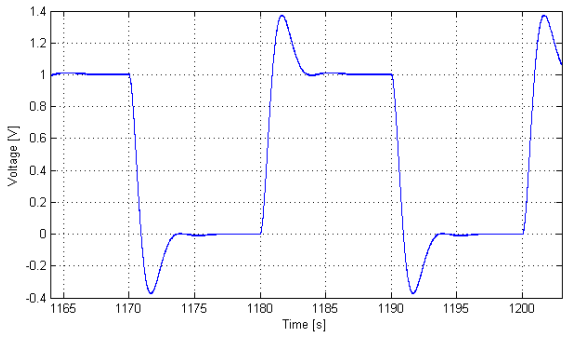

Fig. 19. The response of the closed loop system using a PID controller tuned with the robust tuning method

In Fig. 20, corresponding to the closed loop Scicos model presented in Fig. 4, are shown in the upper part of the snapshot the output of the PID controller which represents the control signal and in the lower part of the snapshot the closed loop response of the process to a rectangular reference signal.

The response time of the closed loop system is 7 seconds, and the overshoot is 37%. Using this method the response time decreases and the overshot slightly increases. The robust tuning method assures a faster response time compared to the relay feedback tuning method.

7. FOPID CONTROLLER

This section describes the *FOPID* block implemented in the RCP toolbox presented in this paper.

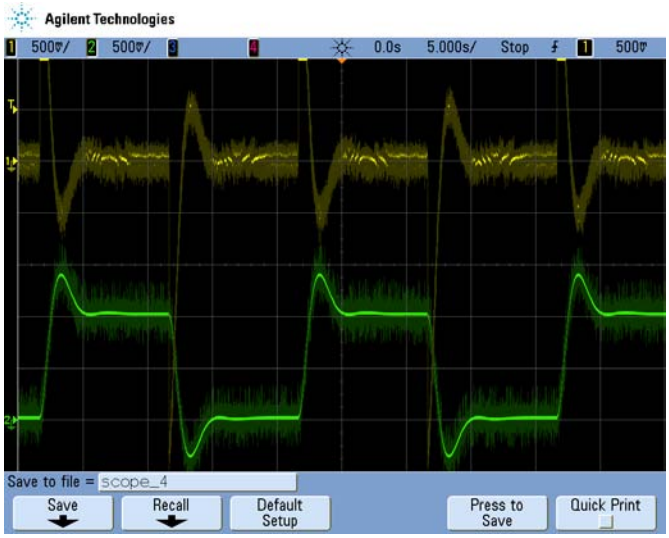

Fig. 20. The control signal and the output of the process for a closed loop system with a PID controller tuned with the robust tuning method

In the literature there are very few papers which present practical implementation of fractional order PID controllers and the control of a real process using such a controller. This difficulty is caused by the mathematical nature of the fractional order operators which are defined by convolution and require "unlimited" memory. The paper (Petras et al., 2003) presents some practical aspects of the implementation of a FOPID controller on a microcontroller.

In the case of the FOPID controller implemented in this paper, due to the fact that the generated real-time executable runs on the processor of a PC, the memory constraints are not so critical, and thus an optimal implementation of the fractional order PID controller is possible.

7.1 FOPID controller implementation

Fractional calculus is a generalization of integration and differentiation to non-integer order fundamental operator ${}_aD_t^{\alpha}$, $\alpha \in \Re$, where *a* and *t* are the limits of the operation. Podlubny (1999) proposed the differential equation for $PI^{\lambda}D^{\mu}$ FOPID:

$$u(t) = K_p e(t) + K_i D_t^{-\lambda} e(t) + K_d D_t^{\mu} e(t), \qquad {}_aD_t^{\alpha} \equiv {}_0D_t^{\alpha}$$

where K_p is the proportional gain, $K_i = K_p / T_i$ is the integration gain, T_i is the integration time, $K_d = K_p T_d$ is the derivative gain, T_d is the derivative time, λ and μ are the integral and the derivative orders respectively.

For an accurate implementation of a FOPID algorithm all the past errors should be memorized. There are two discretization methods used for FOPID controllers: direct discretization and indirect discretization. In indirect discretization methods, frequency domain fitting in continuous time domain is performed first and afterwards the fit s-transfer function is discretized. Several direct discretization methods by finite differences or differential equations were proposed in recent researches, such as: Short memory principle, Tustin Expansion, Lagrange function interpolation method (Chen and More, 2002). Derived from Grunwald-Letnikov definition, the numerical calculation formula for fractional derivative can be achieved as:

$${}_{t-L}D_t^{\alpha} x(t) \approx T_s \sum_{j=0}^{[L/Ts]} c_i x(t - jT_s) \tag{8}$$

where L is the length of the memory and T_s is the sampling time. The weighting coefficient c_j can be calculated recursively by:

$$c_j = (1 - (1 + \lambda)/j) c_{j-1} \tag{9}$$

Using (8) the discrete equation for the FOPID controller is obtained:

$$u_n = K_p e_n + K_i T_s^{\lambda} \sum_{j=0}^{n} q_j e_{n-j} + K_d T_s^{-\mu} \sum_{j=0}^{n} d_j e_{n-j} \tag{10}$$

where the weighting coefficients q_j and d_j are calculated using (9).

For the FOPID controller the following parameters have to be chosen: K_p proportional gain; T_i integration time; T_d derivative time; $u_{\min}$ minimum saturation value; $u_{\max}$ maximum saturation value; λ fractional integral coefficient; μ fractional derivative coefficient; and T_s sampling time. In Fig. 21 is presented the *Set Block properties* window for setting the parameters of the FOPID controller.

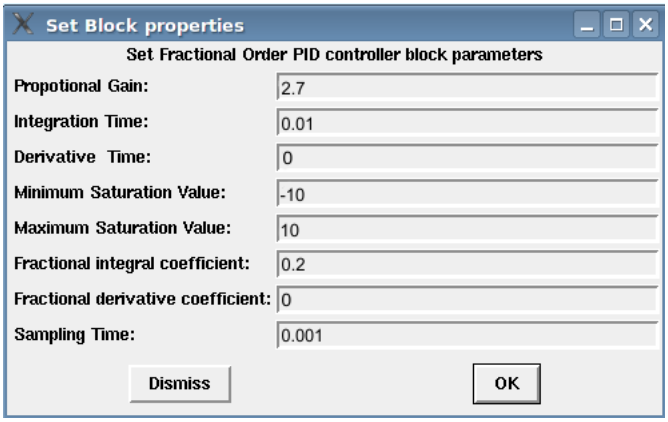

Fig. 21. Window for setting the parameters of the FOPID controller

Equation (10) is used for the implementation of the computation function of the FOPID controller block.

7.2 FOPID controller tuning

The FOPID controller, has five parameters which can be used to tune the controller, thus a higher flexibility can be achieved, than in the case of a classical PID controller. Due to this reason we expect to obtain with the FOPID controller better closed loop performances that the ones obtained with the two previously determined PID controllers.

A FOPID controller is tuned for the process defined in (1). Many papers related to tuning methods for FOPID controllers are published in the literature. Although a simple tuning rule, as in the case of PID controllers, does not exists. Barbosa (Barbosa et al., 2008) proposed an experimental method for tuning FOPID controllers. The starting point are the parameters determined by using Ziegler-Nichols methods. The parameters of the controller are varied until a satisfactory system response is obtained. In order to tune the parameters of the FOPID controller this method is used and the closed loop Scicos model presented in Fig. 22 is implemented.

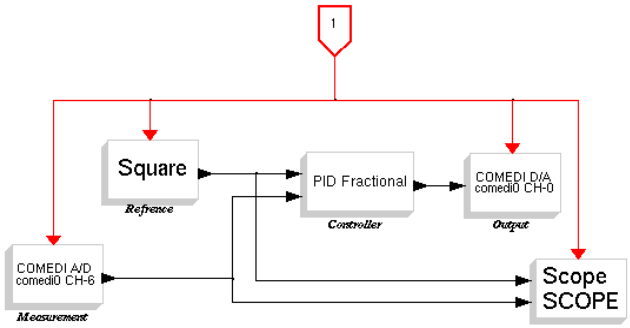

Fig. 22. Scicos model for closed loop control using FOPID controller

The step response of the closed loop system, with the determined FOPID controller, is shown in Fig. 23. The data is logged using the virtual oscilloscope.

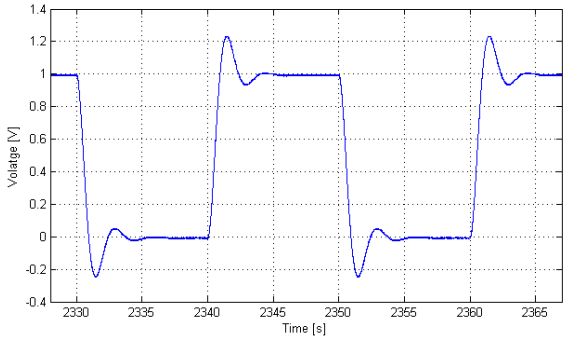

Fig. 23. Closed loop step response using the determined FOPID controller

In Fig. 24 are measured using a digital oscilloscope the output of the FOPID controller, which represents the control signal, in the upper part of the snapshot and in the lower part of the snapshot the closed loop response of the process to a rectangular reference signal.

The response time of the closed loop system is 5.5 seconds, and the overshoot is 22%. With the FOPID controller both the response time and the overshot decreased. Thus the FOPID controller outperforms the two PID controllers tuned in the previous sections of this paper.

8. CONCLUSIONS

The paper presented the practical implementation of three tuning methods. The process is implemented on an analog computer and for the implementation of the control system rapid control prototyping is used. The generated code runs in real-time on the processor of a PC. An AT-MIO-16E-10 acquisition card is used as input/output interface with the controlled process.

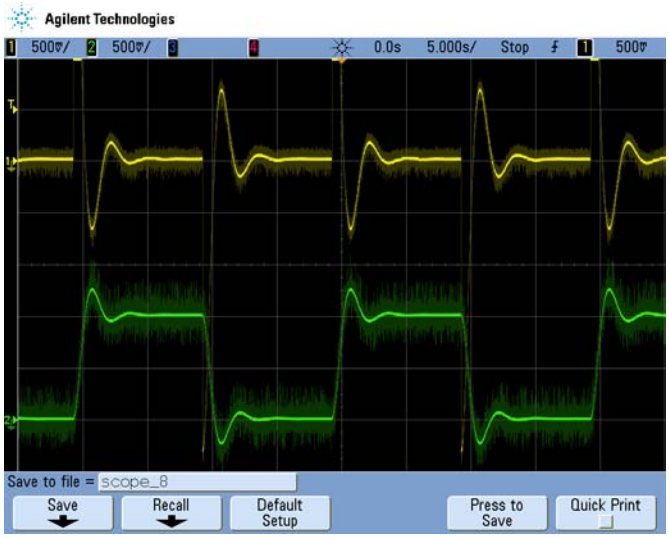

Fig. 24. The control signal and the measured output signal for a closed loop system with FOPID controller

A contribution of this paper is the implementation of a rapid control prototyping toolbox for Scilab/Scicos which generates real-time C code for Linux-RTAI. The real-time generated code is not dependent on the acquisition card used for input/output interface. A discrete implementation for PID and FOPID controllers is presented. This toolbox is a low cost and flexible RCP solution. The main disadvantage of this RCP approach is that the process of compiling a new Linux kernel and installing all the necessary software packages requires advanced Linux operating system knowledge.

Another contribution is the implementation of a Scicos block for the FOPID controller which is used for automatic code generation. Since the real-time generated executable runs on the processor of a PC, the memory constrains are not so critical, and thus an efficient implementation of the fractional order PID algorithm was possible.

Since the process used for controller tuning and closed loop control is implemented on an analog computer the solution presented in this paper is similar to the concept of Processor-In-the-Loop (PIL). In the case of PIL the process is simulated on the PC, using a software like Simulink, and the code runs on the target processor. In the case presented in this paper the real-time code runs on the processor of the PC and for input/output related functionality an acquisition card is used. There are two main advantages of this solution with respect to the classical PIL concept. The first one is the analog implementation of the process which is not numerically simulated. The second one is that the performance of the closed loop control algorithm is not affected by the quantization effect of the analog to digital and digital to analog converters and quantization effect of finite word length.

The implementation effort in the case of the robust tuning method was reduced by using RCP. The obtained practical

results outline the advantages of this tuning method despite the implementation effort.

The process of tuning the parameters of the FOPID controller and the practical implementation of the controller was done using RCP. Taking in to consideration the practical obtained performances we will continue our research work in order to implement a FOPID controller on a microcontroller.

ACKNOWLEDGMENT

This paper was supported by the project "Progress and development through post-doctoral research and innovation in engineering and applied sciences– PRiDE - Contract no. POSDRU/89/1.5/S/57083", project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

REFERENCES

Astrom, K. and Hagglund, T. (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20, 9, 2555−2562.

Barbosa, R. S., Machado, J. A. T., and Jesus, I. S. (2008). On the Fractional PID Control of a Laboratory Servo System. *Proceedings of the 17th IFAC World Congress*, 15273−15278.

Besancon-Voda, A. (2002). Periodic modes analysis in a two-relay feedback system. *Journal of Control Engineering and Applied Informatics*, 4, 3, 47-59.

Bucher, R., Dozio L. and Mantegazza P. (2004). Rapid Control Prototyping with Scilab/Scicos and Linux RTAI. *Scilab Conference*.

Chen, Y. Q. and Moore, K. L. (2002). Discretization schemes for fractional-order differentiators and integrators, *IEEE Trans. on Circuits and Systems*, vol. 49, no. 3, pp. 363-367.

Chen, Y. and Moore, K. L. (2005). Relay feedback tuning of robust PID controllers with iso-damping property, *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, 35, 23-31.

Campbell, S. L., Chancelier, J. P., and Nikoukhah, R. (2009). *Modeling and simulation in Scilab/Scicos with ScicosLab 4.4*, Springer.

Duma, R., Trusca, M., and Dobra, P. (2010). "BLDC Motor Control using Rapid Control Prototyping," in *Journal of Control Engineering and Applied Informatics*, 12, 1, 55-61.

Erkkinen, T. (2003). High-integrity production code generation. *AIAA GN&C Conference*.

Feliu, B. V., Rivas, P. R., Sanchez, R. L., Castilio, G. F. J., and Linarez, S. A. (2008). Robust Fractional Order PI Controller for a Main Irrigation Canal Pool. *Proceedings of the 17th IFAC World Congress*, 15535−15540.

Hagglund, T., Astrom, K. J. (1995) PID Controllers: Theory, Design, and Tuning. *ISA -The Instrumentation, Systems, and Automation Society*, 2nd ed.

Hang, C. C., Astrom, K. J., and Ho, W. K. (1991), Refinements of the Ziegler-Nicholstuning formula, *IEEE Proceedings D Control Theory and Applications*, 138, 111-118.

Hercog, D., Curkovic, M., and Jezernik, K. (2006). DSP Based Rapid Control Prototyping Systems for Engineering Education and Research, *Proceedings of the 2006 IEEE Conference on computer Aided Control Systems Design*, Munich, Germany, October 4-6.

Karimi, A., Garcia, D., and Longchamp, R. (2002). Iterative controller tuning using Bode's integrals. *In Proceedings of the 41st IEEE Conference on Decision and Control*. pages 4227–4232, Las Vegas, Nevada.

Kerhuel, L. (2009). *Simulink - Embedded Target for PIC*.

Machado, J. A. T. (1997). Analysis and Design of Fractional-Order Digital Control Systems. *SAMS Journal of SystemsAnalysis, Modelling and Simulation* 27, 107−122.

Microchip (2009), *MATLAB/Simulink device blocksets for dsPIC DSCs.*

National Instruments (2010). *Getting started with the NI LabVIEW embedded module for ARM microcontrollers*.

Petras, I., Dorcak, L., and Kostial, I. (1998). Control quality enhancement by fractional order controllers. *Acta Montanistica Slovaca* 2, 143−148.

Petras, I., Grega, S., and Dorcak, L.,(2003). Digital fractional order controllers realized by PIC microprocessor: Experimental results.

Podlubny, I. (1999). Fractional-Order Systems and PID controllers. *IEEE Transactions on Automatic Control* 44 (1), 208−214.

Ravn, O. (2006). "Adaptive control using the adaptive toolbox-TAT for Scilab/Scicos", *14th IFAC Symposium on System Identification*, Newcastle, Australia.

Rebeschiess, S. (1999)."MIRCOS- microcontroller-based real time control system toolbox for use with Matlab/Simulink". *Proc. IEEE Int. Symp.Computer Aided Control System Design*, pp. 267-272.

Xue, D., Zhao, C., Chen, Y. Q. (2006). Fractional order PID control of a DC motor with phase elastic shaft: a case study. *Proceedings of the 2006 American Control Conference*, 3182−3187.

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

([Evidence]) Evidence. URL *http://www.comedi.org*.www.evidence.eu.com/content/view/175/216/.