DESIGNING AND TESTING OF MODERN TOOLS FOR CONTROL SYSTEMS PROTOTYPING

Petru Dobra, Mirela Truşcă and Daniel Moga

Technical University of Cluj, Systems Theory Laboratory
St. C. Daicoviciu 15, RO-400020 Cluj-Napoca, ROMANIA
Phone: ++40-264-401433; Fax: ++40-264-592055
Email: {dobra, Mirela.Trusca, Daniel.Moga}@aut.utcluj.ro

Abstract: With the great diversity of applications, a development environment must be flexible and provide exactly the functionality necessary for efficient problem solving. Throughout the whole development process, it must be working with the same MATLAB/Simulink/Stateflow environment and with the same graphical user interfaces, test scripts and parameter sets. In all the development steps, there are off-the-shelf components that make the development task more convenient than ever before. Model-based control design is a highly efficient and widely established approach enabling control engineers to work with a single, visualized system model in an integrated software environment. Experimental results with DC motor illustrate the effectiveness and the simplicity of this development environment of robust PID controller design. *Copyright ©2005 IFAC*

Keywords: PID controller, PID tuning, auto tuning, Rapid Control Prototyping (RCP), relay feedback test.

1. INTRODUCTION

Developing controllers for applications (electrical drive systems) means large expenditure, when performed with usual development methods. The workload comprises development of a mathematical model as well as algorithm design and implementation, off-line simulation, and optimization. The whole process has to be restarted on occurring errors or divergences, which makes the development process timeconsuming and costly.

Rapid Control Prototyping (RCP) is a way out of this situation, especially if the control algorithm is complex and a lot of iteration steps are necessary (Hanselmann and Schütte, 2001), (Vater, 1997) and (Venugopal *et al.*, 2002). Intelligent software and hardware tools relieve the control engineer from cumbersome hand coding. The need to make use of these tools grows with the complexity of the control system to develop.

The RCP tool presented in this paper is based on MATLAB/Simulink, a widely used control development software. It allows to design the controller graphically in the Simulink block diagram environment. Using the Real-Time Interface (RTI) to Simulink the control algorithms are downloaded to a real-time prototyping system, which replaces the handmade prototype and executes the algorithm. This way, changes to the designed controller are much easier to perform and iteration step times are reduced to a minimum.

Hardware for RCP has to be much more powerful than the target controller, especially when complex controller functions shall be implemented. Moreover, I/O of the prototyping system ideally corresponds to the I/O of the target controller. The hardware introduced in this paper is tailored for applications in the fields of electrical drives and provides both, high computational power and a wide range of powerful I/O.

2. REVIEW OF PID CONTROLLERS

The industrialist had concentrated on PI/ PID controllers and had already developed *one-button type* relay auto-tuning techniques for fast, reliable PI/ PID control yet with satisfactory performance (Åström and Hägglund, 1984), (Hang *et al.*, 1991), (Hägglund and Åström, 1995) and (Leva, 1993). Although many different methods have been proposed for tuning PID controllers, until today, the Ziegler-Nichols method (Hang *et al.*, 1991) is still extensively used for determining the parameters of PID controllers. The design is based on the measurement of the critical gain and critical frequency of the plant and using simple formulae to compute the controller parameters. In 1984, Åström and Hägglund (Åström and Hägglund, 1984) proposed an automatic tuning method based on a simple relay feedback test, which uses the describing function analysis to give the critical gain and the critical frequency of the system. This information can be used to compute a PID controller with desired gain and phase margins. In relay feedback tests, it is a common practice to use a relay with hysteresis (Leva, 1993) for noise immunity. Another commonly used technique is to introduce an artificial time delay within the relay closed-loop system, e.g., (Leva, 1993), to change the oscillation frequency in relay feedback tests.

After identifying a point on the Nyquist curve of the plant, the so-called modified Ziegler-Nichols method (Åström and Hägglund, 1984) and (Hägglund and Åström, 1995) can be used to move this point to another position in the complex plane. Two equations for phase and amplitude assignment can be obtained to retrieve the parameters of a PI controller. For a PID controller, however, an additional equation should be introduced. In the modified Ziegler-Nichols method, α, the ratio between the integral time T_i and the derivative time T_d, is chosen to be constant, i.e., $T_i = \alpha T_d$, in order to obtain a unique solution.

The control performance is heavily influenced by the choice of α as observed in (Leva, 1993). Recently, the role of α has drawn much attention, e.g., (Wallén *et al.*, 2002). For the Ziegler-Nichols PID tuning method, α is generally assigned as a magic number $4(four)$ (Hägglund and Åström, 1995). Wallén, Åström and Hägglund proposed that the tradeoff between the practical implementation and the system performance is the major reason for choosing the ratio between T_i and T_d as four (Wallén *et al.*, 2002).

The main contribution of this paper is the use of a new tuning rule which gives a new relationship between T_i and T_d in stead of the equation $T_i = 4T_d$ proposed in the modified Ziegler-Nichols method (Hang *et al.*, 1991) and (Wallén *et al.*, 2002). We propose to add an extra condition that the phase Bode plot at a specified frequency ω_c at the point where sensitivity circle touches Nyquist curve is locally flat which implies that the system will be more robust to gain variations (Bode, 1945). This additional condition can be expressed as $\left.\frac{d\angle L(s)}{ds}\right|_{s=j\omega_c} = 0$, which can be equivalently expressed as

$$\angle \left.\frac{dL(s)}{ds}\right|_{s=j\omega_c} = \angle L(s)|_{s=j\omega_c} \qquad (1)$$

where ω_c is the frequency at the point of tangency and $L(s) = G(s)K(s)$ is the transfer function of the open loop system including the controller $K(s)$ and the plant $G(s)$. In this paper, we consider the PID controller of the following form:

$$K(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right). \qquad (2)$$

This "flat phase" idea proposed above is illustrated in Fig. 1(a) where the Bode diagram of the open loop system is shown with its phase being tuned locally flat around ω_c. We can expect that, if the gain increases or decreases a certain percentage, the phase margin will remain unchanged. Therefore, in this case, the step responses under various gains changing around the nominal gain will exhibit an isodamping property, i.e., the overshoots of step responses will be almost the same. Fig. 1(b) can also explain this where the sensitivity circle touches the Nyquist curve of the open loop system at the flat phase point. Clearly, since gain variations are unavoidable in the real world due to possible sensor distortion, environment change etc., and the iso-damping is a desirable property, which ensures that no harmful excessive overshoot is resulted due to gain variations.To obtain a desired phase margin γ_k at the crossover frequency ω_c we have the following equations to solve:

$$\angle L(s)|_{s=j\omega_c} = \gamma_k - \pi. \qquad (3)$$

$$|L(j\omega_c)| = 1. \qquad (4)$$

With these two conditions (3) and (4) and the new condition (1), all the three parameters of PID controller can be calculated.

As in the Ziegler-Nichols method, T_i and T_d are used to tune the phase condition (3) and K_p is determined by the gain condition (4). However, the condition (1) gives a relationship between T_i and T_d instead of $T_i = \alpha T_d$.

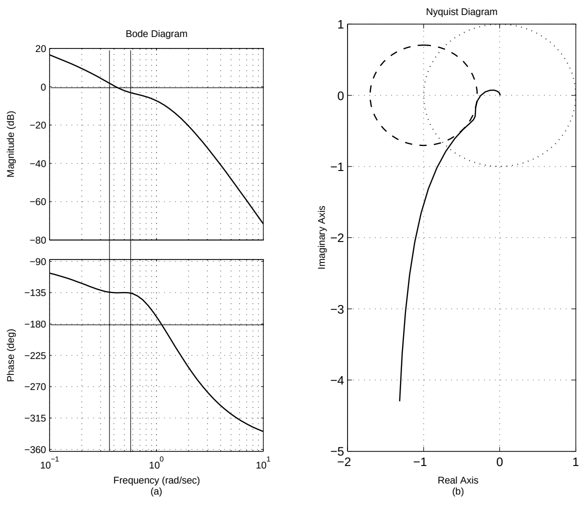

Fig. 1. Illustration of the basic idea for robust PID tuning

In this new tuning method, ω_c is not necessarily the gain crossover frequency. The ω_c is the frequency at which the transfer locus (Nyquist curve) tangentially touches the sensitivity circle. Similarly, γ_k, the tangent phase, is not necessarily the phase margin usually used in previous PID tuning methods. According to (Hägglund and Åström, 1995), the phase margin is always selected from 30 deg to 60 deg. Due to the flat phase condition (1), the derivative of the phase near ω_c will be relatively small. Therefore, if γ_k is selected to be around 30 deg, such as 35 deg, the phase margin will be generally within the desired interval.

3. PID CONTROLLER DESIGN

Frequency response (substitute s by $j\omega$) of open loop system can be written as $L(j\omega) = G(j\omega)K(j\omega)$, where

$$K(j\omega) = K_p\left(1 + \frac{1}{j\omega T_i} + j\omega T_d\right) \tag{5}$$

is the PID controller obtained from (2). The phase of the open loop system is given by

$$\angle L(j\omega) = \angle G(j\omega) + \angle K(j\omega). \tag{6}$$

The derivative of the open loop system $L(j\omega)$ with respect to ω can be written as follows:

$$\frac{dL(j\omega)}{d\omega} = \frac{dG(j\omega)}{d\omega}K(j\omega) + G(j\omega)\frac{dK(j\omega)}{d\omega}. \tag{7}$$

From (1), the phase of the derivative of the open loop system can not obviously be obtained directly from (7). So, we need to simplify (7). The derivative of the controller with respect to ω is

$$\frac{dK(j\omega)}{d\omega} = jK_p\left(T_d + \frac{1}{\omega^2 T_i}\right). \tag{8}$$

To calculate $\frac{dG(j\omega)}{d\omega}$, since we have

$$\ln(G(j\omega)) = \ln|G(j\omega)| + j\angle G(j\omega).$$

Differentiating with respect to ω gives

$$\begin{aligned}\frac{d\ln(G(j\omega))}{d\omega} &= \frac{1}{G(j\omega)}\frac{dG(j\omega)}{d\omega} = \\ &= \frac{d\ln|G(j\omega)|}{d\omega} + j\frac{d\angle G(j\omega)}{d\omega}.\end{aligned}$$

Straightforwardly, we arrive at

$$\frac{dG(j\omega)}{d\omega} = G(j\omega)\left(\frac{d\ln|G(j\omega)|}{d\omega} + j\frac{d\angle G(j\omega)}{d\omega}\right). \tag{9}$$

Substituting (5), (8) and (9) into (7) gives

$$\begin{aligned}\frac{dL(j\omega)}{d\omega} &= K_p G(j\omega)\left\{j\left(T_d + \frac{1}{\omega^2 T_i}\right) + \left[1 + \cdots\right.\right. \\ &\left.\left. +j\left(\omega T_d - \frac{1}{\omega T_i}\right)\right]\left(\frac{d\ln|G(j\omega)|}{d\omega} + j\frac{d\angle G(j\omega)}{d\omega}\right)\right\}\end{aligned} \tag{10}$$

By combining (1), (6) and (10), one obtains the relationship between T_i and T_d at ω_c:

$$T_d = \frac{-T_i\omega_c + 2s_p(\omega_c) + \sqrt{\Delta}}{2s_p(\omega_c)\,\omega_c^2 T_i}, \tag{11}$$

where

$$\Delta = T_i^2\omega_c^2 - 8s_p(\omega_c)T_i\omega_c - 4T_i^2\omega_c^2 s_p^2(\omega_c), \tag{12}$$

and

$$s_p(\omega_c) = \omega_c \times \left.\frac{d\angle G(j\omega)}{d\omega}\right|_{\omega=\omega_c}. \tag{13}$$

The approximation of s_p for stable and minimum phase plant can be given as follows (Karimi *et al.*, 2002):

$$s_p(\omega_c) \simeq \angle G(j\omega_c) + \frac{2}{\pi}\left(\ln(K_g) - \ln|G(j\omega_c)|\right) \tag{14}$$

where $K_g = |G(s)||_{s=0} = |G(0)|$ is the static gain of the plant, $\angle G(j\omega_c)$ is the phase and $|G(j\omega_c)|$ is the gain of the plant at the specific frequency ω_c. It is obvious that T_i and T_d are related by s_p alone. For this new tuning method, s_p includes all the information that we need of the unknown plant.

Suppose that we have known s_p at ω_c. How to experimentally measure $s_p(\omega_c)$ will be discussed in the next section based on the measurement of $\angle G(j\omega)$ and $|G(j\omega)|$.

The formulae for K_p, T_i and T_d, let us summarize what are known at this point. We are given:

i) ω_c, the desired tangent frequency;
ii) γ_k, the desired margin phase;
iii) measurement of $\angle G(j\omega_c)$ and $|G(j\omega_c)|$;
iv) the estimation of $s_p(\omega_c)$.

Furthermore, using (3), (4) and 11, the PID controller parameters can be set as follows:

$$K_p = \frac{\cos(\gamma_k - \angle G(j\omega_c) - \pi)}{|G(j\omega_c)|}, \tag{15}$$

$$T_i = \frac{-2}{\omega_c\left(\frac{s_p(\omega_c)}{\cos^2\left(\gamma_k - \angle G(j\omega_c)\right)} + \tan\left(\gamma_k - \angle G(j\omega_c)\right)\right)}, \tag{16}$$

$$T_d = \frac{1 + \omega_c T_i \tan\left(\gamma_k - \angle G(j\omega_c)\right)}{\omega_c^2 T_i}. \tag{17}$$

The selection of ω_c heavily depends on the system dynamics. For most of plants, there exists an interval for the selection of ω_c to achieve flat phase condition. If no better idea about ω_c, the desired cutoff frequency can used as the initial value. For γ_k, a good choice is within 30 deg to 35 deg.

3.1 Relay Feedback Tests

The parameters of a PID controller can be calculated straightforwardly if we know $\angle G(j\omega_c)$, $|G(j\omega_c)|$ and $s_p(\omega_c)$.

As indicated in (14), $s_p(\omega_c)$ can be obtained from the knowledge of the static gain $|G(0)|$, $\angle G(j\omega_c)$ and $G(j\omega_c)$. The static gain $|G(0)|$ or K_g is very easy to measure and it is assumed to be known. The *relay feedback* test, shown in Fig. 2, can be used to "measure" $\angle G(j\omega_c)$ and $G(j\omega_c)$. In the relay feedback experiments, a relay is connected in closed-loop with the unknown plant as shown in Fig. 2, which is usually used to identify one point on the Nyquist diagram of the plant. To change the oscillation frequency due to relay feedback, an artificial time delay is introduced in the loop. The artificial time delay τ is the tuning knob here to change the oscillation frequency. Our problem here is how to get the right value of τ which corresponds to the tangent frequency ω_c. To solve this problem, an iterative method can be used as summarized in the following:

i) Start with the desired tangent frequency ω_c.

ii) Select two different values (τ_{-1} and τ_0) for the time delay parameter properly and does the relay feedback test twice. Then, two points on the transfer locus (Nyquist curve) of the plant can be obtained. The frequencies of these points can be represented as ω_{-1} and ω_0, which correspond to τ_{-1} and τ_0, respectively. The iteration begins with these initial values (τ_{-1}, ω_{-1}) and (τ_0, ω_0).

iii) With the values obtained in the previous iterations, the artificial time delay parameter τ can be updated using a simple interpolation/extrapolation scheme as follows:

$$\tau_k = \frac{\omega_c - \omega_{k-1}}{\omega_{k-1} - \omega_{k-2}}\left(\tau_{k-1} - \tau_{k-2}\right) + \tau_{k-1}$$

where k represents the current iteration number. With the new τ_k, after the relay test, the corresponding frequency ω_k can be recorded.

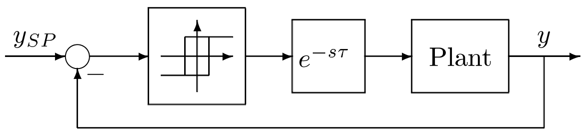

Fig. 2. Relay feedback system

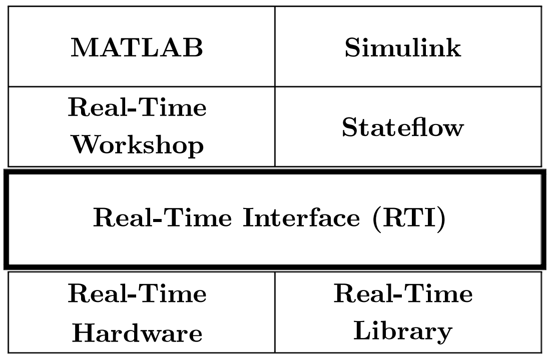

Fig. 3. The RTI in MATLAB/Simulink

iv) Compare ω_k with ω_c. If $|\omega_k - w_c| < \varepsilon$, quit iteration. Otherwise, go to *Step iii)*. Here, ε is a small positive number.

The iterative method proposed above is feasible because in general the relationship between the delay time τ and the oscillation frequency ω is one-to-one.

After the iteration, the final oscillation frequency is quite close to the desired one ω_c so that the oscillation frequency is considered as ω_c. Hence, the amplitude and the phase of the plant at the specified frequency can be obtained. Using (14), one can calculate the approximation of $s_p(\omega_c)$.

4. CONTROLLER IMPLEMENTATION FROM BLOCK DIAGRAMS

4.1 Real-Time Interface to Simulink

Using MATLAB and Simulink for modeling, analysis, design and offline simulation has become a de-facto standard for control system development. The Real-Time Interface (RTI) enhances the Simulink block library with additional blocks, which provide the link between Simulink and the real-time hardware (Fig. 3). These blocks cover the I/O functionality of the prototyping hardware.

To graphically specify an I/O channel the corresponding block icon has to be picked up from the I/O block library and attached to the Simulink controller model. I/O parameters, such as voltage ranges or resolutions, can be set in appropriate dialog boxes. Thus, even complex I/O devices such as incremental encoders or a CAN interface can be configured in the block diagram. For multitasking applications, a pre-emptive scheduler guarantees real-time behavior with response times of a few microseconds. Tasks and priorities are also defined graphically within the Simulink block diagram.

The Simulink model then is transferred into real-time code, using the Real-Time Workshop and the RTI. Code generation includes the I/O channel specification and the multitasking setup, which are translated into appropriate function calls of the Real-Time Library. This library is a C function library providing a high-level programming interface to the hardware.

4.2 Simulink Block Library for DS1102

The block library for the DS1102 DSP Controller Board comprises all I/O units that are directly served by the TMS320C31. Block icons for the standard I/O channels such as A/D, D/A converters, and digital I/O are included, as well as the more complex incremental encoder blocks. An overall encoder setup block provides a comfortable means for defining the parameters of the encoder subsystem, such as counter ranges and signal types. A further block is available for obtaining encoder outputs for position measurement and delta position values for speed measurement. The remaining blocks are used for defining encoder zero positions and for index search. Apart from the master setup block, all block icons can be duplicated to handle multiple encoder channels.

4.3 Multitasking Implementations

Multitasking is supported by Simulink's capability to define the sample rate of timer-driven blocks and to set up triggered and enabled subsystems. A special hardware interrupt block is used to select one of the interrupt sources available on the board as the trigger signal for a subsystem.

An important feature of the Real-Time Interface is the ability to use I/O block icons in any subsystem and, for timer-driven subsystems, with any sample rate. Clearly, the same I/O channel can be used only once. All interrupt-triggered subsystems and model parts with different sample rates are handled as different tasks. The priority setting dialog displays all tasks of the model for assigning priorities. Timer-driven tasks always follow the rate-monotonic scheme: tasks with higher sample rates get higher priorities.

5. CONTROL OF A DC MOTOR

As a typical application, the control of a DC motor is presented for verification of the proposed techniques. The squirrel cage DC motor comprises

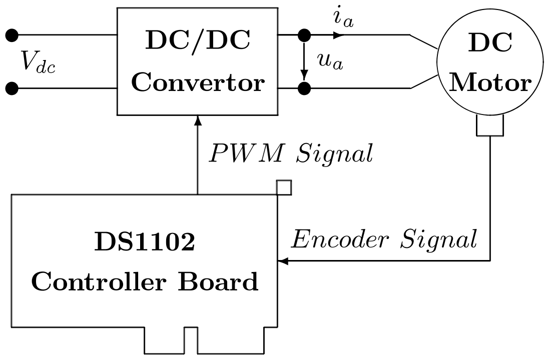

Fig. 4. DC motor control setup with DS1102

the motor, a digital encoder for position and velocity measurement, and a DC/DC electronic converter. The DC/DC electronic converter is driven by pulse width modulated (PWM) signals from the DS1102. The whole test setup is shown in Fig. 4. It is described in greater details in (Squirrel, 1997) and (Trzynadlowski, 1994). The control theory of DC motor is out of the scope in this paper. A deeper view into this topic is presented in (Vater, 1997).

Main purpose of the control scheme presented here is to run the motor at a given velocity. AD conversion and digital encoder counting are performed by the I/O devices on-board. The PWM signal generation requires high time resolution.

The system can easily be set up by exploiting some of the new features available with the latest version of the RTI. The execution of the outmost control layer is triggered by this hardware interrupt generated at the beginning of every PWM period. Receiving the hardware interrupt is utilized by a standard unit block supplied with the RTI.

The output variable is written into an I/O block controlling PWM actions. Input to the main controller block is the desired velocity speed selected by the user, the actual velocity speed (ω). All values are readed in by standard I/O blocks.

5.1 Instrumentation and Data Acquisition

After automatic code generation for the control application, the executable is downloaded to the processor board. At this stage, all task assignments to processor are automatically made by the software. Once the code is loaded, the drive can be controlled by the real-time application. To select operation modes and adjust controller and user variables, the COCKPIT tool with an instrumentation panel as shown in Fig. 5 is used.

With system based on relay feedback (Fig. 5) and algorithm presented in Section 2.1, results the stable oscillation ($A_{osc} = 0.44$ and $T_{osc} = 0.54$ sec). The PID controller designed by using the proposed tuning formulae is

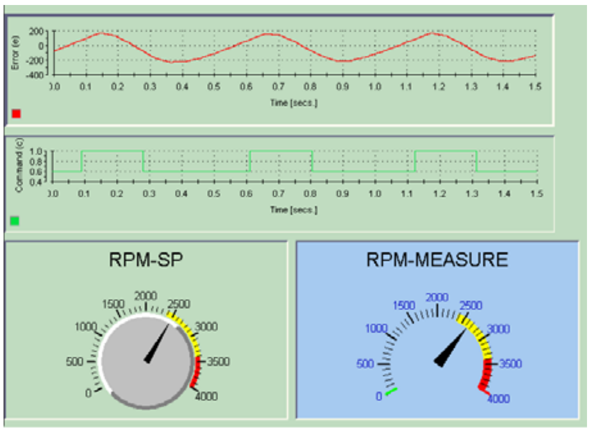

Fig. 5. Instrumentation control panel and data acquisition configuration for relay-based of DC Motor

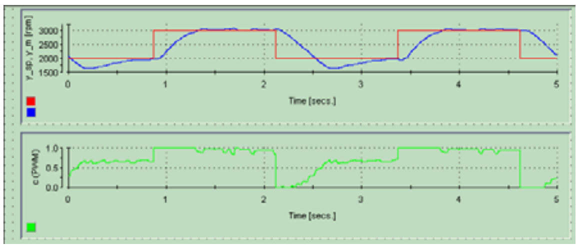

Fig. 6. Step response with PID Controller

$$K_{PID}(s) = 3.46\left(1 + \frac{1}{0.28s} + 0.26s\right).$$

With this PID controller the step response of closed-loop .are shown in Fig. 6.

With this setup the motor can be run with several signal forms as well as with arbitrary static values. The motor velocity is observed among other properties, and main controller parameters are accessible for adjustment. For data acquisition, the TRACE tool is used as also shown in Fig. 6. It allows for data captures of arbitrary model data in real-time utilizing trigger functions, and time plotting as well as trajectory plotting capabilities. In the given trace capture, a comparison of desired and measured drive velocity are shown.

6. CONCLUSION

Rapid Control Prototyping for fast drive systems requires hardware with high-performance floating-point processor and an optimized range of I/O devices. The system presented in this paper provides both. The computing power of the DSP allows for the calculation of large-scale floating-point control algorithms in real-time. Incremental encoder interfaces and PWM outputs make the board a powerful tool for Rapid Control Prototyping especially for electrical drive applications.

In this paper it was presented a drives application, in which a new PID tuning method is proposed for a class of unknown, stable and minimum phase plants. Experimental application for a DC motor illustrates the effectiveness and the simplicity of the proposed method for robust PID controller design. The control algorithm runs on the DSP (TMS320C31) and contains parts, which have to be synchronized with the I/O signals. The slave DSP (TMS320C14), which in addition generates the necessary, interrupts for synchronization, processes these I/O signals.

REFERENCES

Åström, K.J. and T. Hägglund (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica* **20**(5), 645–651.

Bode, H.W. (1945). *Network Analysis and Feedback Amplifier Design*. Van Nostrand, New York.

Hägglund, T. and K.J. Åström (1995). *PID Controllers: Theory, Design, and Tuning*. ISA Society, London.

Hang, C.C., K.J. Åström and W.K. Ho (1991). Refinements of the Ziegler-Nichols tuning formula. *IEEE Proceedings Part-D* **138**(2), 111–118.

Hanselmann, H. and F. Schütte (2001). Control system prototyping, productionizing and testing with modern tools. In: *International Conference on PCIM, Nürnberg*. pp. 121–131.

Karimi, A., D. Garcia and R. Longchamp (2002). PID controller design using Bode's integrals. In: *American Control Conference, Anchorage, Alaska, USA*. pp. 5007–5012.

Leva, A. (1993). PID autotuning algorithm based on relay feedeback. *IEEE Proceedings Part-D* **140**(5), 328–338.

Squirrel, N.N. (1997). Cage induction motor control with DS1102. *Application Note*. dSPACE BmbH. Paderborn.

Trzynadlowski, A.M. (1994). *The Field Orientation Principle in Control of Induction Motors*. Kluver Academic Publishers, Dordrecht.

Vater, J. (1997). The need for and the principle of high resolution incremental encoder interfaces in rapid control prototyping. In: *International Conference on PCIM, Nürnberg*. pp. 1–8.

Venugopal, R., M. Beine and A. Ruekgauer (2002). Real-time simulation of adaptive suspension control using dSPACE control development tools. *International Journal of Vehicle Design* **29**(1/2), 128–138.

Wallén, A., K.J. Åström and T. Hägglund (2002). Loop-shaping design of PID controllers with constant T_i/T_d ratio. *Asian Journal of Control* **4**(4), 403–409.