Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Teaching PID Tuning with IMC Design for Dynamic Systems using ScicosLab

Aline A. Franca*, Antonio S. Silveira*, Antonio A. R. Coelho*, Francisco J. Gomes, Carlos B. Meza*****

** Federal University of Santa Catarina, Department o f Automation and Systems, 88040900, Florianópolis, SC, Brazil (e-mail: aline@das.ufsc.br, toninho@das.ufsc.br, aarc@das.ufsc.br)*
*** Federal University of Juiz de Fora, Faculty of Engineering, 36036-330, Juiz de Fora, MG, Brazil, (e-mail: chico.gomes@ufjf.edu.br)*
**** Costa Rica Institute of Technology, Electronics Engineering Department, Cartago, Costa Rica, (e-mail: cmeza@ietec.org)*

Abstract: In industrial processes a control loop problem is how to choose a suitable set of PID parameters because of their influence on the system asymptotic stability. This paper aims to show the advantages of using Free Open Source Software in control education. A PID control algorithm based on Internal Model Control design for linear plants, in the discrete-time domain, is derived. Some examples for simulating typical control systems and a real-time physical plant are given to explore not only the controller tuning, closed-loop dynamics, robustness in the presence of practical constraints, time-varying parameters and disturbances, but also the strategy by using ScicosLab as a suitable free software tool to be used for learning in process control and to integrate with the data acquisition board MCC 1208LS.

Keywords: Control education, teaching, software tools, PID control, laboratory, control system analysis.

1. INTRODUCTION

In process control education, computer simulations of dynamic systems considerably help the students to understand and to apply theoretical concepts taught in the classroom. Effectively, teaching the basic controller design procedures (modeling, analysis, controller synthesis, implementation) can not be done only through theoretical lectures with little or no practical essays (Grzegorz et al., 2008; Meza et al., 2009).

Extensive use of numerical simulation examples, case studies and practical experiments using computer-based tools have been implemented at an elective course, Introduction for Identification and Adaptive Control, of the Department of Automation and Systems (DAS) at the Federal University of Santa Catarina. The course encompasses one semester spanning 18 weeks, each one of 3 hours and lectures ranging from: i) classical and advanced modeling, ii) control system design including pole placement, PID, IMC and GMV, iii) auto-tuning and self-tuning techniques for modeling and digital control, iv) numerical and real-time control experiments using ScicosLab, a Free Open Source Software.

Free Open Source Software (FOSS) for control systems has achieved a sufficient maturity such that it can be considered as an alternative to proprietary software to be used in universities both in educational and research environments. ScicosLab is a freely distributed tool that is based on Scilab but with a more stable and powerful version of Scicos (ScicosLab, 2010). Unlike ScicosLab, similar commercial software for use in control simulation environments, such as Matlab and LabView, run into cost issues in universities and industries (Bucher and Balemi, 2005; Coelho, 2010). In addition, the cost of a commercial software package for analysis and implementation of control systems is out of the reach in Latin American universities.

This paper discusses, besides the combination of the Internal Model Control (IMC) design, aspects to tune a digital PID, the numerical implementation of two nonlinear processes and a real-time system essay through the free software ScicosLab as an attempt to use it in control education.

Many methodologies to adjust the gains and to increase PID control performance have been developed in the process control literature. The increasing number of case studies and publications related to PID controller and its hybridization with advanced methods have been reported showing the importance of this type of controller to the industry (Tan and Li, 2001; Åström and Hägglund, 2006; Li et al., 2009).

2. LECTURE OF THE DIGITAL IMC DESIGN

The main idea of IMC is to connect the plant model with the real plant in a parallel form and the controller approaches the model of the inverse plant dynamic. For SISO control systems, IMC uses the inverse of the minimum phase part of the model and adds a low-pass filter to guarantee not only the physical implementation of the control but also the stability and robustness (to modeling errors, ensuring adequate closed-loop behavior for setpoint tracking and disturbance rejection) (Morari and Zafiriou, 1989; Li et al, 2009).

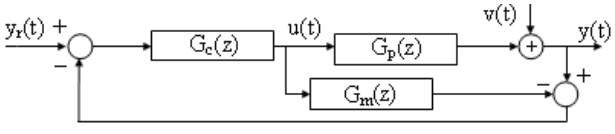

Fig. 1. Structure of the IMC controller.

Assuming that the closed-loop system is stable, the following relationship can be obtained:

$$y(t)=\frac{G_c(z)G_p(z)}{1+G_c(z)\{G_p(z)-G_m(z)\}}y_r(t)+\frac{\{1-G_c(z)G_m(z)\}}{1+G_c(z)\{G_p(z)-G_m(z)\}}v(t) \quad (1)$$

On the assumption of perfect modeling, $G_p(z)=G_m(z)$, and $G_m(z)$ is a minimum phase model, then $G_c(z)=G_m^{-1}(z)$. As can be seen in Fig. 1, there is no output steady-state error in the presence of reference changes and load disturbances. IMC control structure shows better dynamic response and robustness compared with traditional feedback control.

Next, discrete IMC design and its relation with PID tuning for SISO systems is derived. First, it is assumed a CAR (Controlled Auto-Regressive) process model of the form

$$A(z^{-1})y(t)=z^{-k}B(z^{-1})u(t) \quad (2)$$

where $y(t)$ is the system output, $u(t)$ is the control signal, $A(z^{-1})=1+a_1z^{-1}+...+a_{na}z^{-na}$ and $B(z^{-1})=b_0+b_1z^{-1}+...+b_{nb}z^{-nb}$ are related to open-loop poles and zeros, respectively, and $k \geq 1$ is the discrete time-delay. Factorizing (2) as in

$$G_m(z)=G_m^+(z)G_m^-(z)=z^{-k}\frac{B(z^{-1})}{A(z^{-1})} \quad (3)$$

$G_m^+(z)$ is the non-minimum phase part including z^{-k} and $G_m^-(z)$ is the minimum phase part of the plant model. Then, the transfer function of the IMC controller can be obtained as

$$G_c(z)=\frac{1}{G_m^-(z)} \quad (4)$$

Second, a digital low-pass filter $F(z)$ is connected in series with the IMC controller to ensure a causal structure and closed-loop stability. Equation (4) can be rewritten as

$$G_c(z)=\frac{F(z)}{G_m^-(z)} \quad (5)$$

Assuming that $G_p(z)=G_m(z)$, then equation (1) becomes

$$y(t)=G_p(z)G_c(z)y_r(t)+\{1-G_c(z)G_p(z)\}v(t) \quad (6)$$

$$y(t)=F(z)G_m^+(z)y_r(t)+\{1-F(z)G_m^+(z)\}v(t) \quad (7)$$

It can be observed from (7) that IMC provides time-delay compensation, disturbance rejection and the filter imposes the shape of the reference tracking (without offset).

2.1 PID Controller based on IMC Tuning

The IMC system diagram can be represented in an equivalent form for a classical feedback control. Fig. 1 is rearranged in these two following diagrams:

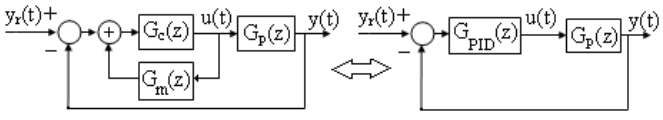

Fig. 2. Relationship between IMC and PID.

Comparing the control system diagrams from Fig. 2, the PID controller assumes the form

$$G_{PID}(z)=\frac{G_c(z)}{1-G_c(z)G_m(z)} \quad (8)$$

A first-order digital filter is selected and represented by

$$F(z)=\frac{b_fz^{-1}}{(1-a_fz^{-1})}=\frac{(1-\alpha)z^{-1}}{(1-\alpha z^{-1})} \quad (9)$$

where $a_f=\alpha=exp(-T_s/\lambda)$, $b_f=(1-a_f)=(1-\alpha)$ and λ is the time constant of the filter that is adjusted on-line by the user to shape the speed of the closed-loop response. As the plant model, it is adopted a discrete second-order model

$$y(t)+a_1y(t-1)+a_2y(t-2)=b_0u(t-k-1)+b_1u(t-k-2) \quad (10)$$

According to the IMC design procedure, the PID controller transfer function is given by

$$G_{PID}(z)=\frac{G_c(z)}{1-G_c(z)G_m(z)}=\frac{\{G_m^-(z)\}^{-1}}{F^{-1}(z)-G_m^+(z)} \quad (11)$$

$$G_{PID}(z)=\frac{b_f}{(b_0+b_1z^{-1})}\frac{(1+a_1z^{-1}+a_2z^{-2})}{(1-a_fz^{-1}-b_fz^{-k-1})} \quad (12)$$

To obtain the PID tuning, the ideal PID controller is considered (Bobál et al., 2005; Åström and Hägglund, 2006)

$$u(t)=K_c\left\{e(t)+\frac{1}{T_i}\int e(t)dt+T_d\frac{de(t)}{dt}\right\} \quad (13)$$

The discrete equation of the PID controller is

$$G_{PID}(z)=\frac{u(t)}{e(t)}=\left[\frac{q_0+q_1z^{-1}+q_2z^{-2}}{(1-z^{-1})}\right] \quad (14)$$

$$q_0=K_c(1+\frac{T_s}{T_i}+\frac{T_d}{T_s})\ ;\ q_1=-K_c(1+2\frac{T_d}{T_s})\ ;\ q_2=K_c\frac{T_d}{T_s} \quad (15)$$

In order to guarantee the PID equation mask, the following simplifications are implemented:

$$b_0+b_1z^{-1}\approx b_0+b_1 \quad (16)$$

$$1-a_fz^{-1}-(1-a_f)z^{-k-1}=(1-z^{-1})\{[1+(1-a_f)z^{-1}+...+(1-a_f)z^{-k}]\big|_{z=1}\}\approx(1-z^{-1})\{1+k(1-a_f)\} \quad (17)$$

that are introduced to ensure the controller gain and to remove the undesirable poles. The transfer function of the PID controller is obtained from (8) as

$$G_{PID}(z)=\frac{b_f}{(b_0+b_1)}\frac{(1+a_1z^{-1}+a_2z^{-2})}{\{1+k(1-a_f)\}(1-z^{-1})} \quad (18)$$

Equations (14) and (18) are related to

$$\bar{K}=\frac{b_f}{(b_0+b_1)\{1+k(1-a_f)\}} \quad (19)$$

$$\bar{K}=K_c(1+\frac{T_s}{T_i}+\frac{T_d}{T_s})\ ;\ \bar{K}a_1=-K_c(1+2\frac{T_d}{T_s})\ ;\ \bar{K}a_2=K_c\frac{T_d}{T_s} \quad (20)$$

and the equations for the PID controller tuning take the form

$$K_c=-\bar{K}(a_1+2a_2)\ ;\ T_i=\frac{-(a_1+2a_2)T_s}{(1+a_1+a_2)}\ ;\ T_d=\frac{-a_2T_s}{(a_1+2a_2)} \quad (21)$$

2.2 Tasks for the PID-IMC Tuning Lab

It is asked to the students to realize the following tasks related to IMC control design: (i) calibrate the closed-loop dynamic of (9), in order to understand how it affects the controlled plant behavior emphasizing industry conservatism needs; ii) set α for the PID tuning using (21), according to design specifications (rise time, settling time, zero steady-state error for setpoint tracking and disturbance rejection); iii) model a transfer function of order greater than two to observe how (16) and (17) affect the gain of the controller and remove undesirable poles of the controller transfer function; iv) simplify the plant model to a discrete first-order equation and to derive a PI controller for the case of slow stable plants.

3. SCICOSLAB FOR SIMULATIONS

Numerical computing software has a key role on control system education not only for the purposes of numerical case studies and real-time applications, but also for the development of synoptic screen for the industry (operation and tuning loops). However, the cost of a commercial software for computer aided control system design for analysis and implementation of control systems is out of the reach for many universities in Latin America.

ScicosLab is a good software for modeling, simulation, analysis and design, providing a large set of functions for system engineering and scientific applications. It offers almost all the functionalities provided by Matlab/Simulink. ScicosLab has a large number of toolboxes that include graphics functions, numerical integration, linear algebra, optimization, among others. Developed and maintained by the *Institut National de Recherche en Informatique et en Automatique*, INRIA, through the Project METALAU (*Method, algorithmes et logiciels pour l'automatique*) and *Ecole Nationale des Ponts et des Chaussées*, ENPC, it can be freely downloaded from the internet (Coelho, 2010).

Some features of ScicosLab are: high-level programming language, dedicated editor, hundreds of mathematical functions, ability to add programs from other languages (C, Fortran), various toolboxes (linear algebra, polynomial, statistics, classical control, identification, among others). The syntax is similar to that of Matlab and the package includes a simulator called Scicos as an alternative to Simulink.

ScicosLab is being developed since 1990, is compatible with Microsoft Windows and Linux operational systems and this work is aimed at the Windows version. The Windows version of ScicosLab lacks drivers interfacing Data Acquisition (DAQ) devices because most of the effort in this area is headed towards the Linux version with Real-Time Application Interface, or RTAI (Meza et al., 2009). Proper modifications on Linux with RTAI grant hard real-time DAQ. This has not been accomplished for the Windows version yet and only a few number of DAQ devices are supported in soft real-time mode. A parallel contribution of this paper is to inform the development of a new driver for the USB-1208LS DAQ (Measurement and Computing) to work with Windows and ScicosLab, which can be found at www.das.ufsc.br/~aarc/FOSS-ADCON/USB1208LS.rar.

4. CONTROL SYSTEM ANALYSIS WITH SCICOSLAB

The following control activities are asked to the students in order to learn the ScicosLab language and gain background knowledge in tasks of modeling, simulation, PID-IMC design and implementation of real-time control applications.

4.1 PID-IMC Design for a Level Plant

The first numerical simulation is a nonlinear coupled liquid-tank system as shown in Fig. 3 (Tan and Li, 2001). Level control systems are common in the industry (chemical, petrochemical, nuclear and cellulose). The equations that characterize a second-order model of the level plant are

$$A_1\dot{h}_1(t)=u(t)-a_1c_1\sqrt{2g\{h_1(t)-h_2(t)\}}$$

$$A_2\dot{h}_2(t)=a_1c_1\sqrt{2g\{h_1(t)-h_2(t)\}}-a_2c_2\sqrt{2g\{h_2(t)-h_0\}}+d(t)$$

where $h_1(t)$ is an intermediate variable representing the liquid level in tank 1, $h_0 = 3\ cm$ is the liquid level of the reservoir, $A_1 = A_2 = 100\ cm^2$ are the cross-section area of both tanks, $a_1 = a_2 = 0.396\ cm^2$ are the orifice areas, $c_1 = 0.53$ and $c_2 = 0.63$ are the discharge constants (of tank 1 and tank 2, respectively) and $g = 981 cm/s^2$ is the gravitational constant.

The control objectives and operational constraints are: i) the input $u(t)$ is used to adjust the liquid level in tank 2, $h_2(t)$, at a desired level, ranging from *zero* to $33.33\ cm^3/s$, ii) the input $d(t)$ in tank 2 is used as a load disturbance of magnitude $8.33\ cm^3/s$, added at $t = 1200\ s$, iii) the level in tank 2 must be regulated in two values from the nominal operation point, iv) an overdamped behavior for $h_2(t)$ with low control variance, disturbance rejection and zero steady-state error are the closed-loop specifications to be achieved by the PID controller tuned with the IMC technique.

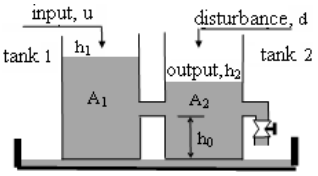

Fig. 3. A nonlinear coupled level process.

The digital PID-IMC controller is implemented according to Table 1 and simulation results are shown in Fig. 4.

Table 1. Parameterization of the level control system.

Discrete Linear Model	$\frac{0.0053z^{-1}+0.0023z^{-2}}{1-1.437z^{-1}+0.461z^{-2}}$
Closed-Loop Dynamic	$\lambda = 25$ s
Sampling Time	2 s

Numerical experiments illustrate that the responses meet the performance specifications and the ultimate gains of the PID controller are given by $K_c = 5.21$, $T_i = 42.92$, $T_d = 1.79$. The

PID-IMC design is the closed-loop pole being associated with the behavior of a first-order system and its adaptability to various processes. The students can perform experiments with different first-order tuning parameters.

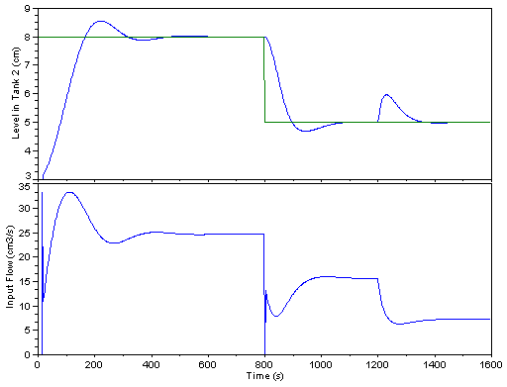

Fig. 4. Dynamics of the level process with PID-IMC.

The corresponding code in ScicosLab of the PID controller with the IMC tuning for the level plant is shown in Table 2.

Table 2. ScicosLab code for the level plant.

```
clear; xdel(0:1); clc;
// ----- Constants of the level plant
ts = 2; d = 8; niter = 800;
A1 = 100; a1 = 0.396; c1 = 0.53; A2 = 100; a2 = 0.396; c2 = 0.63;
// ----- Initial conditions
h1(1:d) = 3; h2(1:d) = 3; h0 = 3; g = 981; umin = 0; umax = 33.33;
pert(1:600) = 0; pert(601:niter) = 8.33; u(1:d) = 12; erro(1:d) = 0;
// ----- Design parameters
lambda = 25; alfa = exp(-ts/lambda);
a1e = -1.437; a2e = 0.461; b0e = 0.0053; b1e = 0.0023;delay = 0;
// ----- Reference signal
yr(1:400) = 8; yr(401:niter) = 5;
// ----- Simulation
for k = d:niter
// ------- Output
  h1(k) = h1(k-1)+(ts/A1)*(u(k-1)-a1*c1*sqrt(2*g*(h1(k-1)-h2(k-1))));
  h2(k) = h2(k-1)+(ts/A2)*(pert(k-1)+a1*c1*sqrt(2*g*(h1(k-1)-h2(k-1)))-...
          a2*c2*sqrt(2*g*(h2(k-1)-h0)));
  erro(k) = yr(k) - h2(k);
  kaux = (1 - alfa)/((b0e + b1e)*(1 + delay*(1 - alfa)));
  kc = -kaux*(a1e + 2*a2e);
  ti = -((a1e + 2*a2e)*ts)/(1 + a1e + a2e);
  td = -(a2e*ts)/(a1e + 2*a2e);
  u(k) = u(k-1)+(kc*(1+ts/ti+td/ts))*erro(k)-(kc*(1+2*td/ts))*erro(k-1)+...
         (kc*(td/ts))*erro(k-2);
  if u(k) <= umin; u(k) = umin; elseif u(k) >= umax; u(k) = umax; end
end
// ----- Results
t = 0:ts:niter*ts-ts;t = t';
subplot(2,1,1),plot(t,h2,t,yr),ylabel('Level-Tank 2 (cm)'),xlabel('Time (s)');
subplot(2,1,2),plot(t,u),ylabel('Input Flow (cm3/s)'),xlabel('Time (s)');
```

4.2 PID-IMC Design for a DC Motor Plant

The second numerical simulation considers the DC motor model for velocity control activities, as shown in Fig. 5 (this plant shows a dynamic that is significantly faster than the coupled tank system).

The DC motor is an electromechanical transducer that converts DC voltage applied at its terminals in mechanical movement of its axis. Typical applications of servomechanisms are: machine tools, industrial robots, positioning systems and conveyors. To control the velocity or position it is necessary to control the voltage applied to the armature terminal. This simplicity makes the DC motor an important component for control systems applications in industrial and domestic environments. The variables of interest are: R is the armature resistance, L is the armature inductance, I is the armature current, $v(t)$ is the armature voltage (input), $e_a(t)$ is the back emf, $w(t)$ is the angular velocity (output), T is the torque developed by the motor, J is the equivalent moment of inertia of the motor and load referred to the motor shaft, and B is the equivalent viscous-friction coefficient of the motor and load referred to the motor shaft.

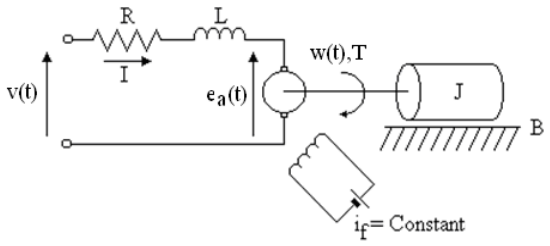

Fig. 5. DC motor process.

The differential equation fitting the open-loop behavior of the DC motor process is given by (Tan and Li, 2001)

$$\frac{d^2w(t)}{dt^2}+\left(\frac{JR+BL}{JL}\right)\frac{dw(t)}{dt}+\left(\frac{BR}{JL}\right)w(t)=\left(\frac{K_T}{JL}\right)v(t)$$

where $v(t) \in [-5\ V\ ,\ 5\ V]$, $K_T = 13.5\ Nm/A$, $R = 9.2\ \Omega$, $L = 0.25\ H$ and $J = 0.001\ kgm^2$. To assess performance aspects, robustness and control activity, a parametric variation in the friction coefficient is simulated. Initially with $B = 2.34x10^{-3}\ Nms$ then switches to $B = 1.34x10^{-3}\ Nms$ and returns to $B = 2.34x10^{-3}\ Nms$ at $t = 3\ s$ and $t = 8\ s$, respectively.

The objective of the control loop is to obtain a controller which provides a closed-loop step response with a minimum rise time and zero steady-state error. The reference signal is given by $y_r(t) = 9\mathbf{1(t)} - 4.5\mathbf{1(t-5)}\ rps$, where $\mathbf{1(t)}$ is the unit step signal, and the experiment takes a total of 100 samples for a sampling time of $10\ ms$.

The digital PID-IMC is implemented according to Table 3 and simulation results are shown in Fig. 6.

The results of the numerical experiments illustrate that the dynamic responses are appropriate from the viewpoints of settling time and control energy, and meeting the performance specifications for reference and parametric changes. PID-IMC gains are given by $K_c = 0.0046$, $T_i = 0.4529$ and $T_d = 0.0213$.

4.3 PID-IMC Design for a Damped Pendulum Plant

In order to show the ScicosLab platform in a real-time control application it is employed a damped pendulum, Fig. 7, for modeling and digital control essays. This practical process belongs to the positional control set of plants developed at DAS/Federal University of Santa Catarina.

Practical systems are of vital importance to control engineering students since control systems are real systems.

Table 3. Parameterization of the DC motor system.

Discrete Linear Model	$\frac{2.378z^{-1}+2.087z^{-2}}{1-1.669z^{-1}+0.676z^{-2}}$
Closed-Loop Dynamic	$\lambda = 0.15$ s
Sampling Time	0.01 s

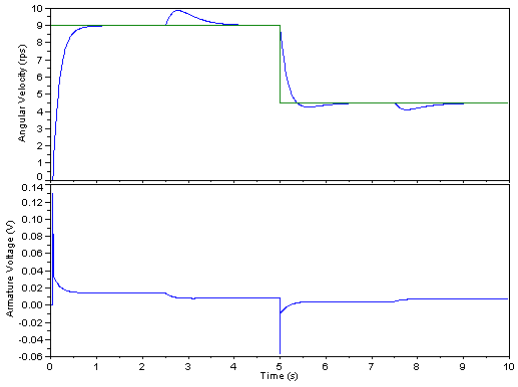

Fig. 6. Dynamics of the DC motor plant with PID-IMC.

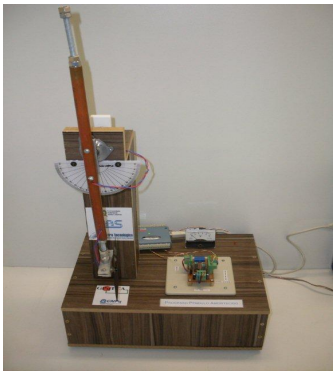

Fig. 7. The damped pendulum apparatus.

4.3.1 Damped Pendulum Identification

For oscillatory plants the following continuous model is utilized where $y(t)$ is the output, $u(t)$ is the input, K_p is the static gain, ζ is the damping factor and w_n is the natural frequency.

$$\frac{d^2y(t)}{dt^2}+2\zeta w_n\frac{dy(t)}{dt}+w_n^2y(t)=K_pw_n^2u(t) \qquad (22)$$

The first part of the estimation task consists of the realization of an open-loop essay, which means to obtain the step response to measure, from Fig. 8, the period *To* of the oscillation and the first two peaks *a1* and *a2*.

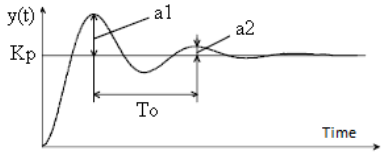

Fig. 8. Estimation of a second-order oscillatory model.

Once these values are available, the estimated model parameters are determined by

$$\zeta=1\Big/\sqrt{1+\left\{2\pi\Big/ln\left(a2\Big/a1\right)\right\}^2}\ ;\ \omega_n=2\pi\Big/To\sqrt{1-\zeta^2}$$

To implement the linear modeling task a step input of *3 volts* is applied to the plant and the measurements of the angular position are saved at a file called *pam.dat*. The *fscanMat* function of the ScicosLab software is used to read an ASCII text matrix from a file as in the code shown in Table 4.

Table 4. ScicosLab code for real-time data acquisition.

```
// Data Acquisition Task
clear; xdel; clc;
// Initialization
exec('loader.sce'); // Load USB-1208LS driver
nit = 100; ts = 0.1; u(1:nit) = 3; t = (0:ts:nit*ts-ts)';
realtimeinit(ts); // Sampling time
for k = 1:nit, // Open-loop
  y(k) = receive_data();
  send_data(u(k));
  realtime(k); // Sampling time hold
end
send_data(0); dados = [t y u];
savematfile('pam.dat','dados','-ascii'); // Save data
// End of real-time simulation
```

During the data acquisition activity the students learn real-time programming with ScicosLab. Once measures are available for the open-loop essay of the damped pendulum, then the second-order classical estimator of Fig. 8 is implemented based on Table 5.

On the second part of the identification, the students are asked to implement a Scicos block diagram (Fig. 9) to validate the second-order estimated model (*num(s) = 8.51*, *den(s) = s^2 + 0.87s + 9.68*). In this way, students are able to verify that the dynamic response to a step of the estimated model is similar to the measurements of the damped pendulum applied to the second-order estimator. The students also learn that an estimated model never exactly matches reality (Fig. 10).

Table 5. ScicosLab code for modeling using real data.

```
clear; xdel; clc;
dados = fscanfMat('pam.dat'); // Load measurements
t = dados(:,1); // Time
y = dados(:,2); // Output
u = dados(:,3); // Input
n = length(y); // Number of samples
media_y = mean(y(n-0.1*n:n)); media_y = media_y*ones(n,1);
// Modelling
a1_y = max(y); a1 = vectorfind(y,a1_y,'r');
minimo = min(y(a1+1:n));
ind_min_tmp = vectorfind(y(a1+1:n),minimo,'r');
ind_minimo = a1 + ind_min_tmp;
a2_y = max(y(ind_minimo+1:n));
a2_tmp = vectorfind(y(ind_minimo:n),a2_y,'r');
a2 = ind_minimo + a2_tmp;
ts = 0.1; // Sampling time
To = (a2 - a1)*ts;
qsi = 1 / (sqrt(1+((2*%pi)/(log(a2/a1)))^2));
wn = 2*%pi / To*sqrt(1-(qsi)^2);
Kp = media_y(n) / 2.5; // Step u = 2.5;
// Estimated model
s = poly(0,'s'); G = Kp*(wn^2) / (s^2 + 2*qsi*wn*s + wn^2);
```

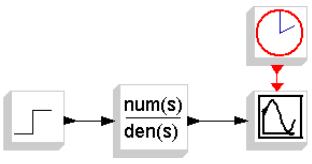

Fig. 9. Scicos for the open-loop damped pendulum dynamic.

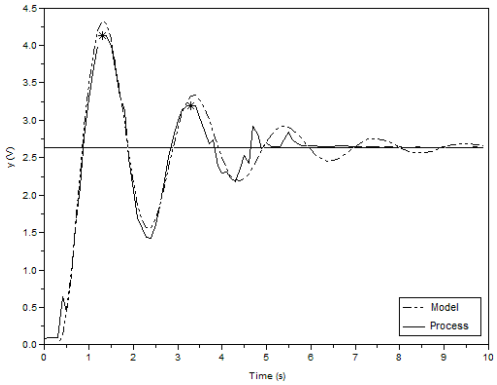

Fig. 10. Open-loop step response of the real process and of the second-order linear model.

4.3.2 Damped Pendulum with PID-IMC Control

With the estimated parameters for the second-order linear model of the damped pendulum, it is possible to design a linear controller to stabilize the plant in different operational points. The tuning parameter τ_{MF} regulates the settling time and the control performance of the closed-loop system. The synthesis of the digital controller is based on PID-IMC tuning of Morari and Zafirou (1989). Table 6 shows the tuning formula for the PID controller with a process with second-order model and IMC design.

Table 6. IMC tuning for the PID.

Plant Model	PID Tuning		
$\frac{K_p}{\tau^2 s^2+2\zeta\tau s+1}$	$K_c=\frac{2\zeta\tau}{K_p(\tau_{MF})}$	$T_i=2\zeta\tau$	$T_d=\frac{\tau}{2\zeta}$

The PID for the real-time control system structure for the damped pendulum is shown in Fig. 11. By using the estimated model obtained from Table 5, the tuning parameters of the PID-IMC are $T_s = 0.1$ *s*, $\tau_{MF} = 1.5$ *s* and, for (14), $num(z) = 0.93z^2 - 1.63z + 0.78$ and $den(z) = z^2 - z$. On Fig. 12 the demonstrative results for setpoint changes of the experiment are shown.

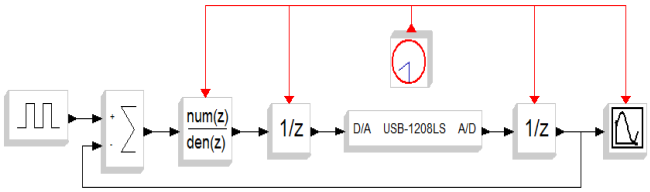

Fig. 11. Scicos scheme for real-time PID control.

These examples described in this section show the possibility of using ScicosLab/Scicos for control educational purposes.

5. CONCLUSIONS

Practical and numerical control systems are of great importance in many industrial environments and offer an opportunity to teach important concepts of process control engineering education. ScicosLab grants no cost with high quality and open source software that can be used on courses of control theory as an alternative to Matlab. This paper has shown the modeling, design, analysis and digital PID-IMC

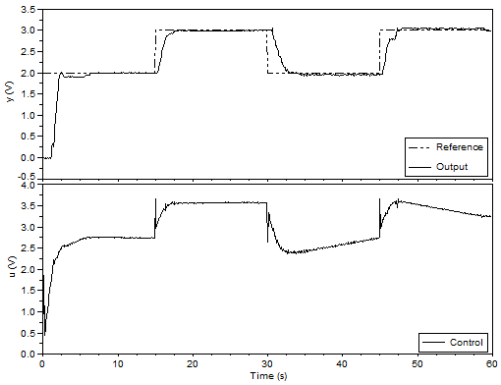

Fig. 12. Damped pendulum response with PID controller.

implementation in three applications emphasizing aspects of education, tuning efficiency and robustness in the presence of dynamic uncertainties, disturbances and setpoint changes, entirely done using FOSS. Programming procedures with ScicosLab to perform PID-IMC control simulation studies can help educators to exchange some lab experiences to illustrate control ideas in the academia.

ACKNOWLEDGEMENTS

We would like to thank CNPq under grant 478828/2009-8.

REFERENCES

Åström, K.J. and Hägglund, T. (2006). *Advanced PID Control*, ISA.

Bobál, V., Böhm, J., Fessl, J. and Machácek, J. (2005). *Digital Self-Tuning Controllers*, Springer.

Bucher, R. and Balemi, S. (2005). Scilab/Scicos and Linux RTAI - A Unified Approach, *IEEE Conf. on Control Applications*, Toronto, Canada, pp. 1121-1126.

Coelho, A. A. R. (2010). Teaching Dynamic System Identification with ScicosLab, *2st HeDiSC Workshop*, San Carlos, Costa Rica, Available at: <http://hedisc.ietec.org/>.

Grzegorz, S., Tomasz, Z. and Andrzej, B. (2008). Rapid Control Prototyping with Scilab/Scicos/RTAI for PC-based and ARM-based Platforms, *Int. Workshop on Real Time Software*, Wisla, Poland, pp. 739-744.

Li, D., Zeng, F., Jin, Q. and Pan, L. (2009). Applications of an IMC based PID Controller Tuning Strategy in Atmospheric and Vacuum Distillation Units, *Nonlinear Analysis: Real World App.*, vol. 10, pp. 2729-2739.

Meza, B.C., Romero, J.A.A., Bucher, R. and Balemi, S. (2009). Free Open Source Software in Control Engineering Education: A Case Study in the Analysis and Control Design of a Rotatory Inverted Pendulum, *14th IEEE Int. Conf. on Emerging Technologies and Automation Education*, Palma de Mallorca, Spain.

Morari, M. and Zafiriou, E. (1989). *Robust Process Control*, Prentice Hall.

ScicosLab Web Site, [Online] Available at: <www.scicoslab.org>.

Tan, C.K. and Li, Y. (2001). Performance-based Control System Design Automation via Evolutionary Computing, *Eng. App. of Artificial Intelligence*, vol. 14, pp. 473-486.