Dual RST-control of an Inverted Pendulum with Simulink S-functions Implementation

Eric Ostertag, Emmanuel Godoy and Joana Carvalho-Ostertag

Abstract—**This paper has a two-fold pedagogical goal. First it describes the use of RST controllers in a cascaded-loop structure. Second it gives the student the tools to design such a control structure for a real-time application and to implement it on a real experimental setup, all in one single lab session. The dual-loop cascaded control involving two discrete-time RST-controllers is applied to an inverted pendulum. The inner loop stabilizes the angle of the pendulum around its unstable (upwards) equilibrium, whereas the outer loop controls the cart position by giving to the inner loop the proper angle reference required to make the cart move on its rail. The burden of programming these two control loops for a real-time experiment is simplified for the students by the use of S-functions in a SIMULINK diagram. The writing of such S-functions for a real-time use alleviates indeed his programming task while obliging him to concentrate on the real-time control task itself. This control scheme is then applied directly to the pendulum by means of the Real Time Windows Target toolbox from MATLAB.**

I. Introduction

THE use of RST-controllers has been described mainly for SISO systems. Their extension to MIMO systems is given here in a cascaded structure of two control loops, applied to a one-input, two-output system, the inverted pendulum.

In many published works, the inverted pendulum is controlled by means of state feedback, eventually with the use of an observer to reconstruct the missing state variables (see [1]-[5]). Only a few publications deal with different control structures to stabilize an inverted pendulum, such as cascade control configurations. In [6] such a control structure is used with two fuzzy controllers, in a configuration similar to the one which will be used in the present paper: an inner control loop stabilizes the angle $\theta(k)$ around the unstable equilibrium (upwards) and an outer loop controls the cart position $x(k)$ on the rail. In an interactive tutorial accessible on line [7], the authors propose a cascade control of an inverted pendulum made of two PID controllers. In a similar on-line tutorial [8], a cascade control with two phase-lead controllers is developed for the same system. A dual-loop control, with two PD controllers in the feedback paths, then a single two-dimensional RST controller, are applied successively in [9] to a one-dimensional gantry crane, which is equivalent to a cart-pendulum system operating around its (downwards) stable equilibrium point. In this book, a single-loop fuzzy control of an inverted pendulum is also described. Finally, a somewhat different cascade control structure is introduced in [10] for an inverted pendulum: here the inner loop controls the cart velocity, by means of a high-gain proportional controller, whereas the outer loop controls the pendulum angle by means of a non-linear swing-up-and-hold controller based on a virtual actuator approach [11]. In none of these works but one, RST controllers have been used.

The purpose of the present paper is to design a cascade control structure to stabilize an inverted pendulum, using two RST controllers. This choice results from the possibility for a student to calculate these controllers, due to their design simplicity (see [12]-[15]), and to implement them on a real experimental setup, all within one laboratory session. In order to avoid the pitfall of the time-consuming C-programming, though letting the student still being confronted with the special needs of real-time programming, the use of S-functions in a SIMULINK diagram is proposed here as an interesting alternative. Thanks to the Real Time Windows Target toolbox of MATLAB, the control scheme can then be applied directly to a real inverted pendulum.

II. Dual-Loop Control of the Inverted Pendulum

A. Cascaded RST-controllers structure

In order to apply a cascade control configuration, the discrete model of the pendulum, a multivariable system, must first be decomposed into two cascaded transfer functions, $G_1(z)$ and $G_2(z)$, as illustrated in Fig. 1. Two RST-controllers, R_1-S_1-T_1 and R_2-S_2-T_2, are then applied to the two loops.

In the case of the inverted pendulum, the outputs of the two previous transfer functions are respectively the z-transforms $\Theta(z)$ and $X(z)$ of the pendulum angle and the cart position. The parameters of our experimental setup are given in the form of a state space model (see Appendix).

Manuscript received September 29, 2006.

E. Ostertag is with the Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection (LSIIT, UMR CNRS-ULP 7005), ENSPS, BP 10413, F-67412 Illkirch Cedex, France, and is Emeritus Professor of the University Louis Pasteur, Strasbourg, France (Corresponding author: phone: +33 3 90 24 44 63, fax: +33 3 90 24 44 80; e-mail: Eric.Ostertag@ensps.u-strasbg.fr).

E. Godoy is Professor at the Ecole Supérieure d'Electricité (Supélec) in Gif-sur-Yvette, Plateau de Moulon, F-91192 Gif-sur-Yvette Cedex, France (e-mail: Emmanuel.Godoy@supelec.fr).

J. Carvalho-Ostertag is Assistant-Professor at the Ecole Nationale Supérieure de Physique, University Louis Pasteur, Strasbourg, France. (e-mail: jcarvalho@ensps.u-strasbg.fr).

ISBN 978-3-9524173-8-6

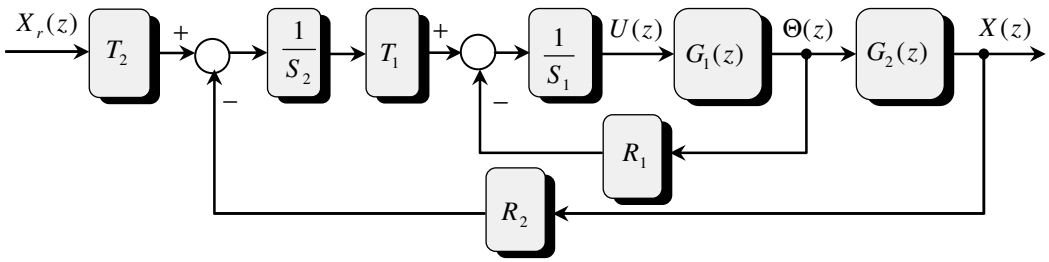

Fig. 1. Cascaded-loop structure with two RST controllers

Since this model gives only access to input-output transfer functions, such as $\Theta(z)/U(z)$ and $X(z)/U(z)$, the plant transfer function of the outer loop must be deduced from the two previous ones by division:

$$G_2(z)=\frac{X(z)}{\Theta(z)}=\frac{X(z)/U(z)}{\Theta(z)/U(z)}=G_1^{-1}(z)\frac{X(z)}{U(z)}. \qquad (1)$$

The first RST controller is then designed with respect to the plant model $G_1(z)=B_1(z^{-1})/A_1(z^{-1})$ and the second RST controller is designed to control a plant $G_{\text{ext}}(z)$, obtained by replacing the inner loop by its closed loop equivalent and multiplying it by $G_2(z)$,

$$G_{\text{ext}}(z)=\frac{B_1T_1}{A_1S_1+B_1R_1}G_2(z), \qquad (2)$$

as shown in Fig. 2. The numerical values of the state space model and the above transfer functions are given in the Appendix.

B. Controller design

For the inner loop a simplified RST controller, with $T_1=R_1$, is designed so as to impose to the closed loop, sampled at $T_s=0.03$ s, a double real pole at the discrete equivalent of $s=-10$ rad/s and a third pole at $s=-20$ rad/s, yielding the following discrete-time characteristic polynomial:

$$A_{m,1}(z^{-1})=1-2.030\,z^{-1}+1.362\,z^{-2}-0.3012\,z^{-3}. \qquad (3)$$

$G_1(z^{-1})$ contains a derivation term (one zero at $z=1$) but no integral action (see Appendix). Since the position regulation results from the application of an adequate angle reference to the inner loop, it is desirable to introduce one integrator into this loop, in order to reject constant load disturbances and guarantee a good tracking of constant or slowly varying references (Fig. 4). This will be done by means of the polynomial S_1, thus by letting $S_1(z^{-1})=(1-z^{-1})S_1'(z^{-1})$ in (4). Recall that, in the case of the simplified RST structure ($T_1=R_1$), the introduction of an integrator in $S_1(z^{-1})$ makes the steady-state error vanish, both for disturbance rejection and reference tracking [13]. During the selection process of terms to be compensated or not (see [14], [15]), it is thus mandatory to place the zero of $G_1(z^{-1})$ at $z=1$ among the terms to be compensated, $B_1^+(z^{-1})$, although it is lying on the unit circle. Failure to do so, thus letting $B_1^-(z^{-1})$ have $(1-z^{-1})$ in factor, would lead indeed to a Diophantine equation,

$$(1-z^{-1})A_1^-S_1'+B_1^-B_m'=A_{m,1}, \qquad (4)$$

in which the polynomial coefficients of the two unknown polynomials $S_1'(z^{-1})$ and $B_m'(z^{-1})$ would have one common factor which does not divide the second member, making thus the equation unsolvable. The second zero of $G_1(z)$, $z=-0.9797$, is not compensated since it lies on the negative real axis [12]. On the denominator side, the two stable poles, $z=0.9494$ and $z=0.865$, are chosen for compensation.

The resolution of (4) by the Sylvester matrix method [12] yields then the following controller polynomials:

$$R_1=5.6497-15.49\,z^{-1}+14.145\,z^{-2}-4.3025\,z^{-3}$$
$$S_1=1-2.096\,z^{-1}+1.192\,z^{-2}-0.096006\,z^{-3};\ T_1=R_1.$$

The outer loop is controlled by a full RST controller, designed so as to impose to the closed loop the discrete equivalent of the following poles: $s=-5;-5;-7$ rad/s, thus a characteristic polynomial

$$A_{m,2}(z^{-1})=1-2.532\,z^{-1}+2.1362\,z^{-2}-0.6005\,z^{-3}.$$

Here, the terms of $G_{\text{ext}}(z)$ chosen for compensation are the two zeros on the positive real axis, $z=0.8736$ and $z=0.9273$, and the three stable poles, $z=0.7408$ (double) and $z=0.5488$. No integrator is added to the outer loop, since a non-vanishing steady-state position error of the cart in response to constant measurement disturbances is considered tolerable here.

The resolution of the Diophantine equation corresponding to this case results then in the following controller polynomials:

$$R_2=-12.069+36.13\,z^{-1}-40.042\,z^{-2}+19.469\,z^{-3}-3.5016\,z^{-4}$$
$$S_2=1-1.9727\,z^{-1}+1.5076\,z^{-2}-0.83817\,z^{-3}+0.31442\,z^{-4}$$
$$T_2=-0.44279+0.89906\,z^{-1}-0.60306\,z^{-2}+0.13337\,z^{-3}.$$

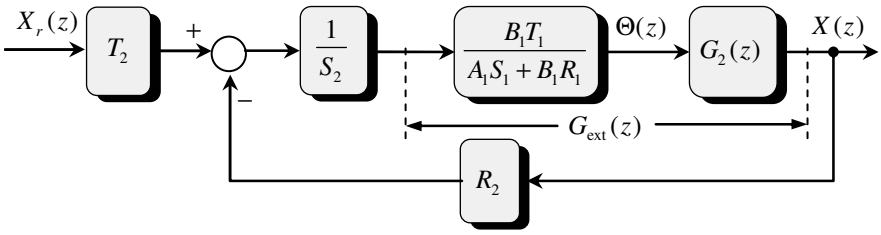

Fig. 2. Block diagram for the design of the outer-loop RST controller

The Nyquist diagram of the open-loop transfer function obtained by breaking the loop at the control signal level, as explained Section V, Fig. 10, is drawn in Fig. 3 with a distorted abscissa scale towards the right, in order to show the leading and trailing extremities of the three half-circles which close the diagram at infinity. It leads to the following remarks:

- there is one positive encirclement around the critical point $(-1,0)$ allowing to conclude to closed loop stability, the open loop having one unstable pole at $z=1.145$ (see Appendix) ;
- the upper and lower gain margins are respectively +2.8 dB and −2.4 dB ;
- the phase margin is somewhat low (12 degrees) but sufficient for experimental tests.

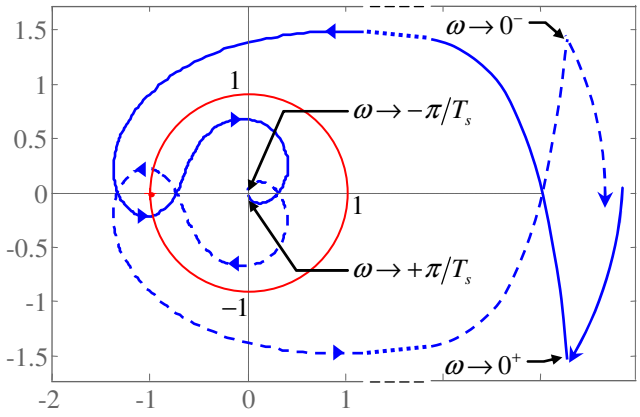

Fig. 3. Open-loop Nyquist diagram

III. Controller Implementation in Simulink by Means of S-Functions

To implement the RST controllers in the Simulink diagram, an easy way, which obliges the student to acquire good understanding of the particular aspects of the real-time operation, consists in making use of S-functions blocks. An RST controller is introduced in such a block as a discrete-time dynamic system, described by its difference equation:

$$s_0u_k = -s_1u_{k-1} - \ldots - s_{ns}u_{k-ns} + t_0y_{r,k} + \ldots + t_{nt}y_{r,k-nt} - r_0y_k - \ldots - r_{nr}y_{k-nr}$$

where ns, nt, nr are the respective orders of the polynomials $S(z^{-1})$, $T(z^{-1})$ and $R(z^{-1})$.

Simulink provides an automatic S-function builder, which makes the rest of the procedure very simple. The student introduces in his Simulink diagram an *S-function Builder* block, which will create automatically the S-function as a C Matlab-executable file (MEX-file) from his specifications and a few lines of C code.

In this builder, the RST controller is considered as a system with two inputs, $y_{r,k}$ and y_k, and one output, u_k. The past values of y_r, y, and u are considered as a state vector, which is shifted one position at each sampling time. To simplify the explanations, we will consider the case of *R*, *S*, and *T* polynomials of a maximum order in z^{-1} equal to 4, thus with $ns=nt=nr=4$, as will also be the case in the application described here.

The values to be entered in the dialog boxes of the *S-function Builder* are thus the following:

- In the *Initialization* pane, a number of discrete states of 12, as will be justified hereafter, initialized at 0, no continuous state, a discrete sample mode with a sample time value equal to the sampling time $T_s=0.03$ s of the plant;
- In the *Data Properties* pane, two 1-dimensional *Input ports*, y_c and y, one 1-dimensional *Output port*, u, and three *Parameters*, *r*, *s*, and *t* (capital letters should be avoided, capital S being a reserved variable name, though not documented by Matlab);
- In the *Outputs* pane, the code that computes the output $u(k)$ of the S-function at time step k ; in the case described here as an example this code is:

```
real_T u_out;
u_out = (-s[1]·xD[0] - s[2]·xD[1] - s[3]·xD[2] - s[4]·xD[3]
        + t[0]·yr[0] + t[1]·xD[4] + t[2]·xD[5] + t[3]·xD[6]
        + t[4]·xD[7]
        - r[0]·y[0] - r[1]·xD[8] - r[2]·xD[9] - r[3]·xD[10]
        - r[4]·xD[11])
        / s[0];
if (u_out > 10) { u[0] = 10; }
else if (u_out < -10) { u[0] = -10; }
else { u[0] = u_out; }
```

The last three lines of code impose a limitation of the output within the dynamic range of the actuator, −10 V to

+10 V . The algorithm illustrates the number of discrete states required. In this example, this number amounts to 12, namely 4 states *xD*(0) to *xD*(3) to store the past control values u_{k-1} to u_{k-4}, 4 states *xD*(4) to *xD*(7) to store the past reference values $y_{r,k-1}$ to $y_{r,k-4}$, and 4 states *xD*(8) to *xD*(11) for the past output values y_{k-1} to y_{k-4}.

- In the *Discrete Update* pane is entered the code that computes at the current time step the values of the discrete states at the next time step:

```
/* Shift past controls */
        int_T ls; int_T lt; int_T lr;
        for (ls = 3; ls > 0; ls--)  xD[ls] = xD[ls-1];
        xD[0] = u[0];
/* Shift past references */
        for (lt = 7; lt > 4; lt--)  xD[lt] = xD[lt-1];
        xD[4] = yr[0];
/* Shift past outputs */
        for (lr = 11; lr > 8; lr--)  xD[lr] = xD[lr-1];
        xD[8] = y[0];
```

The *Discrete Update* code shifts all the discrete states one sample step backwards, forgetting about the oldest ones, and replaces the most recent ones by, respectively, the control value newly computed in the *Output* pane, *u*(0), and the input values $y_r(0)$ and $y(0)$ of the reference and output which have been fed to the controller at sample step *k*. Note that since the syntax of S-function blocks requires the *Input ports* and the *Output ports* to be either vectors or matrices, scalars must be entered as the unique component of a vector of dimension one.

- In the *Build info* pane, only the three boxes *create a debuggable Mex-file*, *Show complete steps*, and *Generate wrapper TLC*, should be checked, the first two ones for obvious debugging purposes, and the third one in order to enable the generated S-function to run later in a real-time model generated by the Real-Time Workshop toolbox of Matlab.

Clicking the *Build* button on top of the S-function builder window completes then the process by creating the MEX-file corresponding to that S-function with the name entered in the field *S-function name* on the upper left of this window, here *RST_4*. This is also the name that will appear in the center of the S-function block.

The coefficients of the three polynomials *R*, *S*, and *T* can be introduced in one of two ways:

- Either the student enters them directly in the form of line-vectors in the respective fields *Value* of the *S-function parameters* pane of the S-function builder window before compiling the S-function; the block is then dedicated;
- Or these *Value* fields are filled simply with the respective variable names, *s*, *r*, and *t*; these variables must then be available in the Matlab environment before the S-function is effectively called.

A parameter masking possibility exists also, but this goes beyond the scope of this paper and will not be discussed here.

IV. Simulation Results

The simulation diagram used to test the cascaded control loop is represented in Fig. 4. The plant model is introduced by means of a discrete state space block, with an output matrix $\mathbf{C} = (\mathbf{C}_1^{\mathrm{T}}\ \mathbf{C}_2^{\mathrm{T}})^{\mathrm{T}}$ as given in the Appendix. A step generator applies a reference step of 1V at time $t = 1$ s to the position control loop, which expresses a displacement of the cart of about 6.6 cm on the rail, a step load disturbance of 1V is applied at time $t = 10$ s to the control input and a step measurement disturbance of -0.5 V at time $t = 20$ s to the angle output of the plant. The time responses are given in Figs. 5 and 6. In Fig. 5 is also plotted the output of the outer loop controller, i.e. the reference signal applied to the inner control loop (dashed line). The constant load and angle measurement disturbances are both completely rejected on the angle output, as expected since the angle controller contains an integrator. The angle measurement disturbance is not rejected however on the position output, as results from

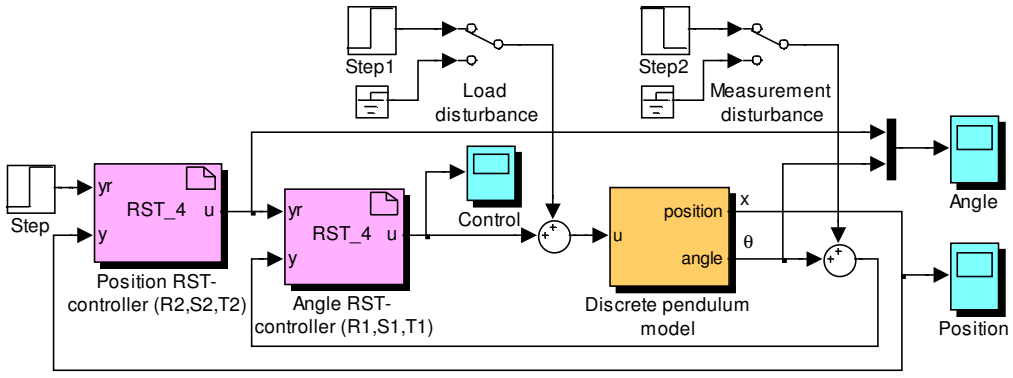

Fig. 4. Simulation diagram of the two-cascaded control loop of the inverted pendulum

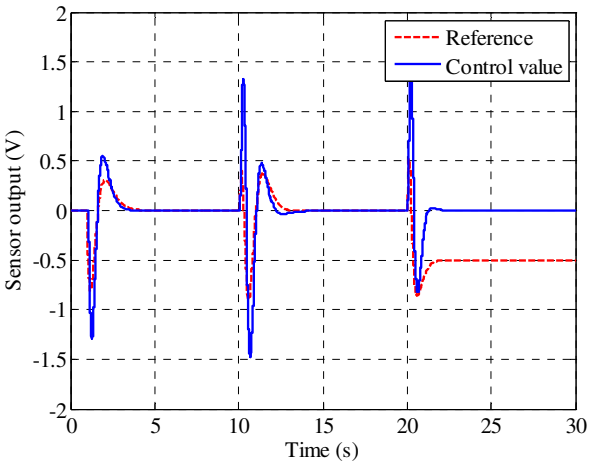

Fig. 5. Simulated time response of the pendulum angle

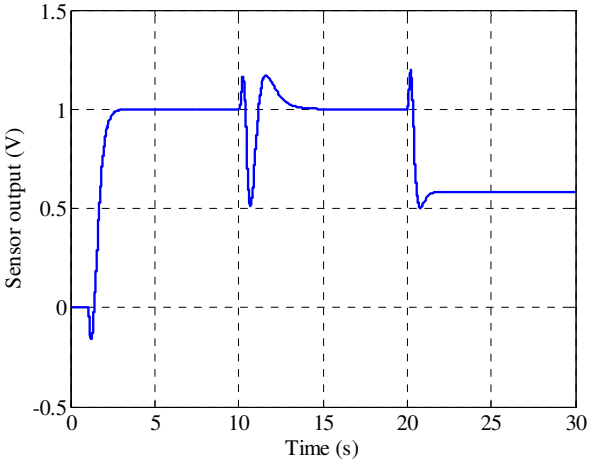

Fig. 6. Simulated time response of the cart position

the absence of an integrator in the outer loop between the comparator output and the point of application of the disturbance.

V. Real-Time Experiment

The two RST controller S-function blocks have then been applied to the real-time control diagram of Fig. 7, with only a slight modification. Here, a third input, *start*, has been connected to the RST_4 blocks. This logic signal is set to one by the *Trigger Logic*, after the pendulum has been maintained in its upper position during a few seconds. A few lines of C-code ensure then that $u\_out = 0$ before that instant, so as to provide a bumpless start of the regulation.

The interface with the inverted pendulum driver box occurs by means of three DLL modules, the DAC *dac6214da*, the ADC *dac6214ad*, and the Digital Output *dac6214do* of the figure. These three modules have been written by one of the authors on the basis of an existing template provided by Matlab in its documentation [16], so as to be used in conjunction with the Real Time Windows Target toolbox.

The real-time model of Fig. 7 is then built during the Real Time Workshop build process.

Figs. 8 and 9 illustrate the time responses of the pendulum angle and cart position, recorded in this real experiment. A step reference of 2 V, which corresponds to a cart displacement of about 13 cm, is applied at $t = 15$ s. A limit cycle with a period of about 2.8 s is clearly visible in these figures. This limit cycle is caused by the nonlinearity resulting from the dry friction force of the cart on its rail.

The gain of 2 placed on the angle feedback loop in Fig. 7 has been introduced experimentally in order to cope with this nonlinearity. As expected, the angle value tracks closely its reference value computed by the position controller (Fig. 8). A theoretical approximation of the oscillation frequency can be obtained by the method of the describing function analysis [17]. To obtain the transfer function of the open loop compensated plant we have applied the Matlab instruction *dlinmod* to the Simulink diagram of Fig. 10. The control loop is broken at the level of the control signal

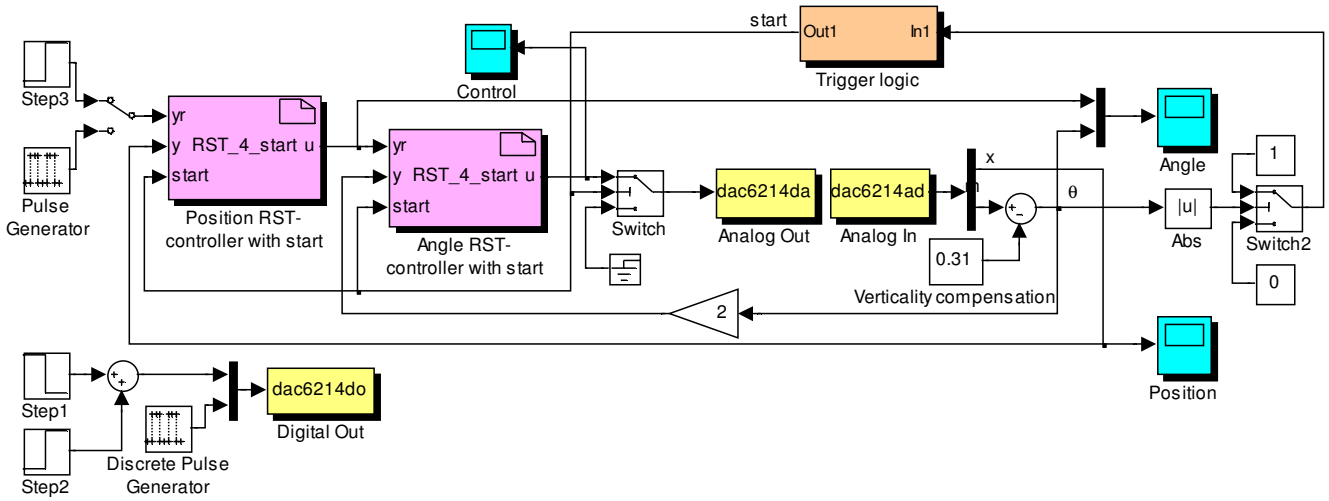

Fig. 7. Simulink diagram for the real-time control of the inverted pendulum

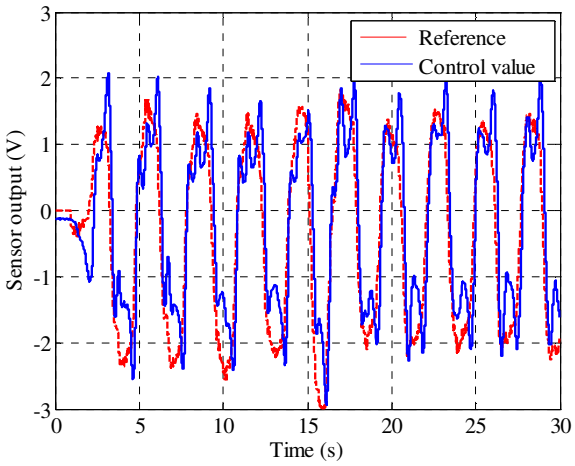

Fig. 8. Experimental time response of the pendulum angle

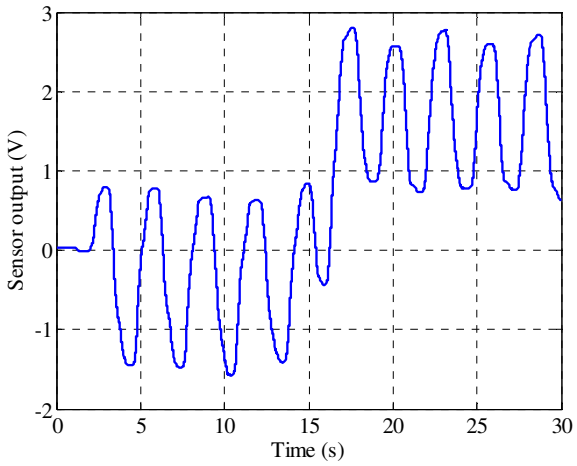

Fig. 9. Experimental time response of the cart position

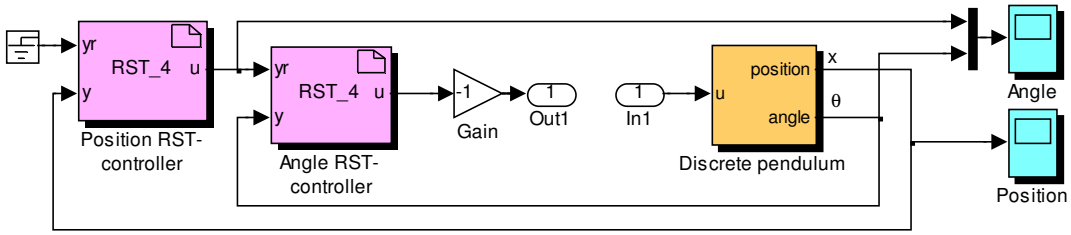

Fig. 10. Determination of the compensated open-loop transfer function

applied to the inverted pendulum model. The intersection of the Nyquist plot of the resulting linear transfer function with the critical locus of the nonlinear element, a dead zone, occurs at an angular frequency of 6 rad/s, i.e. 0.95 Hz (Fig. 11). The critical locus of such a nonlinearity is the portion of the real axis comprised between $-\infty$ and -1. The discrepancy between theoretical and experimental values is probably due to the limits of validity of the describing function analysis and to the fact that the rotation friction of the pendulum has not been taken into account.

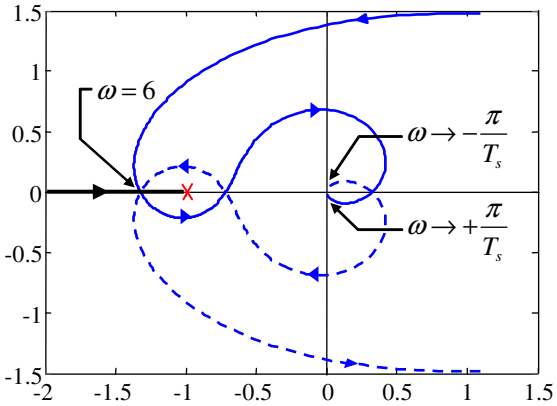

Fig. 11. Nyquist diagram of the open loop and critical locus of the nonlinear element.

VI. Conclusion

In this work, we have proposed a pedagogical way for the students to implement a digital controller of RST structure in a lab work involving a real experimental setup and a Simulink driven control scheme. An advantage of the implementation by means of S-functions is that it represents a compromise between the wish to expose the student to the practical aspects of a real-time algorithm, and the need to avoid long and cumbersome C-code writing, which often fails due to the limited time of a lab session.

Appendix

Our experimental setup is shown in Fig. 12. Its continuous-time state-space model is

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}_c\mathbf{x} + \mathbf{b}_c u \\ \mathbf{y} = \mathbf{C}\mathbf{x} \end{cases}$$

for a state vector $\mathbf{x} = (x \quad \theta \quad \dot{x} \quad \dot{\theta})^{\mathrm{T}}$, with the following parameter values as given in [3] :

$$\mathbf{A}_c = \begin{pmatrix} 0 & 0 & -1.9503 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.1289 & -1.9148 & 8.17\times10^{-4} \\ 0 & 21.496 & 26.339 & -0.1362 \end{pmatrix}; \mathbf{b}_c = \begin{pmatrix} 0 \\ 0 \\ -6.1347 \\ 84.386 \end{pmatrix}$$

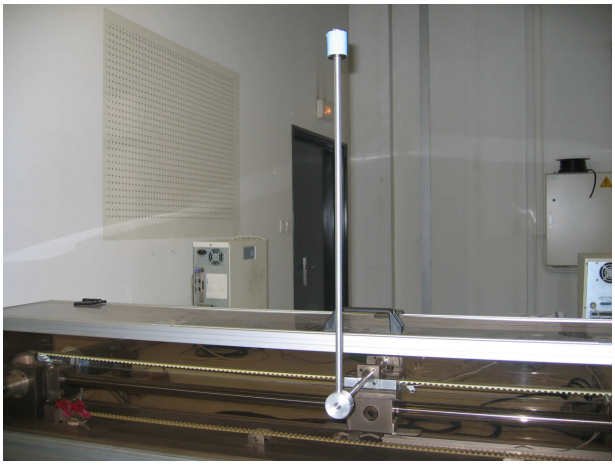

Fig. 12. The Amira LIP-100 inverted pendulum

These matrices correspond to the model in which all variables are expressed in their equivalent electrical units. The conversion factors between physical and electrical quantities are 14.9 V/m and -7.64 Vs/m for the cart position and velocity, respectively, -52.27 V/rad for the pendulum angle, and 2.6 N/V for the horizontal force applied to the cart by the driving mechanism.

The continuous-time transfer functions between the control input and the two measured outputs are derived from the state-space model by choosing successively, as the output matrix $\mathbf{C}$, $\mathbf{C}_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}$ and $\mathbf{C}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$:

$$G_1(s) = \frac{\Theta(s)}{U(s)} = \frac{84.39\,s}{(s+1.732)(s+4.833)(s-4.513)}$$

$$\frac{X(s)}{U(s)} = \frac{11.96\,(s+4.504)(s-4.379)}{s(s+1.732)(s+4.833)(s-4.513)}$$

The discrete model is obtained by sampling the continuous-time state-space model at $T_s = 0.03$ s and the two input-output discrete-time transfer functions are derived again from it by selecting successively the output matrices $\mathbf{C}_1$ and $\mathbf{C}_2$ as above. The following transfer functions result:

$$G_1(z) = \frac{\Theta(z)}{U(z)} = \frac{0.03727\,(z-1)(z+0.9797)}{(z-1.145)(z-0.9494)(z-0.865)}$$

$$\frac{X(z)}{U(z)} = \frac{0.005282\,(z-1.140)(z-0.8736)(z+0.9809)}{(z-1)(z-1.145)(z-0.9494)(z-0.865)}$$

$G_2(z)$ and $G_{\text{ext}}(z)$ of Fig. 2 are derived from these two transfer functions according to (1) and (2):

$$G_2(z) = \frac{X(z)}{\Theta(z)} = \frac{0.1417\,(z+0.9809)(z-0.8736)(z-1.140)}{(z-1)^2(z+0.9797)}$$

$$G_{\text{ext}}(z) = \frac{0.02984\,(z-0.9273)(z+0.9809)(z-0.8736)(z-1.140)}{(z-1)^2(z-0.7408)^2(z-0.5488)}$$

REFERENCES

[1] N. Becker, E. Ostertag. "Zur Berechnung der Zustandsrückführmatrix für Strecken mit mehreren Eingangsgrößen", *Automatisierungs-technik at*, vol. 5, pp. 214-215, 1987.

[2] K. Ogata, *Modern Control Engineering*, 2nd ed. Englewood Cliffs, N.J., Prentice Hall, 1990.

[3] "*Laboratory Setup Inverted Pendulum LIP-100*". Technical manual, Amira GmbH, Duisburg, 1992.

[4] K. Ogata, *Designing Linear Control Systems with MATLAB*, Englewood Cliffs, N.J., Prentice Hall, 1994.

[5] E. Ostertag, *Commande et estimation multivariables*, Ellipses, Coll. Technosup, Paris, 2006, ch. 1, pp. 9-60.

[6] E. Ostertag and M.J. Carvalho-Ostertag. "Fuzzy control of an inverted pendulum with fuzzy compensation of friction forces". *Int. J. System Sciences*, vol. 24, no. 10, pp. 1915-1921, 1993.

[7] B. Messner and D. Tilbury, "CTM: Control Tutorials for Matlab, Inverted Pendulum", Carnegie Mellon University and University of Michigan, 1996 [Online], Available: http://www.engin.umich.edu/group/ctm/index.html.

[8] G. Goodwin, S. Graebe and M. Salgado, "Interactive simulations", University of Newcastle, 1999 [Online], Available: http://csd.newcastle.edu.au/simulation.html

[9] M. Mokhtari and M. Marie, *Engineering Applications of MATLAB 5.3 and SIMULINK 3: Process Control, Fuzzy Logic, Neural Networks, Signal Processing*, Praxis Publishing, 2000, pp.303-340; pp. 381-440.

[10] P.J. Gawthrop and L. Wang, "Intermittent predictive control of an inverted pendulum", *Control Engineering Practice* 14 (11), 2006, pp. 1347-1356.

[11] P.J. Gawthrop and D.J. Ballance, "Virtual actuator control of mechanical systems", *Proc. 2005 International Conf. on Bond Graph Modelling and Simulation (ICBGM'05),* 2005 Simulation Series, New Orleans, pp. 233-238.

[12] K.-J. Åström, and B. Wittenmark, *Computer Controlled Systems: Theory and Design*, 3rd ed.,. Prentice-Hall, Upper Saddle River, N.J., 1997, pp. 170-175.

[13] E. Ostertag, "Steady-state error-free RST-controller design: a double Diophantine equation approach," *Proc. 5th European Control Conference*, *ECC'99*, CD-ROM F0920.pdf, Karlsruhe, Germany, 1999.

[14] E. Godoy, and E. Ostertag, *Commande numérique des systèmes : approches fréquentielle et polynomiale,* Ellipses (Coll. Technosup), Paris, 2003, ch. 6, pp. 167-194.

[15] E. Godoy, and E. Ostertag, "RST-controller design: a rational method based on two Diophantine equations", *7th IFAC Symp. On Advances in Control Education*, Madrid, Spain, 2006.

[16] *Simulink: Writing S-Functions*, Version 6, Chap. 3: "Writing S-Functions in C", The MathWorks, 2005, and template "*sfuntmpl_basic.c*" in the *Matlabroot\simulink\src* directory.

[17] J.J. Slotine, and W. Li, *Applied Non Linear Control*, Prentice-Hall, 1991, ch. 5, pp. 159-191.