Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A remote laboratory on PID autotuning

Alberto Leva [*], **Filippo Donida** [**]

[*] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy*
leva@elet.polimi.it
[**] *PhD student at the Dipartimento di Elettronica e Informazione*
donida@elet.polimi.it

Abstract: This manuscript presents a remote, web-enabled laboratory devoted to PID autotuning—a subject of significant importance from both the theoretical and the application-oriented point of view, but seldom available in remote laboratories. In detail, the manuscript presents two PI/PID autotuning experiments on physical control systems; the experiments are accessible by means of a browser, and the code is available as free software. Some pedagogical issues are also briefly discussed.

Keywords: Remote laboratory; control education; PID autotuning.

1. INTRODUCTION AND MOTIVATION

A huge number of automatic control experiments is nowadays offered by remote laboratories, as witnessed by works such as Aktan et al. (1996); Valera et al. (2005); Aliane et al. (2006), and many others; such resources are nowadays very important for control education at all levels (Dormido Bencomo, 2004). However, a very small minority of experiments available from remote laboratories is relative to the domain of autotuning, a notable exception being e.g. Balda et al. (2004).

This is a lack to be filled, for at least three reasons. First, there is a vast literature on autotuning, with particular reference to the PI/PID controller structure (Åström and Hägglund, 1995; Leva et al., 2002; O'Dwyer, 2003). When a subject is so extensively investigated, also the availability of well designed laboratories on that subject becomes a necessity, both for research and for education purposes. Second, autotuning capabilities are now included in many industrial regulators, see e.g. Li et al. (february 2006), and users tend more and more to rely on them. This fact is of particular importance in process control, for example, where a systematic application of autotuning to the hundreds of low- or mid- level loops involved in the typical plant allows to shorten commissioning and maintenance time significantly, and also to focus attention on the top-level plant controls. Also in motion control, to quote a somehow complementary field with respect to process control, autotuning yields significant benefits, as witnessed by the numerous products that nowadays offer such a feature. In synthesis, then, mastering not only the practical the use, but also (which is even more important) the theory of operation of an autotuner is important for the control engineer—and is not as easy a task as one may think at a first glance, as will appear later on. Third, *constructing* an autotuner is a challenging and fascinating activity, and probably the core of the competencies to induce when educating engineers. Turning a method to (auto)tune a regulator from the form it takes in the scientific literature to that of a complete procedure capable of running autonomously on a control computer involves a large amount of design choices, which can be taken consciously only if the designer possesses an adequate mix of control-theoretical and software engineering capabilities.

From this standpoint, experimenting with the *code* of autotuners is highly beneficial. It is worth stressing, in this particular respect, that a deep interaction with autotuning code is beneficial also for those professionals who most likely will use, but never *design*, an autotuner. In fact, seeing how such a thing works (at least once in one's career) helps understanding what can be expected from an autotuner and what cannot, provides the capability of discriminating myths from reality, in other words induces a correct balance between trust and skepticism, which is indeed the best attitude to have toward such a powerful and potentially dangerous technology as autotuning.

To summarise, then, teaching autotuning is beneficial for control and plant engineers' education, is recognised to be useful since a long time ago (Åström and Östberg, october 2006; Hang and Lee, 1990; Yurkovich and Passino, 1996), and for that purpose 'autotuning laboratories' (also, not to say in particular, remote) are surely of help.

Including autotuning in remote laboratories is not an easy task, however, particularly if one wants to put the students in contact with industry standards and 'real-life' development tools, which is not only advisable (also in accordance with the reasoning above), but also very consistent with recent education trends; in fact, quoting from (Balda et al., 2004, p. 36), 'the shift from very special dedicated application to the present state open industrial standard approaches and global program packages is observable in the field of remote and virtual laboratories development'.

This manuscript humbly attempts to propose a possible solution, by describing some web-enabled experiments located at the Cremona Automation Laboratory (CrAutoLab) of the Politecnico di Milano (the CrAutoLab URL is `www.cremona.polimi.it/crautolab`) and relative to the autotuning of PI/PID regulators.

From a general point of view, and also within the framework of the session "Virtual-remote Labs in control education: real experiences", the main contribution of this manuscript is that an existing remote, web-enabled laboratory is completed with autotuning experiments that

- involve physical control systems,

- have the architectural structure of 'real' industrial autotuners, with respect to both the software structure and the operator interface,
- and are built with a development system that is actually used in industrial applications.

This allows to provide the laboratory users with a realistic feel of the distance between specifying an autotuning method and developing a functional autotuner, in the attempt to cast (to the maximum reasonable extent) the 'practical' problems of the second activity in a 'theoretical' framework like that where the first one lives. Several autotuning methods are implemented, to allow comparisons between them also from the standpoint of the problems they pose when implemented (formalising those problems as much as possible, in accordance with the overall goal of the research). All the involved code, with particular reference to the autotuning procedures, is released as free software, to make sure that the user knowledge of the matter is complete.

2. THE IMPLEMENTED AUTOTUNING REGULATORS

2.1 Overview

At present, two autotuning regulators are implemented in the remote laboratory, namely

- an autotuning PID based on the IMC-PID tuning formulæ Rivera et al. (1986); Morari and Zafiriou (1989); Leva and Colombo (2004) and employing a First Order Plus Dead Time (FOPDT) process model, parametrised by means of a multiple step experiment,
- and an autotuning PI using the IMC approach and the FOPDT model structure too, but with the model parametrised by means of a relay experiment as suggested in Leva (2005).

The architecture of autotuning experiments, like that of all the CrAutoLab remote control experiments, is summarised in figure 1. Those experiments are based on the National Instruments LabVIEW web server technology, that allows users to remotely control a LabVIEW program ('Virtual Instrument', or 'VI' for short, in the LabVIEW terminology) running on a host machine, by means of a browser with a convenient plugin.

The LabVIEW web server, however, is conceived much more for 'teleoperation' at large than for systems subject to real-time constraints. Exporting a VI to the web by means of the LabVIEW web server, in fact, affects the execution time of that VI in a potentially severe manner, and generally hampers fast and accurate timing. The LabVIEW web server can therefore be used in remote laboratories to solve the low-level communication problems, while the more control-oriented software engineering (i.e., basically, interaction timing and operator interface) are left to the designer.

At CrAutoLab, the LabVIEW web server technology is employed as summarised in the following, and described in (Leva, 2006) with more detail and specific reference to fast-sampling experiments.

Each experiments is implemented by means of three VIs. A 'Control system' (CS) VI runs continuously on the host machine, implements the control system, is connected to the involved physical apparatus via the I/O interfaces of the host machine, and is not exported via the LabVIEW web server.

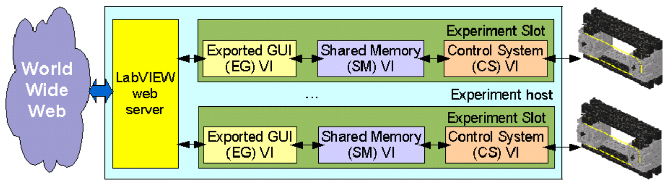


Fig. 1. Scheme of the architecture of LabVIEW-based remote control experiments at CrAutolab.

An 'Exported GUI' (EG) VI runs continuously on the host machine, is made available via the LabVIEW web server for remote user control, and provides the GUI (Graphical User Interface) for the experiment. Finally, a 'Shared memory' (SM) VI, composed of LabVIEW global variables, provides the memory space for asynchronous communication between the CS VI, which is subject to strict real-time specifications, and the EG VI, which is not. Such an organisation may appear complex but proved to be necessary, since - as anticipated - the LabVIEW web server interacts with the timing properties of the exported VIs up to a potentially significant extent. Should the experiments be implemented with a single VI (managing both the control system and the user interface), achieving timesteps of a few milliseconds would not be possible.

In the following, the two implemented autotuners are described in detail, compatibly with space limitations.

2.2 A step-based autotuning PID

This method considers the ISA PID regulator (ISA stands for the Instrumentation, Systems, and Automation society), i.e.,

$$R_{PID}(s) = K\left(1+\frac{1}{sT_i}+\frac{sT_d}{1+sT_d/N}\right), \tag{1}$$

and the FOPDT model

$$M_{FOPDT}(s) = \mu_M \frac{e^{-sL_M}}{1+sT_M}. \tag{2}$$

where μ_M is the gain, T_M the time constant (assumed here positive, which limits the scope to asymptotically stable processes as in the majority of autotuning applications) The IMC-PID tuning formulæ determine $R_{PID}(s)$ as

$$R_{PID}(s) = \frac{Q(s)F(s)}{1-Q(s)F(s)M_r(s)}, \tag{3}$$

where $Q(s)$ is the inverse of the minimum-phase part of $M(s)$, the rational model $M_r(s)$ is obtained with a (1,1) Padé approximation of the delay, i.e., substituting e^{-sL} with $(1-sL/2)(1+sL/2)$, and $F(s)$ (the so-called 'IMC filter') is first-order, i.e., $Q(s) = (1+sT_M)/\mu_M$, $F(s) = 1/(1+s\lambda)$, $M_r(s) = \mu_M(1-sL_M)/(1+sT_M)$. This leads, see e.g. Leva and Colombo (2004) to the tuning formulæ

$$\begin{aligned} T_i &= T_M + \frac{L_M^2}{2(L_M+\lambda)}, & K &= \frac{T_i}{\mu_M(L_M+\lambda)}, \\ N &= \frac{T_M(L_M+\lambda)}{\lambda T_i} - 1, & T_d &= \frac{\lambda L_M N}{2(L_M+\lambda)}. \end{aligned} \tag{4}$$

The step-based PID tuning procedure implemented in the remote laboratory obtains model (2) from an open-loop double-step experiment, and is structured as follows.

Step 1: wait for steady state. This is done by observing the derivative of the controlled variable, computed numerically and suitably filtered, after the loop has been set to manual by the procedure. To prevent erratic behaviours, the procedure anyway waits for a prescribed number of timesteps.

Step 2: measure noise band. The controlled variable is monitored for a length decided at the previous step, to figure out the amplitude of the noise present on it.

Step 3: step experiment, phase 1. Apply a control variable step of prescribed entity, and wait until the process variable has moved away from its average value, computed during step 2, by a suitable multiple of the noise band amplitude. The proportionality coefficient between the two is a parameter, and in the procedure there are many other arbitrarities relative to how the controlled variable is filtered and averaged to avoid spurious exits from this phase—looking at the code here is very instructive.

Step 4: step experiment, phase 2. Apply a second control variable step, opposite to the first one and of slightly larger amplitude (to ensure that the controlled variable be driven back to the initial value); wait until the conrolled variable reenters the noise band around its initial average value.

Step 5: step experiment, phase 3. Wait for a duration equal to step 4, then use the recorded response to reconstruct the process unit step response (computations are trivial).

Step 6: compute model parameters. From that unit step response, determine a model in the form (2): this can be done in many ways, ranging from thresholding (as in the VIs on the site) to the method of areas, and more; again, the possibility of interacting with the code is useful.

Step 7: compute PID parameters. This is done by means of (4).

The double-step experiment is summarised in figure 2, for better clarity and also to evidence how many arbitrary choices it involves.

2.3 A relay-based autotuning PI

This method (described e.g. in Leva (2005)) considers the ISA PI regulator

$$R_{PI}(s) = K\left(1+\frac{1}{sT_i}\right), \tag{5}$$

and, again, the FOPDT model (2). The IMC tuning formulæ also in the PI case, determine $R_{PI}(s)$ from (3) as for the PID, but substituting e^{-sL} with the (1,0) Pad approximation $1-sL$. The result, see again e.g. Leva and Colombo (2004), is

$$K = \frac{T_M}{\mu_M(L_M+\lambda)}, \quad T_i = T_M. \tag{6}$$

The formulæ (6) lead to the nominal open-loop transfer function

$$L_{n,PI}(s) = R_{PI}(s)M(s) = \frac{e^{-sL_M}}{s(L_M+\lambda)}. \tag{7}$$

and therefore to the nominal cutoff frequency ω_{cn} and the nominal phase margin φ_{mn}

$$\omega_{cn} = \frac{1}{L_M+\lambda}, \quad \varphi_{mn} = \frac{\pi}{2} - \frac{L_M}{L_M+\lambda}, \tag{8}$$

If a point $P(j\omega_o) = Ae^{j\varphi}$ of the process Nyquist curve is available, if (6) are used with model (2) parametrised so that $M(j\omega_o) = Ae^{j\varphi}$, and if λ is selected so that $\omega_{cn} = \omega_o$ and $\varphi_{mn} = \varphi_m$, the system

$$\begin{aligned} \frac{1}{L_M+\lambda} &= \omega_o, \\ \frac{\pi}{2} - \frac{L_M}{L_M+\lambda} &= \varphi_m \\ \frac{\mu_M}{\sqrt{1+(\omega_o T_M)^2}} &= A, \\ -\arctan(\omega_o T_M) - \omega_o L_M &= \varphi. \end{aligned} \tag{9}$$

is obtained. Solving (9) for (μ_M, L_M, T_M, φ_m) leads to

$$\begin{aligned} L_M &= \frac{1}{\omega_o} - \lambda, \quad T_M = -\frac{\tan(\varphi + \omega_o L_M)}{\omega_o}, \\ \mu_M &= A\sqrt{1+(\omega_o T_M)^2}, \quad \varphi_m = \frac{\pi}{2} - \frac{L_M}{L_M+\lambda} \end{aligned} \tag{10}$$

which via (6) provides the PI tuning.

The relay-based PID tuning procedure implemented in the remote laboratory obtains model (2) from an open-loop double-step experiment, and is structured as follows.

Step 1: connect the relay. Exclude the PI and connect to the process the cascade of a relay and an integrator, the switching points of the relay being centred around the present value of the controlled variable with a small hysteresis (so that spurious relay toggles be avoided, but the critical point locus of the relay can still be assumed to be the real negative axis); the integrator slope is a parameter

Step 2: wait for a permanent oscillation. This can be done in several ways. A viable (and in general not very critical) procedure is to count a prescribed number of oscillations (five is here the default) and take the last one as ‘permanent’.

Step 3: compute a point of the process Nyquist curve. With trivial computations, the oscillation induced by the relay-integrator cascade leads to obtain the point of the process Nyquist curve with phase -90°.

Step 4: compute PI parameters. This is done by means of (10) and (6).

For the relay-based PI, we also report the G code of the corresponding VI, included in the CS VI of the experiments using that autotuner (reporting also the code of the step-based PID would be too lengthy here, recall anyway that all of the code is available as free software). Figure 3 shows the front panel (i.e., the user interface and the calling structure) and the block diagram (i.e., the source code in the LabVIEW ‘G’ graphical programming language) of the autotuning PI.

As a final remark, at present no booking or access control mechanism is implemented at CrAutoLab, relying for the latter aspect on the server hosting the site. It would be straightforward to implement those features, however, and if necessary it will be done in the future, with some of the numerous tools readily available for that purpose.

3. PEDAGOGICAL CONSIDERATIONS

By using the step-based autotuning PID, the student cannot avoid getting in contact with important issues, the most important ones being described below:

- the determination of the steady state is critical, and possibly residual motion can alter the nose band measurement;
- though the double step experiment is better than a single-step one (where the problem of detecting the settling

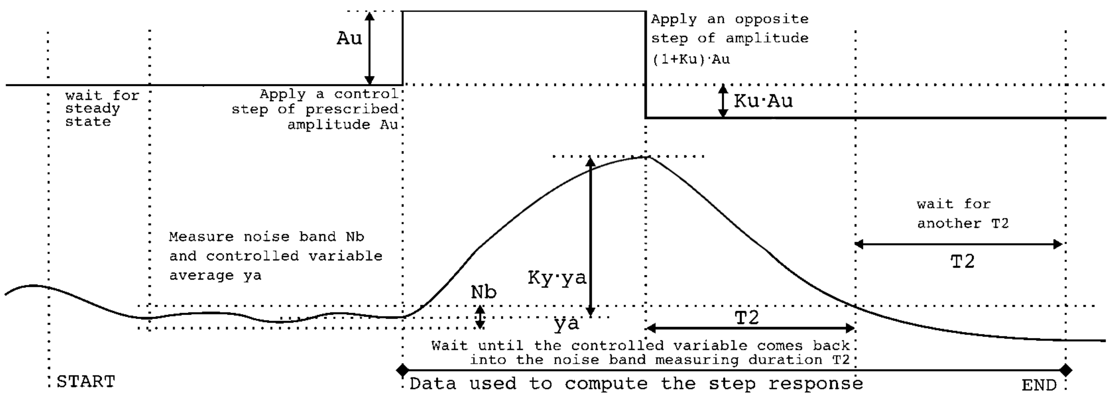
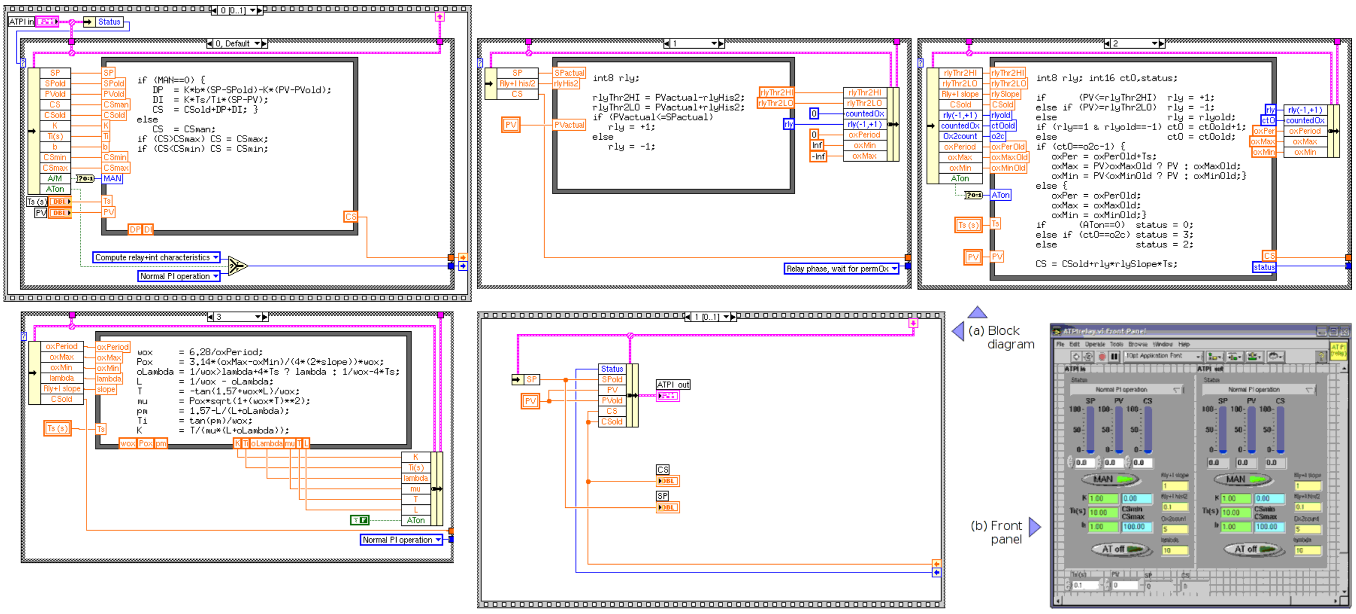

Au
Apply an opposite step of amplitude (1+Ku)·Au
wait for steady state
Apply a control step of prescribed amplitude Au
Ku·Au
Measure noise band Nb and controlled variable average ya
Nb
Ky·ya
ya
wait for another T2
T2
T2
Wait until the controlled variable comes back into the noise band measuring duration T2
START
Data used to compute the step response
END

Fig. 2. Double step experiment.



```
if (MAN==0) {
   DP  = K*b*(SP-SPold)-K*(PV-PVold);
   DI  = K*Ts/Ti*(SP-PV);
   CS  = CSold+DP+DI; }
else
   CS  = CSman;
if (CS>CSmax) CS = CSmax;
if (CS<CSmin) CS = CSmin;
```

```
int8 rly;

rlyThr2HI = PVactual-rlyHis2;
rlyThr2LO = PVactual+rlyHis2;
if (PVactual<=SPactual)
   rly = +1;
else
   rly = -1;
```

```
int8 rly; int16 ct0,status;

if       (PV<=rlyThr2HI)  rly = +1;
else if  (PV>=rlyThr2LO)  rly = -1;
else                      rly = rlyold;
if (rly==1 & rlyold==-1) ct0 = ct0old+1;
else                      ct0 = ct0old;
if (ct0==o2c-1) {
   oxPer = oxPerOld+Ts;
   oxMax = PV>oxMaxOld ? PV : oxMaxOld;
   oxMin = PV<oxMinOld ? PV : oxMinOld;}
else {
   oxPer = oxPerOld;
   oxMax = oxMaxOld;
   oxMin = oxMinOld;}
if      (ATon==0)  status = 0;
else if (ct0==o2c) status = 3;
else               status = 2;

CS = CSold+rly*rlySlope*Ts;
```

```
wox      = 6.28/oxPeriod;
Pox      = 3.14*(oxMax-oxMin)/(4*(2*slope))*wox;
oLambda  = 1/wox>lambda+4*Ts ? lambda : 1/wox-4*Ts;
L        = 1/wox - oLambda;
T        = -tan(1.57+wox*L)/wox;
mu       = Pox*sqrt(1+(wox*T)**2);
pm       = 1.57-L/(L+oLambda);
Ti       = tan(pm)/wox;
K        = T/(mu*(L+oLambda));
```

Fig. 3. Block diagram (a) and front panel (b) of the relay-based autotuning PI.

would be significant), an erratic noise band estimation, or an excessive number of outliers in the previous phases, cn easily impair a correct reconstruction of the step response;

- parameters relative to the step experiment (amplitude, and so on) are critical, and if set ‘incorrectly’ may even prevent the procedure from ending (e.g., if the process gain is so small that the controlled variable does not move enough from the original value);
- the way the FOPDT model is parametrised based on the step response record has a significant impact on the tuning results.

Experimenting with that autotuner is ‘practically’ useful to understand how deeply the various phases of the overall procedure interact, and how many precautions (timeouts, ‘default’ choices, etc.) the designer has to take to obtain a product applicable by the typical plant staff. From a more ‘theoretical’ point of view, the students are led to understand that (a) the parametrisation of the process model cannot be conceptually separated from the tuning phase, and (b) autotuners that require the process to be initially at steady state, and collect process data in the form of time domain responses, are powerful but often critical to operate; to obtain applicable procedures, it is nearly unavoidable to specialise them to a particular type of control loop (e.g. pressure, flow, and so forth, as is actually done in many industrial products) and to maintain a conservative synthesis policy.

By using the relay-based autotuning PI, the students can appreciate ‘in practice’ that it tends to allow less control on the desired closed-loop transients, but on the other hand the overall procedure is far less critical, and less sensitive to ‘slightly’ erratic choices regarding its parameters—in other words, relay-based tuning *seems* less powerful than model-based tuning, but when *implementation* issues come into play (and an engineer cannot neglect such issues), it immediately reveals to be easier to use, and also less keen to behaviours that are inexplicable for non theory-aware staff. It is also possible to investigate the *conceptual* reasons for the facts above, and the students easily understand that a relay experiment is not only less critical than a step one (e.g., no steady state is initially required), but also involves less ambiguities and arbitrarities along the way from the collected input/output data to the process description needed for the tuning.

Having both physical and simulated experiments, see also section 4, is particularly useful for the autotuning domain, both for general reasons (Exel et al., 2000) and owing to how difficult it is to implement an autotuner. i.e.. not only ‘computing some

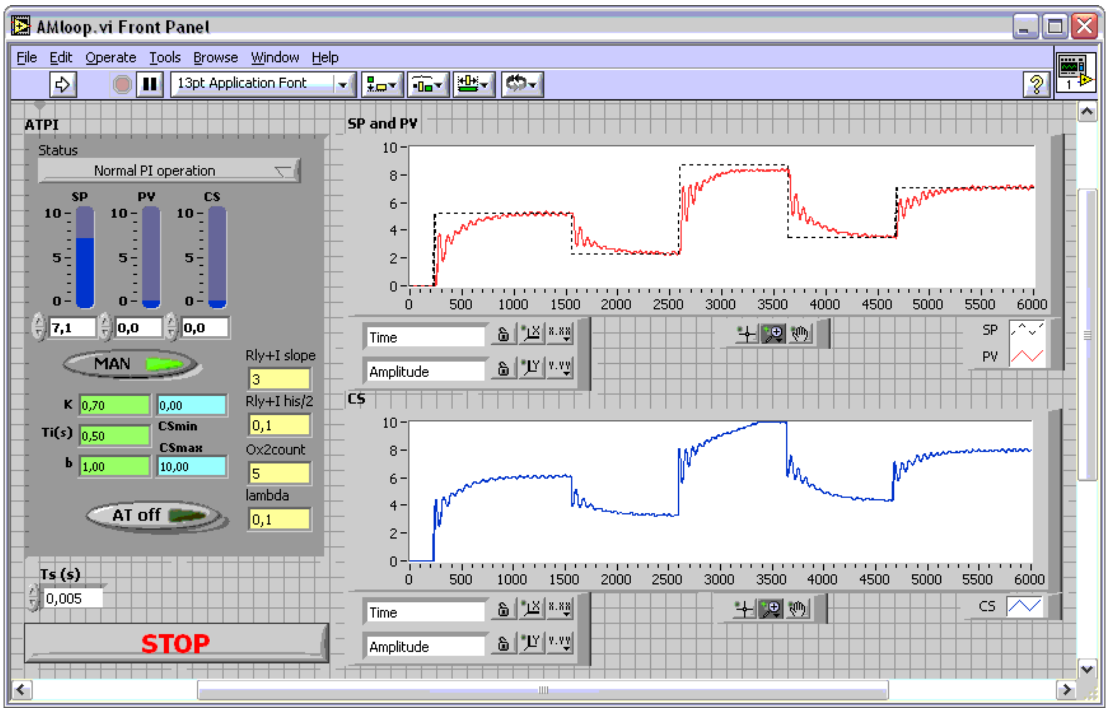


Fig. 4. Front panel of the VI for experiments with the relay-based autotuning PI applied to the speed control apparatus.

parameters and closing a loop', but having that task done by an *autonomous* procedure, running in the absence of theory-aware personnel.

On the same front, looking for a moment also at research and not only at education, the availability of ready-to-run autotuner code, both in the form of remote laboratories and downloadable programs to modify, can be of help *per se* for the scientific community. In fact, many researchers are presently spending a lot of efforts on the investigation of autotuning implementation problems from a control-theoretical standpoint, and those efforts are leading to improved rules (see e.g. Skogestad (2003)) or (which is maybe even more important) to a convergence between the experiment aimed at obtaining the necessary process description, and the subsequent regulator tuning phase. As discussed in works such as Leva and Piroddi (2007), a unitary view of experiment and tuning (together with the recognition of the very peculiar characteristics of the autotuning problem, that for example hinder the application of many neat results of the 'identification for control' domain) is very important for the research on the matter, and to formalise and assess such a unitary view, efficient experimental facilities are very useful.

4. AVAILABLE EXPERIMENTS

The presented autotuners at CrAutoLab are used for single-loop controls, using a temperature control (Leva, 2002) and a speed control apparatus (Leva and Schiavo, 2005).

The first apparatus is composed of a small metal plate heated by two transistors: one of those transistors is used as the actuator to control the plate temperature, while the second one can introduce a load disturbance. This apparatus is quite slow, having open-loop response times of some tenths of seconds, and has of course a well overdamped behaviour. Experimenting with that apparatus gives then a good impression of the typical regulation loop encountered in process control; the presence of the second heating transistor allows to test the performance of autotuners also from the point of view of (load) disturbance rejection—a very important characteristics in process control.

The second apparatus is composed of two LEGO motors. One, driven by a simple current amplifier the input of which is the control signal, acts as 'motor'; the other one, connected to the first by means of an elastic belt, plays at the same time the roles of mechanical load and of tacho generator, its filtered output being the controlled variable. Experimenting with that apparatus gives a good impression of the typical set point tracking problem encountered in servo speed loops.

Besides experiments involving physical control systems, simulated experiments are also available, in which the 'process' is specified as a transfer function. Since the code is downloadable, users with LabVIEW can also replicate the simulated experiments at their own site, also modifying the autotuning code (which of course is not possible with the physical experiments on the CrAutoLab site). It is worth noticing that all of this work is at present in progress: autotuning controls are being tested locally, and will be available on the site as soon as that testing is finished, together with the corresponding LabVIEW code for download as free software.

To provide an idea of the operator interface of the autotuning regulators, figure 4 shows the front panel of the VI used to perform experiments with the relay-based autotuning PI applied to the physical speed control apparatus; the operator interfaces for the other experiments are very similar.

As an example of the available experiments, we report here the autotuning of a PI controller applied to the speed control apparatus, using the procedure of section 2.3, and the VI of figure 4. Figure 5 reports the set point, the controlled variable, and the control signal (in V) during two subsequent tuning experiments. The first tuning is done with quite conservative a value of parameter λ, namely 0.6, while the second tuning is more aggressive, having $\lambda = 0.1$.

After each tuning operation, a couple of set point steps is applied to the obtained control system, to qualitatively evaluate the PI tuning. It can be seen that in both cases the autotuner operates correctly, and the tuning results are in accordance with the requirements made by the operator through parameter λ—a very important fact to make an autotuning regulator an accepted product in the application domain.

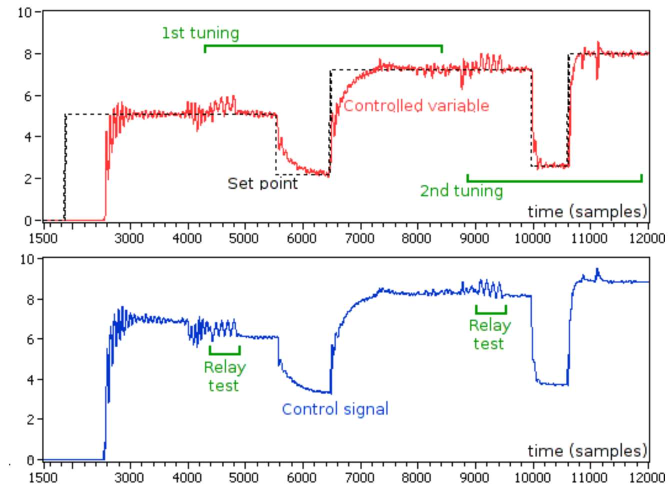


Fig. 5. Results of an experiment with the relay-based autotuning PI applied to the speed control apparatus.

5. CONCLUSIONS AND FUTURE WORK

A remote laboratory devoted to PID autotuning was presented. PID autotuning is seldom present in automatic control remote laboratories, and filling such a lack constitutes a peculiarity of the research described herein.

To maximise educational value from both the theoretical ant the professional points of view, the presented laboratory employs standard, industrial development tools, and makes all the autotuning code available as free software.

At present two autotuning PI/PID regulators are present, and the available experiments include simulated control loops, temperature control, and speed control. Future work will be devoted to the inclusion in the laboratory of other autotuners, possibly of different types. Plans are also underway to consider the autotuning not only of simple (PI/PID) loops, but also of more complex control structures (like e.g. cascade controls), thanks to the flexibility of the physical systems available at the laboratory, with particular reference to the temperature control.

Besides attaining the proposed educational goal, it is the authors' hope that the presented autotuning experiments will help disseminate knowledge on the matter, and possibly stimulate research cooperations to address the difficult problem of formalising the process of turning (auto)tuning formulæ into autotuning procedures.

REFERENCES

B. Aktan, C.A. Bohus, L.A. Crowl, and M.H. Shor. Distance learning applied to control engineering laboratories. *IEEE Transactions on Education*, 39(3):320–326, 1996.

N. Aliane, A. Martínez, A. Fraile, and J. Ortiz. LABNET: a remote control engineering laboratory. *iJOE International Journal of Online Engineering*, pages www.i–joe.org, 2006.

K. J. Åström and T. Hägglund. *PID Controllers: Theory, Design, and Tuning - 2nd Edition*. Instrument Society of America, Research Triangle Park, NY, 1995.

K. J. Åström and A.B. Östberg. A teaching laboratory for process control. *IEEE Control Systrems Magazine*, pages 37–42, october 2006.

P. Balda, M. Schlegel, and M. Štětina. An open remote and virtual laboratory for automatic control teaching. *Journal of Cybernetics and Informatics*, Special Issue "New Trends in Education of Automation and Information Technology":36–41, 2004.

S. Dormido Bencomo. Control learning: present and future. *Annual Reviews in Control*, 28(1):115–136, 2004.

M. Exel, S. Gentil, F. Michau, and D. Rey. Simulation workshop and remote laboratory: two web-bsed taining approaches for control. In *Proc. ACC 2000*, Chicago, IL, 2000.

C.C. Hang and T.H. Lee. Incorporating practical contents in control engineering courses. *IEEE Transactions on Education*, 33(3):279–284, 1990.

A. Leva. A hands-on experimental laboratory for undergraduate courses in automatic control. *IEEE Transactions on Education*, 46(2):263–272, 2002.

A. Leva. Simple model-based PID autotuners with rapid relay identification. In *Proc. 16th IFAC World Congress*, Praha, Czech Republic, 2005.

A. Leva. Real-time web-based interactive control experiments with fast sampling times. In *Proc. ACE 2006*, Madrid, Spain, 2006.

A. Leva and A.M. Colombo. On the IMC-based synthesis of the feedback block of ISA-PID regulators. *Transactions of the Institute of Measurement and Control*, 26(5):417–440, 2004.

A. Leva and L. Piroddi. On the parametrisation of simple process models for the autotuning of industrial regulators. In *Proc. 1ACC 2007*, New York, NY, 2007.

A. Leva and F. Schiavo. Low-cost flexible speed control experiments. In *Proc. 16th IFAC World Congress*, Praha, Czech Republic, 2005.

A. Leva, C. Cox, and A. Ruano. Hands-on PID autotuning: a guide to better utilisation. *IFAC Professional Brief*, 2002.

Y. Li, K.H. Ang, and C.Y. Chong. Patents, software, and hardware for PID control—an overview and analysis of the current art. *IEEE Control Systrems Magazine*, pages 42–54, february 2006.

M. Morari and E. Zafiriou. *Robust process control*. Prentice-Hall, Upper Saddle River, NJ, 1989.

A. O'Dwyer. *Handbook of PI and PID Controller Tuning Rules*. World Scientific Publishing, Singapore, 2003.

D.E. Rivera, S. Skogestad, and M. Morari. Internal model control 4: PID controller design. *Industrial Engineering Chemistry Process Design Development*, 25:252–265, 1986.

S. Skogestad. Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, 13:291–309, 2003.

A. Valera, J.L. Diez, M. Valles, and P. Albertos. Virtual and remote control laboratory development. *IEEE Control Systems Magazine,*, 25(1):35–39, 2005.

S. Yurkovich and K.M. Passino. A laboratory course for teaching intelligent control. In *Proc. 35th Conference on Decision and Control*, pages 3898–3983, Kobe, Japan, 1996.