Chapter 10
Rapid Prototyping Environment for Control Systems Implementation

Damir Vrančić

10.1 Introduction

Two trends can be highlighted currently in the field of process control. First, the number of control loops is rapidly increasing. Among them, the share of unstable, higher order, and nonlinear processes, processes with dead-time, and processes controlled over a network is also increasing [8, 19]. Second, ongoing cost-reduction activities in companies have resulted in staff minimisation, especially in the domain of maintenance. Moreover, this activity is being outsourced to specialised companies. Thus, the amount of knowledge in industries related to control is rapidly decreasing [18].

In order to cope with the ever-increasing number of control loops, industrial end-users and engineering companies are facing a need for efficient software support that would shorten the time required for commissioning and maintaining control loops. On the other hand, software support should also foster the implementation of advanced control algorithms in production processes and plants.

Some software tools supporting control design activities in the process industry have been commercially available for approximately two decades. Nowadays, the most well-known products are INTUNE™ [11], BESTune [1], ExperTune [6], INCA PID Tuner [12], Sintolab [9], U-Tune PID [21], and LOOP-PRO™ TUNER [4], which all support PID controller design and tuning. Some of the products also perform closed-loop analysis and simulation. Yet the main disadvantages remain that (i) they focus mainly on PID controllers and (ii) validation of closed-loop behaviour is carried out off-line.

Recently, the MATLAB®/Simulink® programming environment implemented real-time connection with processes via OPC (OLE for Process Control) signals and off-line tuning of PID controllers [16, 17]. Due to the on-line OPC connection and available control toolboxes, it can also be used for rapid design of more

D. Vrančić (✉)
Department of Systems and Control, Jožef Stefan Institute, Ljubljana, Slovenia

complex control algorithms based on mathematical models. However, the software is relatively expensive and requires the installation of MATLAB/Simulink with selected toolboxes. Moreover, MATLAB is a relatively slow interpreting language and the experimentation environment is rather complex, hence less suitable for system integrators, especially those in small- and medium-sized enterprises.

The needs of industry, on the one hand, and the deficiencies of the existing tools, on the other hand, motivated us to design and develop a new tool that would enable rapid prototyping of classical and advanced control methods in an industrial environment. In this way various control approaches could be quickly verified before making the final decision on the selection of a particular solution and its implementation. Such a tool would help control designers and system integrators as well as process personnel in industry to substantially reduce the time needed for the development cycle.

The basic requirements for the tool, referred to as the "Rapid Prototyping (RaPro) environment" were as follows: the environment should (i) enable simple access to real plants via OPC, (ii) enable efficient controller design, (iii) enable direct on-line validation at the target plant, (iv) be independent of the control hardware, i.e., it should be applicable to a wide family of process platforms, (v) enable execution of on-line open-loop or closed-loop experiments on the process, data conditioning, process model identification, controller tuning, and report generation, and (vi) consist of only one executable file without software installation.

The aim of this chapter is to present the concept and architecture of RaPro, to introduce its main functions, and to illustrate its use with practical examples.

The chapter is organised as follows. First, the concept of the RaPro environment is explained. Then, the architecture and functions of the rapid prototyping environment are introduced. The use of the prototyping environment is illustrated with two examples. The first one is temperature control in a hydrogen fuel cell power module, and the second one is level control in a three-water-tank laboratory setup. The RaPro environment is then evaluated in the context of theory/practice issues. The chapter ends with conclusions.

10.2 The Concept and Control Structures of the Rapid Prototyping Environment

The main motive behind the development of the RaPro environment is to shorten and simplify the design, commissioning and maintenance of control loops and also to increase the work efficiency of system integrators, engineers and maintenance staff. In order to achieve these goals a set of control structures, types of experiments, and verification strategies had to be selected. Namely, careful selection of control structures, experiments and verification strategies can significantly improve control efficiency, decrease commissioning time and simplify operation for a broad range of users.

10.2.1 The Concept of the Environment

The concept on which the proposed tool is built tries to combine some characteristics of the existing environments which support the control design stage. In general, there are two types of control design approaches. The first one is related to control of machines and devices (e.g., in the aerospace or automotive industries), and the second one to production processes (e.g., in the chemical, pharmaceutical, or other kinds of process industries) [20]. The same distinction applies to control engineering tools [3].

When machines and devices are concerned (like e.g. in mechatronics [13]), the tools typically support theoretical (first principle) modelling, the design of complex control schemes and algorithms, testing the algorithms using a mathematical model, automatic code generation, hardware-in-the-loop verification, etc. The advantage of this approach, often referred to as "Rapid Control Prototyping" (RCP) [2, 10], is that it can rely on professional universal tools (e.g., MATLAB/Simulink, dSPACE [5]) which support a practically unlimited set of solutions, and it also offers on-line verification of design results. However, the approach is often very time consuming and requires a high level of expertise of engineers.

In the case of complex production processes, the tools are more oriented towards experimental modelling, the use of predefined controllers or control structures, the tuning of parameters, off-line verification of control performance, and downloading control parameters into the target platform. These tools are less flexible and do not support on-line verification on a prototyping platform, but are in turn less time consuming and more adapted to engineers with a lower level of expertise.

The proposed concept behind our Rapid Prototyping environment is, on the one hand, similar to what we see in PID-tuners for the process industry because it is based on fixed control structures in order to simplify the engineering process and reduce the required expertise of engineers. However, to compensate for the loss of flexibility and universality caused by this decision, it attempts to introduce a larger set of predefined structures. On the other hand, it includes on-line verification of the control algorithms using a prototyping platform which is a typical feature of RCP environments. Note that on-line verification of algorithms is in practice often very important because it enables better selection of final HW and SW platforms and/or proper adaptation of available control algorithms.

Such a concept and the resulting prototyping environment, in our experience, is in line with what engineers, especially those from small- and medium-sized enterprises, need when solving practical control problems.

10.2.2 Feedback Structures and Controller Types

The Rapid Prototyping environment supports several feedback control structures and controller types. The currently supported feedback structures are single-loop

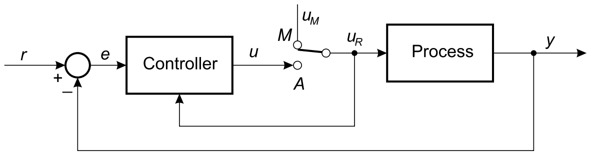


Fig. 10.1 A single-loop controller structure

control, cascade control loop, feed-forward control loop, and a multivariable control structure with two inputs and two outputs (a TITO structure).

A single control loop structure is shown in Fig. 10.1. It consists of a single-input-single-output (SISO) controller, a switch for manual (M) to automatic (A) mode, and the process. The feedback signal u_R to the controller is used for anti-windup (bumpless) protection.

A cascade control loop consists of two controllers and one process with two outputs: an inner signal y_i and an outer signal y_o, and one input (u) to the system (see Fig. 10.2). Switches for manual/automatic control are present but are not shown for the sake of brevity. The main task of the inner loop is to speed-up the inner system in order to decrease lag time for the outer process. Therefore, the outer closed-loop system can be made faster and more efficient.

The feed-forward (FF) control loop consists of a compensator G_{FF}, which improves the closed-loop disturbance rejection performance by measuring and compensating for disturbance d (see Fig. 10.3). If the compensator transfer function becomes

$$G_{FF}(s) = -\frac{G_{DY}(s)}{G_P(s)} \tag{10.1}$$

where $G_P(s)$ is the process transfer function, then the measured disturbance is completely eliminated.

Multivariable systems have more than one input and output. The RaPro environment supports a two-input/two-output (TITO) controller structure. A typical closed-loop control configuration is shown in Fig. 10.4. The TITO controller consists of two independent controllers c_1 and c_2 and two decouplers d_1 and d_2. The decouplers' main task is to decouple a multivariable process into two independent (or less coupled) SISO systems. In an ideal case, there are no interactions between the first controller output v_1 and the second process output y_2, and vice versa. Therefore, controllers c_1 and c_2 can be designed independently, as in the SISO case. The TITO

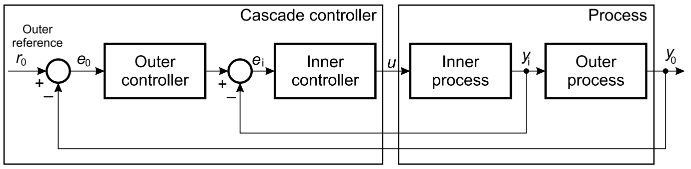


Fig. 10.2 Cascade controller structure

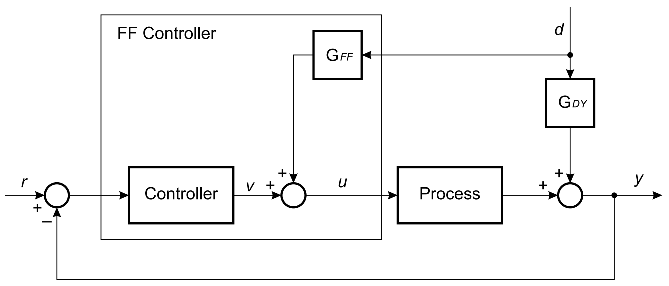
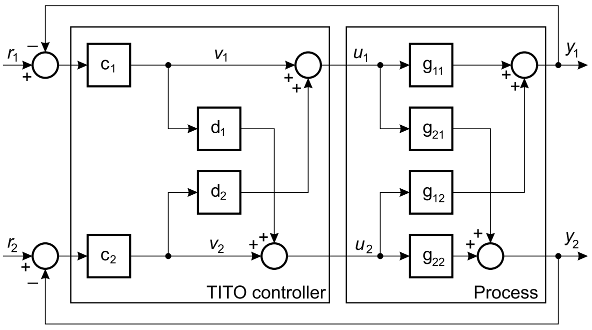


Fig. 10.3 Feed-forward controller structure

Fig. 10.4 The TITO controller structure



system in Fig. 10.4 is ideally decoupled when

$$d_1(s) = -\frac{g_{21}(s)}{g_{22}(s)} \tag{10.2}$$

$$d_2(s) = -\frac{g_{12}(s)}{g_{11}(s)} \tag{10.3}$$

By comparing the TITO controller and FF compensator structures, it can be seen that they are equivalent when the process transfer function G_P is g_{11}, $d_1 = 0$, $d_2 = G_{FF}$, $c_2 = 0$, $g_{12} = G_{DY}$, and $g_{21} = g_{22} = 0$. Therefore, the feed-forward compensator is only a subset of the TITO controller and its parameters can be calculated from the TITO controller.

The controller types supported by the RaPro environment are the PID controllers and Smith predictors.

The PID controller structure is shown in Fig. 10.5. It consists of a reference pre-filter, PID terms, an output filter, limitations and a manual/automatic switch. Anti-windup protection is realised by the feedback loop with gain $1/K_a$. Selection of $K_a = K$ usually gives the best anti-windup protection.

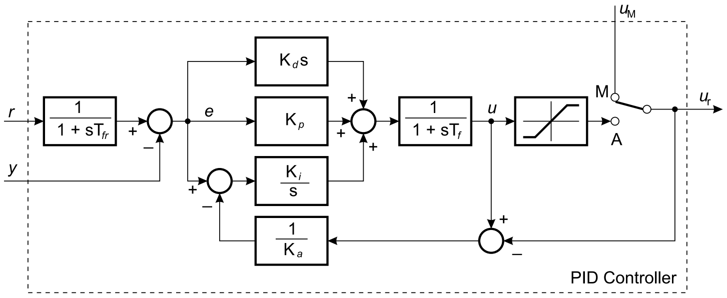


Fig. 10.5 Block scheme of the PID controller

The PID controller output can be given by the following expression in Laplace form:

$$U(s) = \frac{K_P + K_d s}{1 + sT_f} E(s) + \frac{K_i}{s(1 + sT_f)} \left(E(s) - \frac{U(s) - U_r(s)}{K_a} \right) \tag{10.4}$$

where E denotes control error

$$E(s) = \frac{R(s)}{1 + sT_{fr}} - Y(s) \tag{10.5}$$

and s is a complex variable.

The parameters K_p, K_i and K_d are the proportional, integral and derivative gains of the PID controller, respectively. The parameter T_f is a filter time constant, which should be selected according to the process noise. Typical values are

$$\frac{K_d}{4K_p} < T_f < \frac{K_d}{10K_p} \tag{10.6}$$

Parameter T_{fr} is a reference pre-filter and can be used to optimise reference following and disturbance rejection performance.

The Smith predictor is shown in Fig. 10.6. It consists of the process model and the PID controller. If the process model perfectly matches the actual process, signals y and y_{md} are the same. In this case, the PID controller actually controls the process without a time delay [29]. Therefore, the Smith predictor is very efficient in controlling processes with larger time delays (compared to the main time constant of the process).

The compensator G_{FF} (Fig. 10.3), decouplers d_1 and d_2 (Fig. 10.4), and the process model in the Smith predictor (Fig. 10.6) are realised by the following transfer function:

$$G(s) = \frac{K_{PR} e^{-sT_{del}}}{1 + a_1 s + a_2 s^2} \tag{10.7}$$

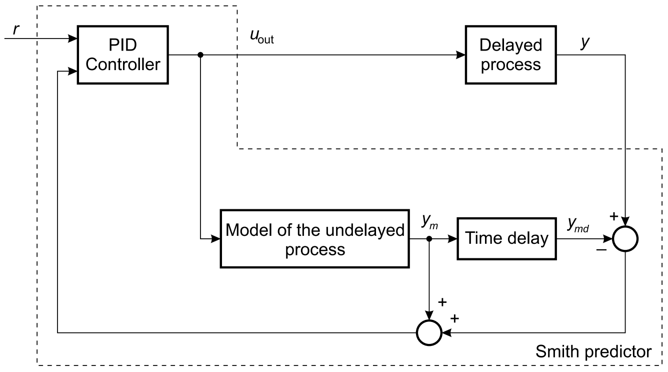


Fig. 10.6 Smith predictor block diagram

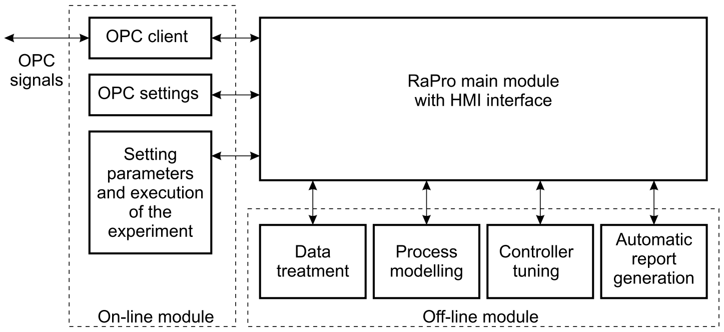
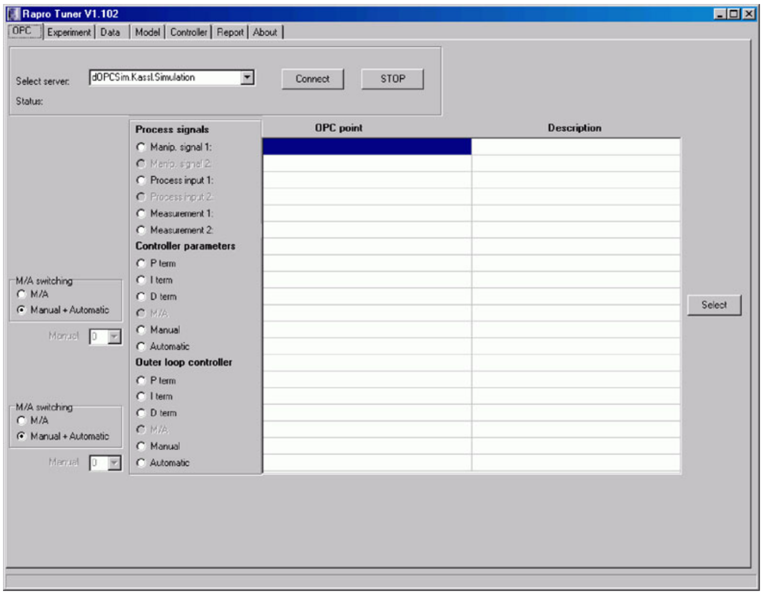

OPC signals
OPC client
OPC settings
Setting parameters and execution of the experiment
On-line module
RaPro main module with HMI interface
Data treatment
Process modelling
Controller tuning
Automatic report generation
Off-line module

Fig. 10.7 Block diagram of the RaPro environment

where K_{PR} and T_{del} are the steady-state gain and pure time delay, respectively. Parameters a_1 and a_2 are dynamic parameters (the sum and multiplication of the two time constants, respectively).

10.3 The Architecture and Functions of the Rapid Prototyping Environment

The RaPro environment consists of three modules: the on-line module, the off-line module, and the main module. The on-line module enables users to measure process signals and to perform open-loop or closed-loop experiments on the process. The main task of the off-line module is to filter the data obtained by the experiments and to calculate the process model and controller parameters. The main module enables communication among the sub-modules and supports human-machine interaction. The block scheme of the RaPro environment is given in Fig. 10.7.



Select server: dOPCSim.Kassl.Simulation

M/A switching: M/A [✓ Manual + Automatic]

M/A switching: M/A [✓ Manual + Automatic]

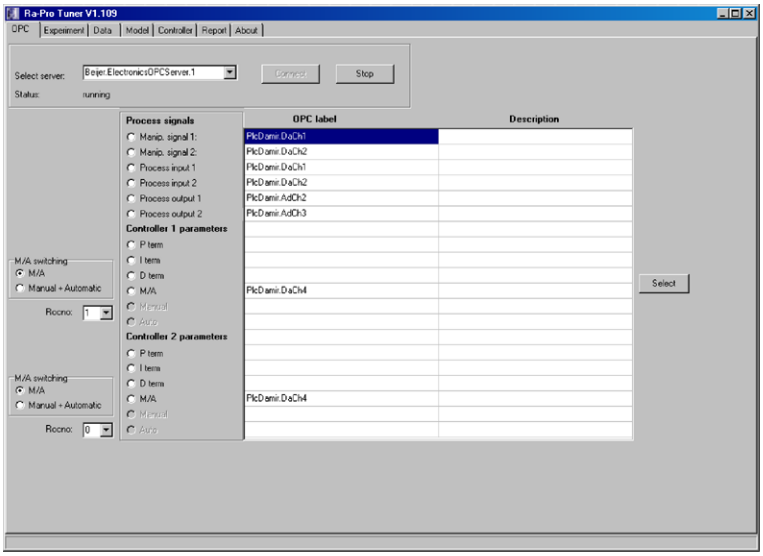
Fig. 10.8 OPC settings for SISO systems

The RaPro environment has been realised as a program package. The main functions are the following:

- selection of the OPC server and OPC points for the process signals and controller parameters;
- setting the experiment parameters and execution of the experiment;
- data treatment;
- calculation of the process model;
- calculation of controller parameters; and
- automatic report generation.

The HMI of the RaPro environment consists of the main program window (consisting of several tabs) and some smaller windows which are opened during execution of an experiment or for filling in some specific data related to the current task. The main program window is shown in Fig. 10.8. Tabs at the top of the main window define the different functions which can be performed with the RaPro environment. A detailed explanation of each function with related tasks will be given in the following sub-chapters.

Due to the complexity of multivariable systems, the RaPro environment was separately developed for single-input/single-output (SISO) and for two-input/two-output (TITO) systems. Herein, the main emphasis will be devoted to the SISO development environment. However, the TITO environment will be mentioned where it significantly differs from the SISO environment.



Select server: Beijer.ElectronicsOPCServer.1
Status: running

M/A switching: [✓ M/A] Manual + Automatic
Rocno: 1

M/A switching: [✓ M/A] Manual + Automatic
Rocno: 0

	OPC label	Description
Process signals		
Manip. signal 1:	PlcDamir.DaCh1	
Manip. signal 2:	PlcDamir.DaCh2	
Process input 1	PlcDamir.DaCh1	
Process input 2	PlcDamir.DaCh2	
Process output 1	PlcDamir.AdCh2	
Process output 2	PlcDamir.AdCh3	
Controller 1 parameters		
P term		
I term		
D term		
M/A	PlcDamir.DaCh4	
Manual		
Auto		
Controller 2 parameters		
P term		
I term		
D term		
M/A	PlcDamir.DaCh4	
Manual		
Auto		

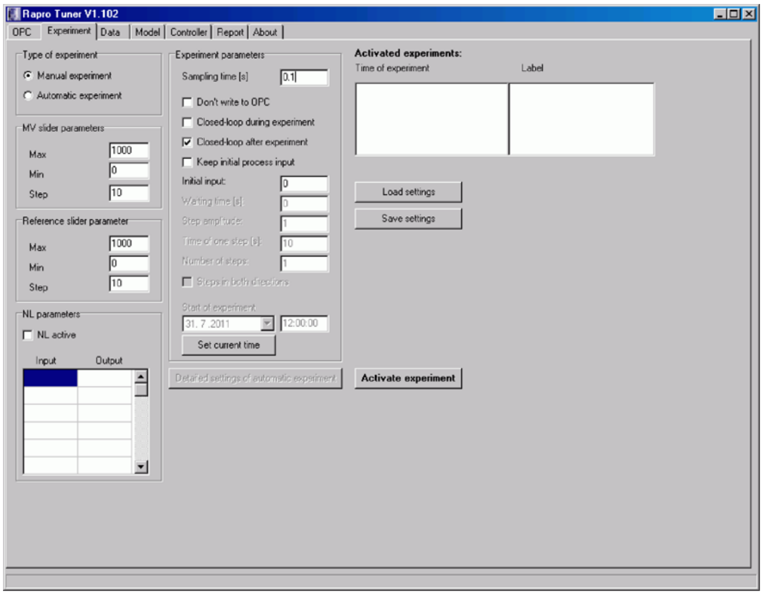
Fig. 10.9 OPC settings for TITO systems

10.3.1 Selection of the OPC Server and OPC Points for the Process Signals and Controller Parameters

Figure 10.8 shows the OPC window (tab "OPC"). The main task of the OPC window is to select the OPC server, process signals, and controller terms. Here we can select the appropriate OPC server and start or stop communication. The required process signals are the following: the process input (usually the same as the controller output), the process output (a measured or controlled variable), and the manipulative (manipulating) signal. The manipulative signal is the signal which can actually change the process steady-state. In an experiment with an external controller[1] connected to the process and running in the open-loop configuration (manual mode), the manipulative signal is the manual control signal. On the other hand, if an external controller is running in the closed-loop configuration (the external controller is in automatic mode), the manipulative signal is the reference (set-point) of the controller. In order to be able to switch between the manual and automatic mode of the controller, the signal which defines the manual and automatic modes should be selected as well.

Similarly as for SISO systems, the OPC window for the TITO system is shown in Fig. 10.9. Consequently, two manipulative and two process input signals should be selected.

[1]The external controller is a PID controller which actually controls the process. It is either a standalone controller or running within the PLC.



Type of experiment
- [✓ Manual experiment]
- Automatic experiment

MV slider parameters
- Max 1000
- Min 0
- Step 10

Reference slider parameter
- Max 1000
- Min 0
- Step 10

NL parameters
- ☐ NL active

Input	Output

Experiment parameters
- Sampling time [s] 0.1
- ☐ Don't write to OPC
- ☐ Closed-loop during experiment
- ☑ Closed-loop after experiment
- ☐ Keep initial process input
- Initial input: 0
- Waiting time [s]: 0
- Step amplitude: 1
- Time of one step [s]: 10
- Number of steps: 1
- ☐ Steps in both directions
- Start of experiment 31. 7. 2011 12:00:00

Activated experiments:

Time of experiment	Label

Fig. 10.10 Setting the parameters of the experiment

10.3.2 Setting the Parameters and Execution of the Experiment

This function is activated by clicking the tab "Experiment" in the upper part of the main program window (see Fig. 10.10). The window which appears is intended for setting the parameters of the experiment and starting the execution of the experiment.

10.3.2.1 Setting the Parameters of the Experiment

There are two types of experiments: manual and automatic. A manual experiment is executed entirely by the user (by changing sliders for manipulative signal during the experiment), while an automatic experiment is carried out without user interaction.

Both types of experiment require the definition of the slider range for the manipulative signal (MV) and sampling time. Here we can also define the controller mode during and after the experiment (open-loop or closed-loop mode) and whether the manipulative signal should change when starting the experiment. There is an additional "Don't write to OPC" checkbox. If it is ticked, the RaPro environment will only read OPC signals and will not write to the OPC. This option is added for security reasons.

The TITO version of the experiment screen is very similar, except that it contains a definition of two MV and reference sliders.

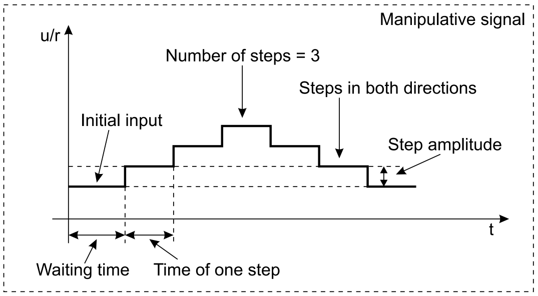


Fig. 10.11 Definition of the parameters for the manipulative signal in an automatic experiment on the process

The automatic experiment is more complex, since several additional parameters should be defined to run the experiment without user interaction. The most important ones are the idle time (waiting time) before applying a step-change to the manipulative signal, the amplitude of the step-signal, and the duration of one step. It is also possible to apply several steps (with the option to return back to the initial value of the manipulative signal). An automatic experiment also enables the selection of the starting time of the experiment (with the date and time). The meaning of some of the parameters is more clearly depicted in Fig. 10.11.

10.3.2.2 Execution of an Experiment

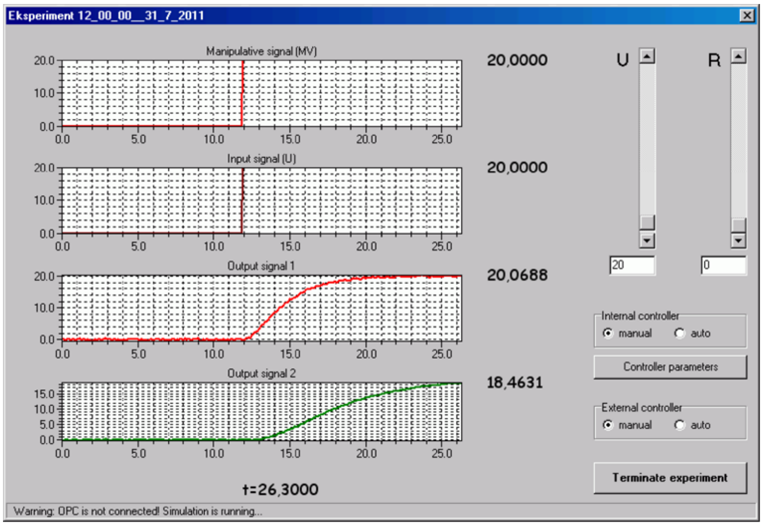
An experiment (manual or automatic) is started by clicking the button "Activate experiment". When the experiment is started (note that an automatic experiment will actually start at a defined date/time), an experiment window is opened (see Fig. 10.12). The most important functions of the experiment window are the following:

- displays the process input and output signals; and
- modifies the manipulative signal (process input or reference) in order to change the steady-state of the process (by changing the slider position).

When performing a manual experiment, an experiment on the process should be simple but informative enough to be able to calculate a process model of sufficient quality. One of the simplest experiments on the process is to change the process steady-state by applying a step-change at the process input. Note that it is not required that the process transients die out completely.

The experiment is terminated by clicking the button "Terminate experiment" or if the amount of time for an automatic experiment has expired.

The TITO version of the experiment window shows more signals (two manipulative variables, two process inputs, and two process outputs) (see Fig. 10.13). During an experiment on the TITO system, the first process input should be modified. When both process output signals settle, a change to the second process input should be performed, as shown in Fig. 10.13.



Warning: OPC is not connected! Simulation is running...

Fig. 10.12 An experiment window during execution of an experiment on the SISO system

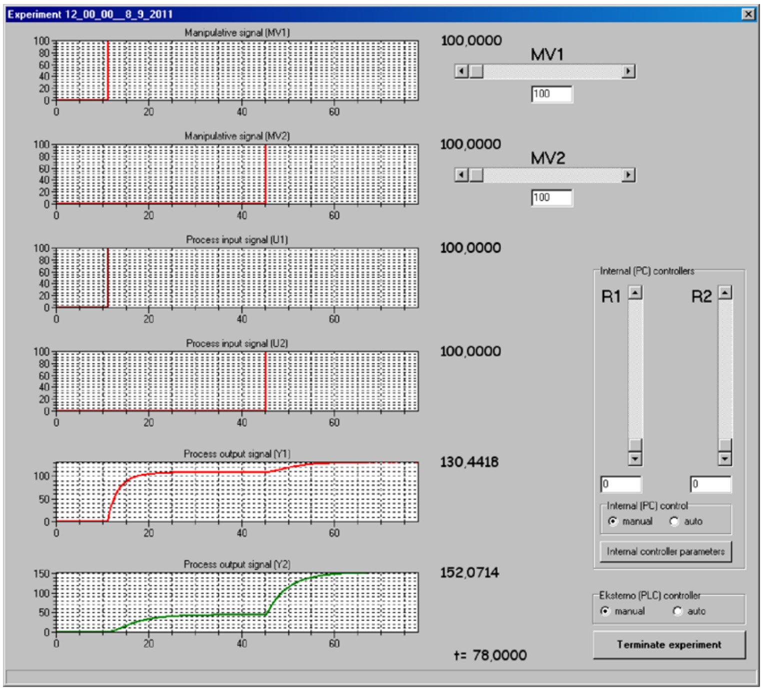
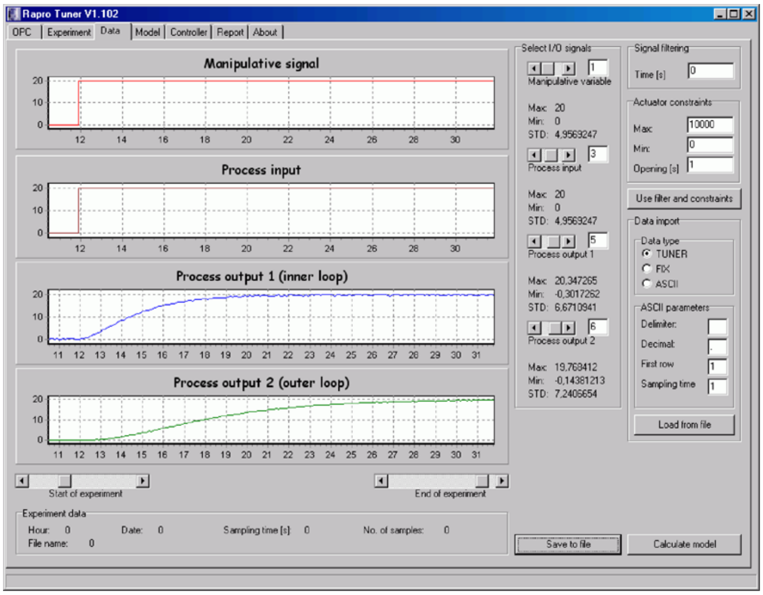

Experiment 12_00_00__8_9_2011
Manipulative signal (MV1)
Manipulative signal (MV2)
Process input signal (U1)
Process input signal (U2)
Process output signal (Y1)
Process output signal (Y2)
100,0000
MV1
100
100,0000
MV2
100
100,0000
100,0000
130,4418
152,0714
Internal (PC) controllers
R1
R2
0
0
Internal (PC) control
manual
auto
Internal controller parameters
Eksterno (PLC) controller
manual
auto
Terminate experiment
t= 78,0000

Fig. 10.13 Experiment window during execution of an experiment on the TITO system


Rapro Tuner V1.102
OPC | Experiment | Data | Model | Controller | Report | About
Manipulative signal
Process input
Process output 1 (inner loop)
Process output 2 (outer loop)
Start of experiment
End of experiment
Use filter and constraints
Load from file
Save to file
Calculate model

Select I/O signals

Manipulative variable: 1
Max: 20
Min: 0
STD: 4,9563247

Process input: 3
Max: 20
Min: 0
STD: 4,9563247

Process output 1: 5
Max: 20,347265
Min: -0,3017262
STD: 6,6710941

Process output 2: 6
Max: 19,768412
Min: -0,14381213
STD: 7,2405654

Signal filtering
Time [s]: 0

Actuator constraints
Max: 10000
Min: 0
Opening [s]: 1

Data import
Data type: (●) TUNER ( ) FIX ( ) ASCII

ASCII parameters
Delimiter:
Decimal: .
First row: 1
Sampling time: 1

Experiment data
Hour: 0 Date: 0 Sampling time [s]: 0 No. of samples: 0
File name: 0

Fig. 10.14 Data treatment window for SISO systems

10.3.3 Data Treatment

The data treatment window (Fig. 10.14) is opened after the experiment is terminated (or by clicking the tab "Data" in the main window). Its main purpose is to display the measurements. The measurements can be additionally truncated (by using sliders) or filtered by applying a first-order filter if the signals are too noisy.

Moreover, it often happens that the actuator signal is limited (e.g., the opening of the valve cannot be higher than 100 % or lower than 0 % and the actual opening time depends on the speed of the actuator motor). Since the actual process input can be different from the controller output signal, the process model might become inaccurate. Therefore, the prototyping environment offers additional fields to enter the actuator's constraints in order to calculate the actual process input signals and therefore reduce process model error in subsequent stages. The required parameters are the minimum and maximum values of the actuator signal and the time required for the actuator to change from the minimum to the maximum value.

If the displayed signals are too small, they can be enlarged by double clicking the corresponding graph (see Fig. 10.15). A histogram of the selected signal is obtained by right-clicking the graph. The minimum and maximum values and the standard deviation (STD) of the signals are given on the right side of the displayed signals. Decimal points are denoted with commas due to local operating system.

When the data is truncated, filtered, and limited, the RaPro environment can calculate the process model by clicking the button "Calculate model".

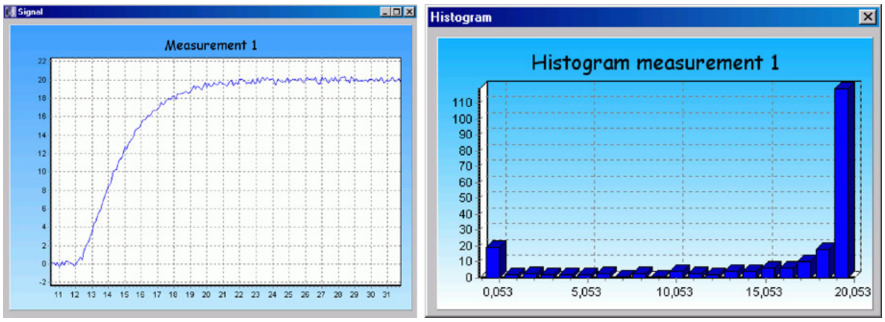
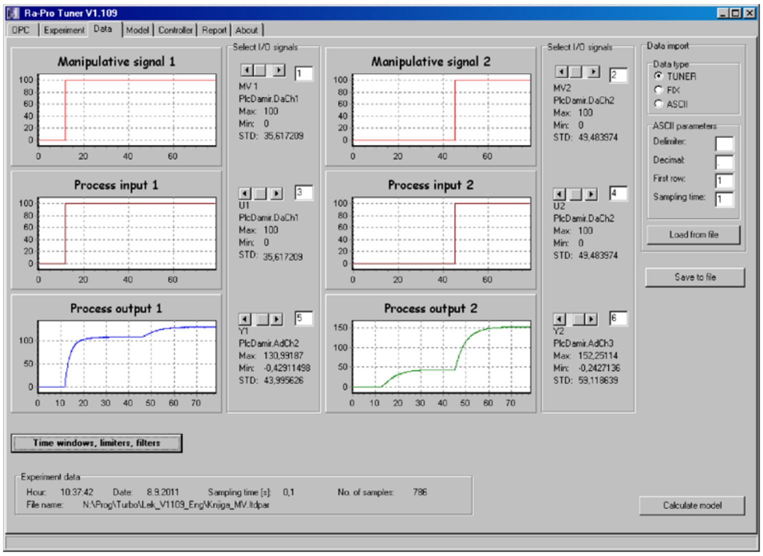


Fig. 10.15 Detailed view of the measured signal (by double-clicking signal graph) and histogram of the measured signal (by right-clicking signal graph)



Select I/O signals

1
MV 1
PlcDamir.DaCh1
Max: 100
Min: 0
STD: 35,617209

3
U1
PlcDamir.DaCh1
Max: 100
Min: 0
STD: 35,617209

5
Y1
PlcDamir.AdCh2
Max: 130,99187
Min: -0,42911498
STD: 43,995626

Select I/O signals

2
MV2
PlcDamir.DaCh2
Max: 100
Min: 0
STD: 49,483974

4
U2
PlcDamir.DaCh2
Max: 100
Min: 0
STD: 49,483974

6
Y2
PlcDamir.AdCh3
Max: 152,25114
Min: -0,2427136
STD: 59,118639

Data import
Data type
[✓ TUNER] FIX ASCII
ASCII parameters
Delimiter:
Decimal: .
First row: 1
Sampling time: 1

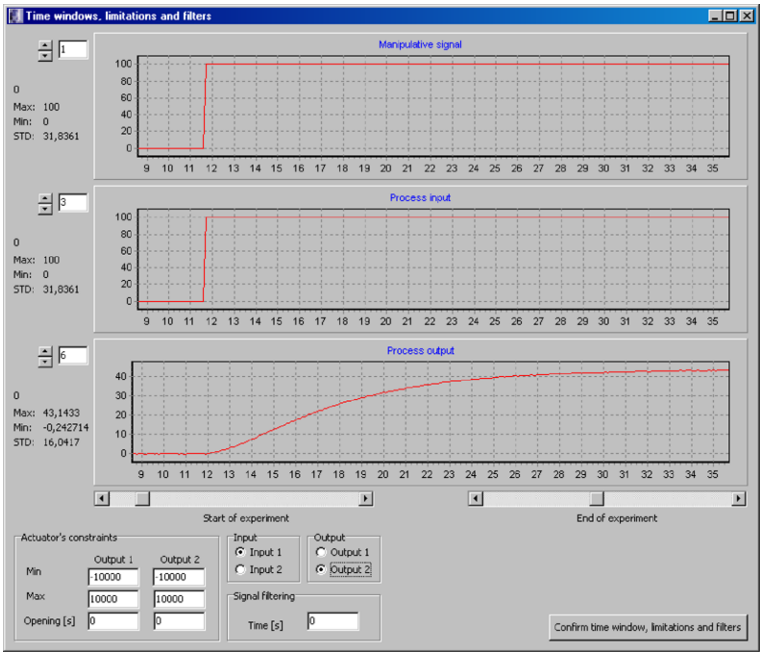
Experiment data
Hour: 10:37:42 Date: 8.9.2011 Sampling time [s]: 0,1 No. of samples: 786
File name: N:\Prog\Turbo\Lek_V1109_Eng\Knjiga_MV.ltdpar

Fig. 10.16 Data treatment window for TITO systems

The TITO user interface (Fig. 10.16) shows four time responses. Each combination of process input-output responses can have different start and/or termination times. Therefore, they should be defined separately by clicking the button “Time windows, limiters, filters”, as shown in Fig. 10.17.

10.3.4 Calculation of the Process Model

The model is shown in graphic and analytic form under the tab “Model” (see Fig. 10.18). It displays the process model parameters and provides a comparison



1
0
Max: 100
Min: 0
STD: 31,8361

3
0
Max: 100
Min: 0
STD: 31,8361

6
0
Max: 43,1433
Min: -0,242714
STD: 16,0417

Actuator's constraints

	Output 1	Output 2
Min	-10000	-10000
Max	10000	10000
Opening [s]	0	0

Input: [✓ Input 1] Input 2

Output: Output 1 [✓ Output 2]

Signal filtering: Time [s] 0

Fig. 10.17 Definition of the time window, limiter, and filter for a chosen process input and/or output

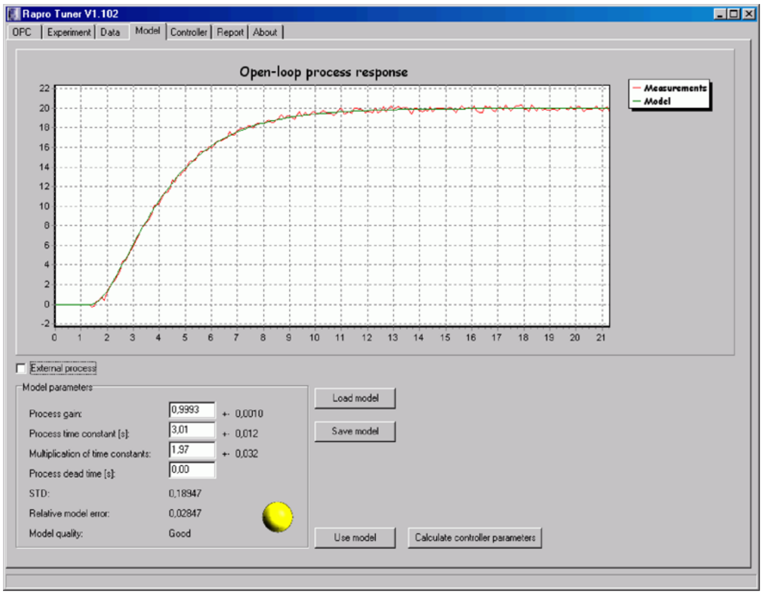
between the measurement and the model in graphic form. The obtained linearised continuous-time process model is a second-order model with delay, as shown in Eq. (10.7).

The process model is obtained by using a least-squares system identification with instrumental variables and state-variable filters [7]. The algorithm calculates the most appropriate process structure (a delayed process, first-order delayed process, or second-order delayed process) from measured data for different time delays. The process model with the smallest sum of squared error between the model response and the measurements is then selected.

It is important to validate the obtained model. If the quality of the model is inappropriate, the calculated controller parameters can be far from ideal. In some cases we have to change the truncating, filtering, and/or limiting parameters in the data treatment stage, change the model manually, or even repeat the experiment. Some model identification and validation methods (e.g., correlation of error vs. input signals) were omitted since we are using a linearised process model around a selected working point.

The relative model error is calculated as the ratio between two standard deviations,

$$E_R = \frac{\sigma(y_m - y)}{\sigma(y)} \tag{10.8}$$



Process gain:	0,9993	+- 0,0010
Process time constant [s]:	3,01	+- 0,012
Multiplication of time constants:	1,97	+- 0,032
Process dead time [s]:	0,00	
STD:	0,18947	
Relative model error:	0,02847	
Model quality:	Good	

Fig. 10.18 Process model window

where y and y_m represent the measured and the process model responses, respectively.

The TITO user interface is very similar, therefore it will not be shown here. The only difference is that the user selects the desired process sub-model by choosing the proper process input and the proper process output.

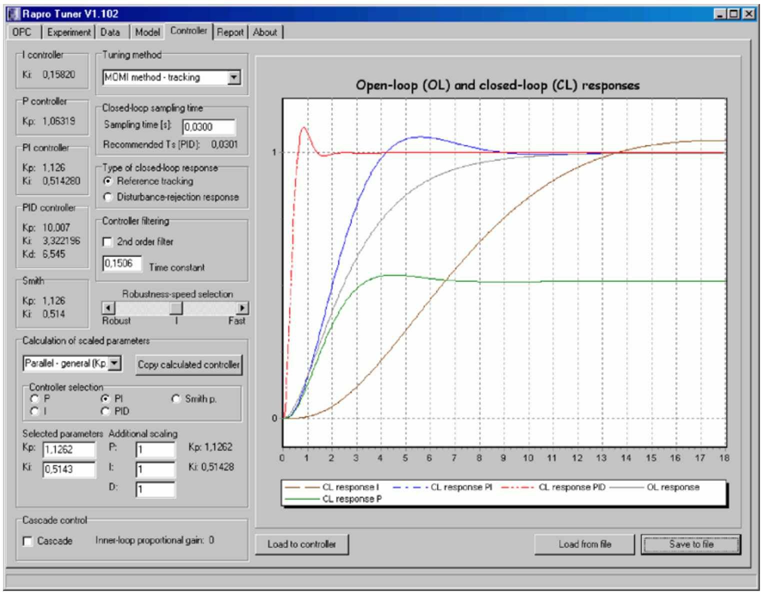
When the quality of the process model is high enough, the controller parameters can be calculated by clicking the button "Calculate controller parameters".

10.3.5 Calculation of Controller Parameters

The user interface for the tab "Controller" is shown in Figs. 10.19 and 10.20 for SISO and for TITO systems, respectively. The difference is in the higher number of controllers and additional decouplers in the TITO interface.

The most important tasks of the window for the controller parameters are to:

- select the controller type and structure;
- select the tuning method;
- select the degree of robustness;
- select the closed-loop sampling time;
- select the tracking/disturbance rejection response;
- display the controller parameters; and
- graphically present the anticipated closed-loop response.



I controller
Ki: 0,15820

P controller
Kp: 1,06319

PI controller
Kp: 1,126
Ki: 0,514280

PID controller
Kp: 10,007
Ki: 3,322196
Kd: 6,545

Smith
Kp: 1,126
Ki: 0,514

Tuning method: MOMI method - tracking

Closed-loop sampling time
Sampling time [s]: 0,0300
Recommended Ts [PID]: 0,0301

Type of closed-loop response
[✓ Reference tracking]
Disturbance-rejection response

Controller filtering
2nd order filter
0,1506 Time constant

Robustness-speed selection
Robust I Fast

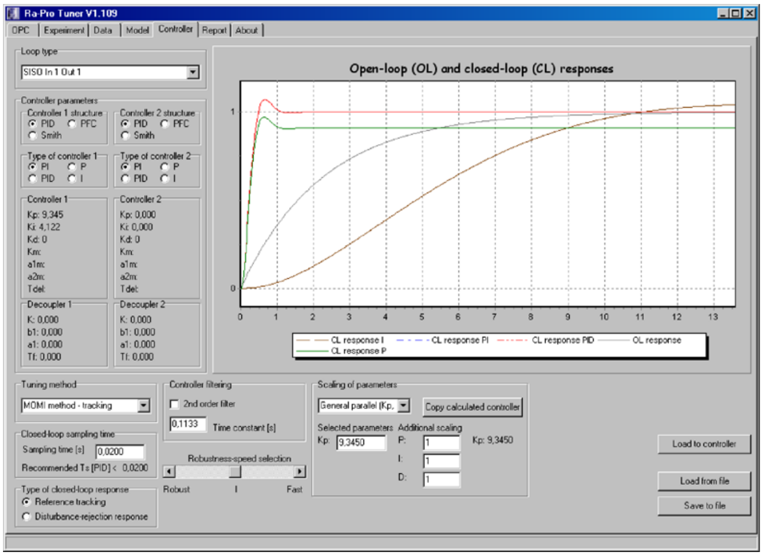
Calculation of scaled parameters
Parallel - general (Kp

Controller selection
P [✓ PI] Smith p.
I PID

Selected parameters Additional scaling
Kp: 1,1262 P: 1 Kp: 1,1262
Ki: 0,5143 I: 1 Ki: 0,51428
D: 1

Cascade control
Cascade Inner-loop proportional gain: 0

Fig. 10.19 Controller parameters user interface for SISO systems



Loop type: SISO In 1 Out 1

Controller parameters

Controller 1 structure: [✓ PID] PFC Smith
Controller 2 structure: [✓ PID] PFC Smith

Type of controller 1: [✓ PI] P PID I
Type of controller 2: [✓ PI] P PID I

Controller 1	Controller 2
Kp: 9,345	Kp: 0,000
Ki: 4,122	Ki: 0,000
Kd: 0	Kd: 0
Km:	Km:
a1m:	a1m:
a2m:	a2m:
Tdel:	Tdel:

Decoupler 1	Decoupler 2
K: 0,000	K: 0,000
b1: 0,000	b1: 0,000
a1: 0,000	a1: 0,000
Tf: 0,000	Tf: 0,000

Tuning method: MOMI method - tracking

Closed-loop sampling time
Sampling time [s] 0,0200
Recommended Ts [PID] < 0,0200

Type of closed-loop response
[✓ Reference tracking]
Disturbance-rejection response

Controller filtering
2nd order filter
0,1133 Time constant [s]

Robustness-speed selection
Robust I Fast

Scaling of parameters
General parallel (Kp,

Selected parameters Additional scaling
Kp: 9,3450 P: 1 Kp: 9,3450
I: 1
D: 1

Fig. 10.20 Controller parameters user interface for TITO systems

Two different controller structures can be selected: a PID controller and a Smith predictor (a PFC controller is under development). All of them use the same PID controller structure, as given by Fig. 10.5 and Eq. (10.4).

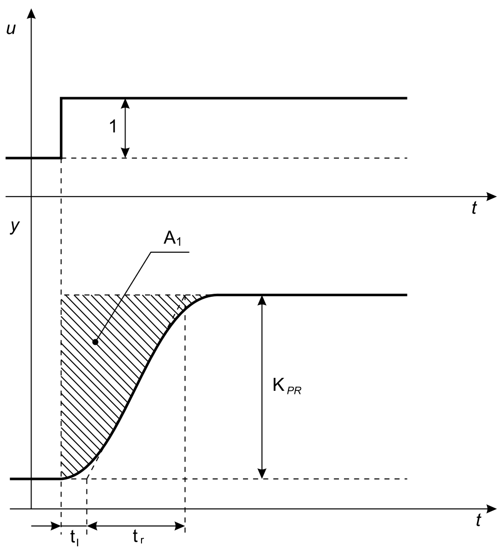


Fig. 10.21 Process gain, lag and rise times

Table 10.1 PID controller parameters for several tuning methods

	K_p	K_i	K_d
ZN tuning method			
P	$\frac{T_r}{T_l K_{PR}}$	–	–
PI	$\frac{0.9T_r}{T_l K_{PR}}$	$\frac{0.3T_r}{T_l^2 K_{PR}}$	–
PID	$\frac{1.2T_r}{T_l K_{PR}}$	$\frac{0.6T_r}{T_l^2 K_{PR}}$	$\frac{0.6T_r}{K_{PR}}$
CHR tuning method for tracking			
PI	$\frac{0.35T_r}{T_l K_{PR}}$	$\frac{0.3}{T_l K_{PR}}$	–
PID	$\frac{0.6T_r}{T_l K_{PR}}$	$\frac{0.6}{T_l K_{PR}}$	$\frac{0.3T_r}{K_{PR}}$
CHR tuning method for disturbance			
PI	$\frac{0.6T_r}{T_l K_{PR}}$	$\frac{0.15T_r}{T_l^2 K_{PR}}$	–
PID	$\frac{0.95T_r}{T_l K_{PR}}$	$\frac{0.4T_r}{T_l^2 K_{PR}}$	$\frac{0.4T_r}{K_{PR}}$
BT tuning method			
PI	$\frac{0.5(1+(\frac{a_1}{a_1+T_{del}})^2)}{K_{PR}}$	$\frac{1}{K_{PR}(a_1+T_{del})}$	–

Several different tuning methods can be selected. Besides the Ziegler–Nichols (ZN) and Chien–Hrones–Reswick (CHR) tuning methods, the Magnitude Optimum Multiple Integration (MOMI) tuning method for reference tracking [27] and disturbance rejection [25, 28] or the Balanced Tuning (BT) method [15] can be chosen.

The ZN and CHR methods are based on measured process gain (K_{PR}), lag time (T_l), and rise time (T_r) (see Fig. 10.21). The mentioned parameters are obtained from the process model by means of simulation. The PID controller tuning parameters, when using ZN and CHR tuning methods (for tracking and disturbance rejection), are given in Table 10.1.

The BT and MOMI tuning methods are based on the calculated process model. The PI controller parameters for the BT method are also given in Table 10.1 (the method is not defined for PID controllers).

The MOMI method is defined for tracking response [27] and for disturbance rejection [25, 28]. The method can be used on process transfer functions of an arbitrary order with a time delay or on non-parametric process data in a time-domain (e.g., process step-response) by multiple integration of the process response. Here we applied the method to the second-order process model (10.7). The PID controller parameters for tracking the response are as follows [27]:

$$\begin{bmatrix} K_i \\ K_p \\ K_d \end{bmatrix} = \begin{bmatrix} -A_1 & A_0 & 0 \\ -A_3 & A_2 & -A_1 \\ -A_5 & A_4 & -A_3 \end{bmatrix}^{-1} \begin{bmatrix} -0.5 \\ 0 \\ 0 \end{bmatrix} \tag{10.9}$$

where the so-called characteristic areas A_i are defined as follows:

$$\begin{aligned} A_0 &= K_{PR} \\ A_1 &= K_{PR}(a_1 + T_{del}) \\ A_2 &= K_{PR}\left(-a_2 + \frac{T_{del}^2}{2!}\right) + A_1 a_1 \\ A_3 &= K_{PR}\left(\frac{T_{del}^3}{3!}\right) + A_2 a_1 - A_1 a_2 \\ A_4 &= K_{PR}\left(\frac{T_{del}^4}{4!}\right) + A_3 a_1 - A_2 a_2 \\ A_5 &= K_{PR}\left(\frac{T_{del}^5}{5!}\right) + A_4 a_1 - A_3 a_2 \end{aligned} \tag{10.10}$$

Note that area A_1 is graphically depicted in Fig. 10.21. The PI or I (integral) controller parameters can be calculated by reducing the matrix and vector dimensions in Eq. (10.9). The PI controller parameters are as follows:

$$\begin{bmatrix} K_i \\ K_p \end{bmatrix} = \begin{bmatrix} -A_1 & A_0 \\ -A_3 & A_2 \end{bmatrix}^{-1} \begin{bmatrix} -0.5 \\ 0 \end{bmatrix} \tag{10.11}$$

The I controller can be simply calculated as

$$K_i = \frac{0.5}{A_1} \tag{10.12}$$

Note that the PI and I controller do not require output filtering in Eq. (10.4). Therefore, $T_F = 0$.

The MOMI tuning method for disturbance rejection [28] defines the PI controller parameters as:

$$
\begin{aligned}
K_P &= \frac{\xi_2 - \mathrm{sgn}(\xi_2) A_1 \sqrt{A_2^2 - A_1 A_3}}{\xi_1} \\
K_i &= \frac{(1 + K_P A_0)^2}{2A_1}
\end{aligned}
\tag{10.13}
$$

where

$$
\begin{aligned}
\xi_1 &= A_0^2 A_3 - 2A_0 A_1 A_2 + A_1^3 \\
\xi_2 &= A_1 A_2 - A_0 A_3
\end{aligned}
\tag{10.14}
$$

The parameters A_0 to A_3 depend on the process parameters, as given in (10.10). The PID controller parameters for disturbance rejection are calculated according to the following expression [25]:

$$
\begin{aligned}
K_P &= \frac{A_0 A_3 - A_1 A_2 + K_d (A_0^2 A_2 - A_0 A_1^2) + \sqrt{\alpha}}{2A_0 A_1 A_2 - A_1^3 - A_0^2 A_3} \\
K_i &= \frac{(1 + K_P A_0)^2}{2(A_1 + A_0^2 K_d)} \\
\alpha &= \left[A_2^2 - A_1 A_3 + K_d \left(2A_0 A_1 A_2 - A_0^2 A_3 - A_1^3\right)\right]\left(A_1 + A_0^2 K_d\right)^2
\end{aligned}
\tag{10.15}
$$

where derivative gain K_d is calculated from Eq. (10.9).

The user can modify the closed-loop sampling time and immediately observe a change in the predicted closed-loop response. There is a slider for choosing the trade-off between the robustness and the speed of the closed-loop response and buttons for selection of the reference following or disturbance rejection closed-loop response.

If a Smith predictor is chosen instead of an ordinary PI(D) controller, the Smith predictor's PI controller parameters are calculated according to the process model (10.7) without a pure time delay ($T_{del} = 0$).

Calculation of the controller and decoupler parameters for TITO systems is more complex. In principle, the calculation is based on Eqs. (10.2), (10.3), and (10.9). However, the actual calculation of the decoupler and controller parameters is, due to its simplicity, based on the MOMI tuning method. More details about the tuning procedure can be found in [26].

10.3.6 Automatic Report Generation

A report in the form of a Microsoft Word® file can be generated by clicking the tab "Report" and choosing the appropriate parameters. The report can contain mea-

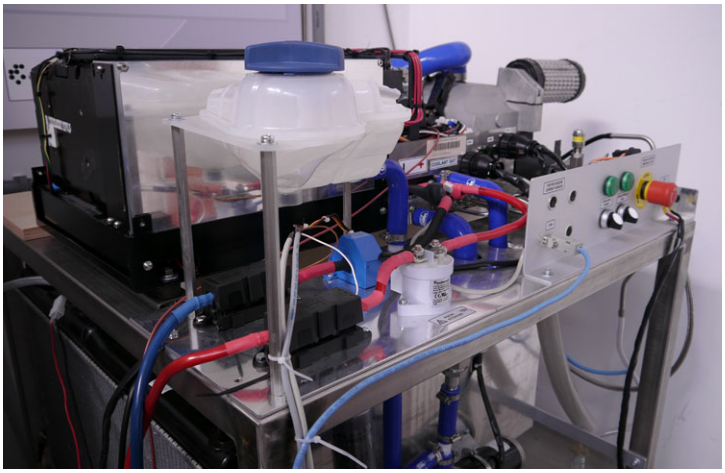


Fig. 10.22 Photo of the fuel cell made by Hydrogenics

surement data (in graphical form), the process model, and the calculated controller parameters with predicted closed-loop responses.

10.4 Experimental Testing

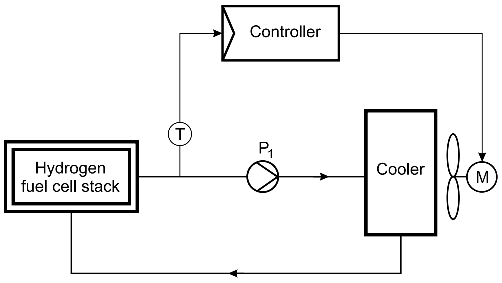
The RaPro environment thus far has been tested on several laboratory and industrial plants. Among others, it has been used in the pharmaceutical industry (HVAC control), the production of TiO_2 (temperature and level control), the production of plastics (temperature and pressure control), and plasma coating processes (pressure control).

In order to illustrate the features and characteristics of the environment, its use in two examples will be presented. The first one is temperature control in a hydrogen fuel cell power module, which is a single input/single output system, and the second one is level control in a laboratory setup with three water tanks, which is a two input/two output multivariable system.

10.4.1 Temperature Control in a Hydrogen Fuel Cell Power Module

The system under consideration is a HyPM HD8 8 kW hydrogen fuel cell power module manufactured by Hydrogenics (see Fig. 10.22). The fuel cell produces electricity and heat from hydrogen gas and oxygen from the air. In order to maintain the normal working temperature of the hydrogen cell, the excess heat should be removed from the stack of cells (the upper-left part of the picture) by means of

Fig. 10.23 Fuel cell stack temperature control loop



a motorised fan connected to a radiator (the bottom-left side of the picture). The reference temperatures should be kept tightly under control. Otherwise, the cell is automatically switched off.

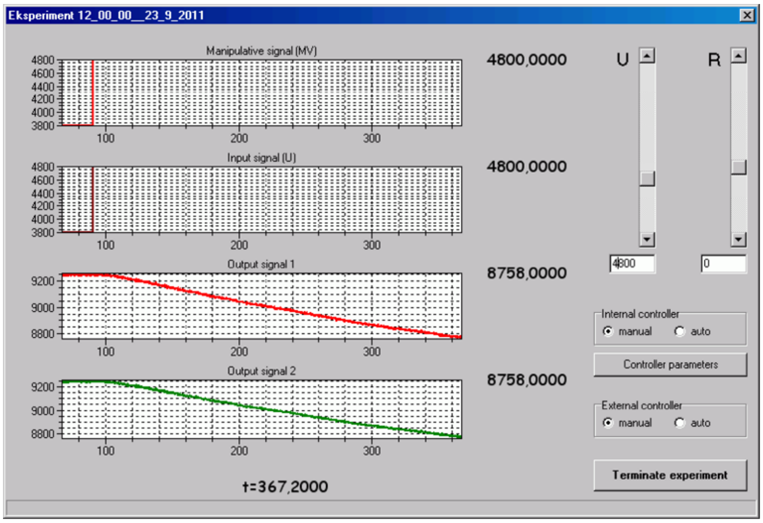
The existing temperature controller has a relatively large gain, causing large changes in the fan motor speed, similar to an on/off controller. Our goal was to determine if the existing control loop can be made more stable.

A simplified scheme of the temperature control loop is given in Fig. 10.23. The coolant temperature (T) is controlled by the speed of the fan motor M. The process input is the voltage on the motor M (the voltage and current boost are externally provided by a 3 kW power supply, EA-PSI 8080-120 from Elektro Automatik GmbH) and the process output is the stack coolant temperature (converted into voltage by the Pt100 temperature sensor). Voltage signals are measured and provided by a Mitsubishi Melsec Q62P PLC controller with A/D and D/A modules. The D/A module has an output span from 0 to 16,000 digits, which corresponds to fan motor voltages from 0 to 10.11 V. The A/D module has input span values from 3200 (live zero) to 16,000 digits, which corresponds to stack temperatures from 0 °C to 100 °C. A Beijer Electronics OPC Server was running on a personal computer.

After connecting the OPC server, which defined the process input and output OPC signals and sampling time $T_s = 0.2$ s, the manual experiment was initiated. The fuel cell was loaded by a 20 A current drawn by a PLW9K-120-1000E load from Amrel and the process input value was set to a value of 2.4 V (3800 digits). After the stack reached a steady-state temperature of approximately 47.2 °C (9240 digits), the process input changed to a value of 3.03 V (4800 digits). The experiment window during execution of the experiment is shown in Fig. 10.24. The stack temperature changed to about 40.5 °C (8380 digits). The process input and output signals during the open-loop experiment are shown in Fig. 10.25.

When the experiment is terminated, the signals are shown in a data management window (Fig. 10.26). Note that both process outputs are the same, since the process is single-input/single-output. Since the measurements are relatively smooth, additional filtering was not applied.

Now we can define the start and end of the measurements which will be taken into consideration for calculating the process model. The start of the measurement is set just before the first change in the process input (the fan motor voltage) signal and the end of the measurement is set when the temperature change becomes almost



4800,0000

4800,0000

8758,0000

8758,0000

t=367,2000

U: 4800

R: 0

Internal controller: [✓ manual] auto

External controller: [✓ manual] auto

Fig. 10.24 The experiment window during execution of the experiment on the fuel cell

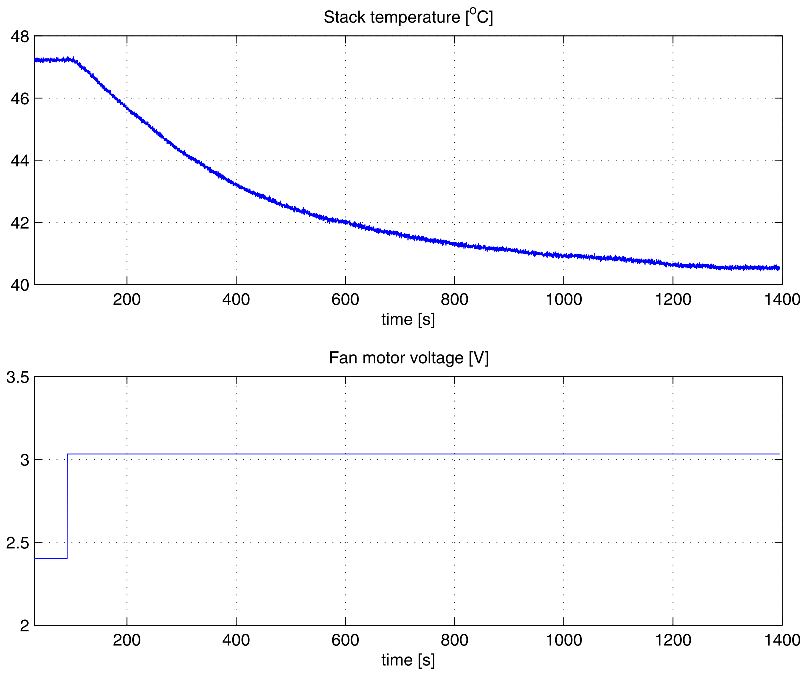
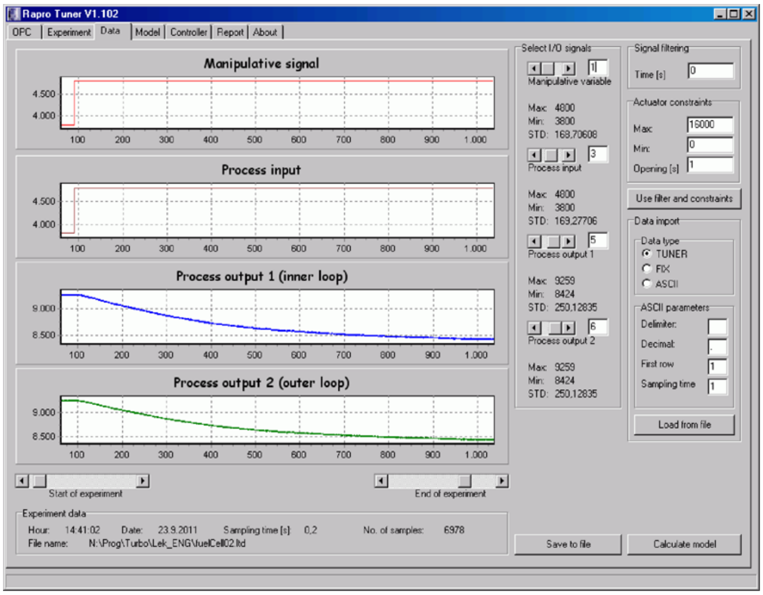


Fig. 10.25 The process input and output signals during the open-loop experiment on the fuel cell


Rapro Tuner V1.102
OPC | Experiment | Data | Model | Controller | Report | About
Manipulative signal
Process input
Process output 1 (inner loop)
Process output 2 (outer loop)
Start of experiment
End of experiment
Save to file
Calculate model
Use filter and constraints
Load from file

Select I/O signals
Manipulative variable: 1
Max: 4800 Min: 3800 STD: 168.70608
Process input: 3
Max: 4800 Min: 3800 STD: 169.27706
Process output 1: 5
Max: 9259 Min: 8424 STD: 250.12835
Process output 2: 6
Max: 9259 Min: 8424 STD: 250.12835

Signal filtering
Time [s]: 0

Actuator constraints
Max: 16000
Min: 0
Opening [s]: 1

Data import
Data type: [✓ TUNER] FIX ASCII

ASCII parameters
Delimiter:
Decimal: .
First row: 1
Sampling time: 1

Experiment data
Hour: 14:41:02 Date: 23.9.2011 Sampling time [s]: 0,2 No. of samples: 6978
File name: N:\Prog\Turbo\Lek_ENG\fuelCell02.txt

Fig. 10.26 Collected data during an open-loop experiment on the fuel cell (SISO system)

constant. The calculated process model is the following:

$$G_P(s) = \frac{-0.8503}{1 + 333.7s + 8356s^2} \tag{10.16}$$

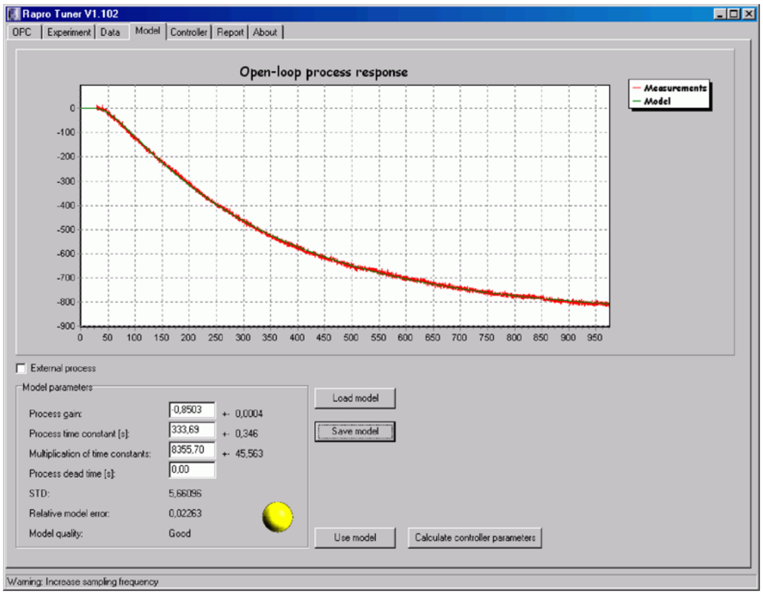
Note that the process model has been obtained from the actual A/D and D/A signals (Fig. 10.26) instead of the physical values (Fig. 10.25).

A comparison of the model response and the process measurement is depicted in Fig. 10.27.

It can be seen that the model quality is good (yellow dot), and the calculated model fits the actual measurements tightly (the relative model error is only 2.3 %), so we can proceed to the next stage: calculation of the controller parameters.

The MOMI tuning method for tracking was chosen for the calculation of the controller parameters since the actual stack temperature should follow the internal temperature reference profile. The chosen controller closed-loop sampling time was the same as in the open-loop experiment (0.2 s) and the PID controller filter time constant was 5 s. The calculated PI and PID controller parameters are given in Table 10.2 and Fig. 10.28. Note that the PI and the Smith predictor parameters were calculated for filter time constant $T_f = 0$ s and are therefore not equal to the calculated parameters in Fig. 10.28 (which were calculated for filter time constant $T_f = 5$ s).

The calculated Smith predictor and the PI controller parameters are the same. The reason is that the process model does not contain a pure time delay. Therefore, the Smith predictor cannot improve the closed-loop response and will not be tested herein.



☐ External process

Model parameters

Process gain:	-0,8503	+- 0,0004
Process time constant [s]:	333,69	+- 0,346
Multiplication of time constants:	8355,70	+- 45,563
Process dead time [s]:	0,00	
STD:	5,66096	
Relative model error:	0,02263	
Model quality:	Good	

Warning: Increase sampling frequency

Fig. 10.27 The process model and the actual measurement of the fuel cell temperature

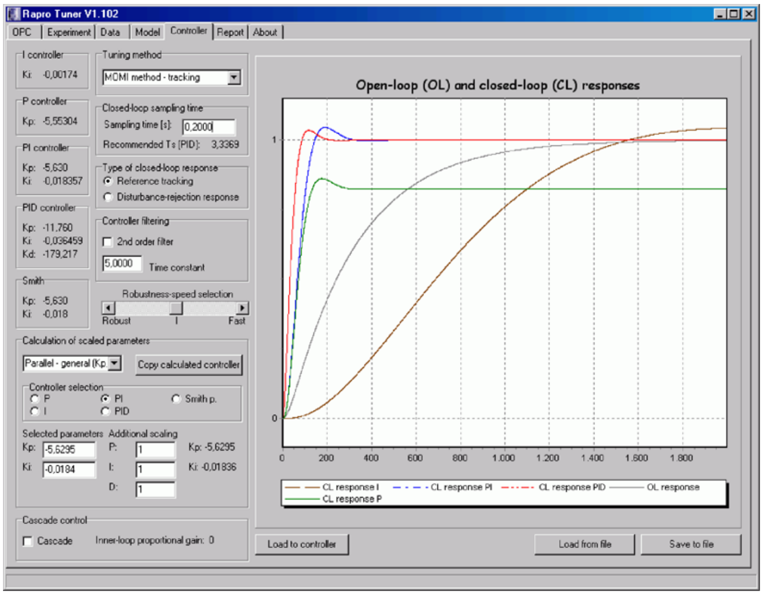
Table 10.2 PI, PID, and Smith predictor controller parameters for the temperature loop of the fuel cell

	K_p	K_i	K_d
PI	−6.66	−0.0217	–
PID	−11.76	−0.0365	−179.2
Smith	−6.66	−0.0217	–

The predicted closed-loop responses are given in Fig. 10.28. It can be seen that the anticipated closed-loop responses are relatively fast and stable. Therefore, we decided to test the calculated controllers by using the embedded controllers.

To do this, we started a new experiment by clicking “Activate experiment” under the “Experiment” Tab. The experimentation window was the same as before (Fig. 10.25). Before switching from manual to automatic mode in “Internal controller”, the appropriate reference value should be set for the embedded controller (e.g., by changing the R slider position). The controller parameters can be verified by clicking the button “Controller parameters”. A new window then appears (Fig. 10.29) which displays the chosen controller parameters. The controller parameters can be imported by clicking on the button “Insert given and calculated parameters” and copied to the embedded controller by clicking the button “Use controller parameters”.

When changing to automatic mode for the internal (embedded) controller (Fig. 10.30), we actually started the closed-loop control experiment with the calculated controller. By changing slider R (reference), the temperature set-point can



I controller
Ki: -0,00174

P controller
Kp: -5,55304

PI controller
Kp: -5,630
Ki: -0,018357

PID controller
Kp: -11,760
Ki: -0,036459
Kd: -179,217

Smith
Kp: -5,630
Ki: -0,018

Tuning method: MOMI method - tracking

Closed-loop sampling time
Sampling time [s]: 0,2000
Recommended Ts [PID]: 3,3369

Type of closed-loop response
[✓ Reference tracking]
Disturbance-rejection response

Controller filtering
☐ 2nd order filter
5,0000 Time constant

Robustness-speed selection
Robust — Fast

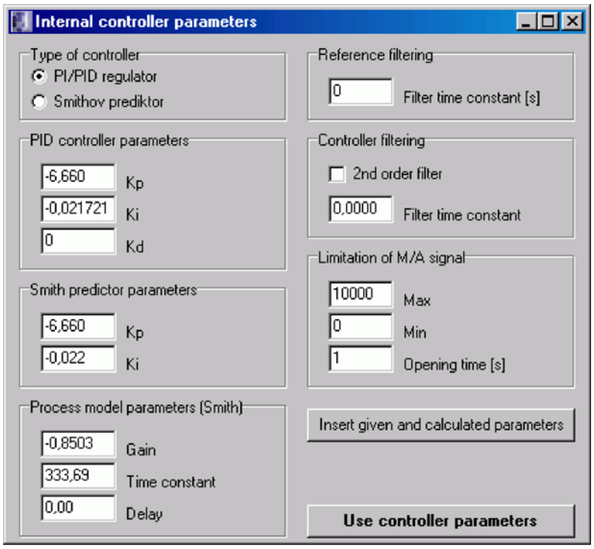
Calculation of scaled parameters
Parallel - general (Kp

Controller selection
P [✓ PI] Smith p.
I PID

Selected parameters		Additional scaling		
Kp:	-5,6295	P:	1	Kp: -5,6295
Ki:	-0,0184	I:	1	Ki: -0,01836
		D:	1	

Cascade control
☐ Cascade Inner-loop proportional gain: 0

Fig. 10.28 The predicted closed-loop responses for the fuel cell temperature control


Internal controller parameters
Insert given and calculated parameters
Use controller parameters

Type of controller
[✓ PI/PID regulator]
Smithov prediktor

PID controller parameters
-6,660 Kp
-0,021721 Ki
0 Kd

Smith predictor parameters
-6,660 Kp
-0,022 Ki

Process model parameters (Smith)
-0,8503 Gain
333,69 Time constant
0,00 Delay

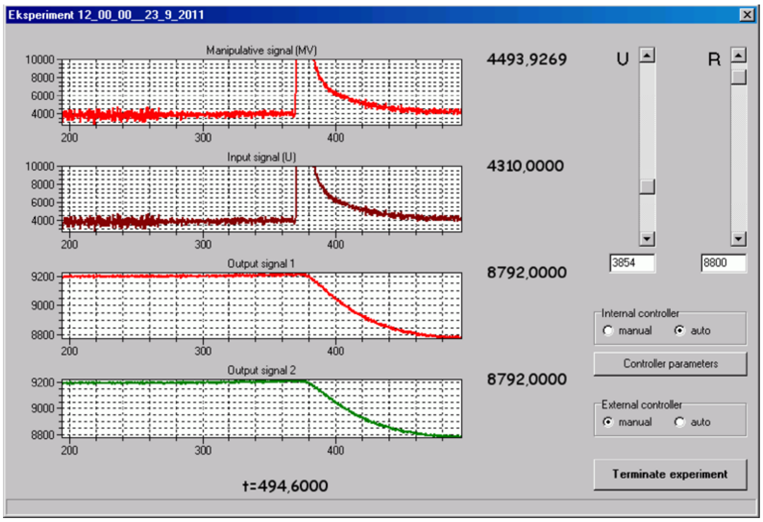
Reference filtering
0 Filter time constant [s]

Controller filtering
☐ 2nd order filter
0,0000 Filter time constant

Limitation of M/A signal
10000 Max
0 Min
1 Opening time [s]

Fig. 10.29 Parameters of the embedded PI controller for the fuel cell temperature control

be modified. The temperature set-point has been changed from 40.6 °C (8400 digits) to 46.9 °C (9200 digits) by using the calculated PI controller, and from 46.9 °C (9200 digits) to 43.8 °C (8800 digits) by using the calculated PID controller. The results are shown in Figs. 10.31 and 10.32. The resulting closed-loop responses are relatively fast and stable. The PID controller resulted in a faster closed-loop



Eksperiment 12_00_00__23_9_2011

4493,9269

4310,0000

8792,0000

8792,0000

t=494,6000

U: 3854

R: 8800

Internal controller: manual [✓ auto]

External controller: [✓ manual] auto

Fig. 10.30 Experiment window during the closed-loop control of the fuel cell temperature with embedded PID controller

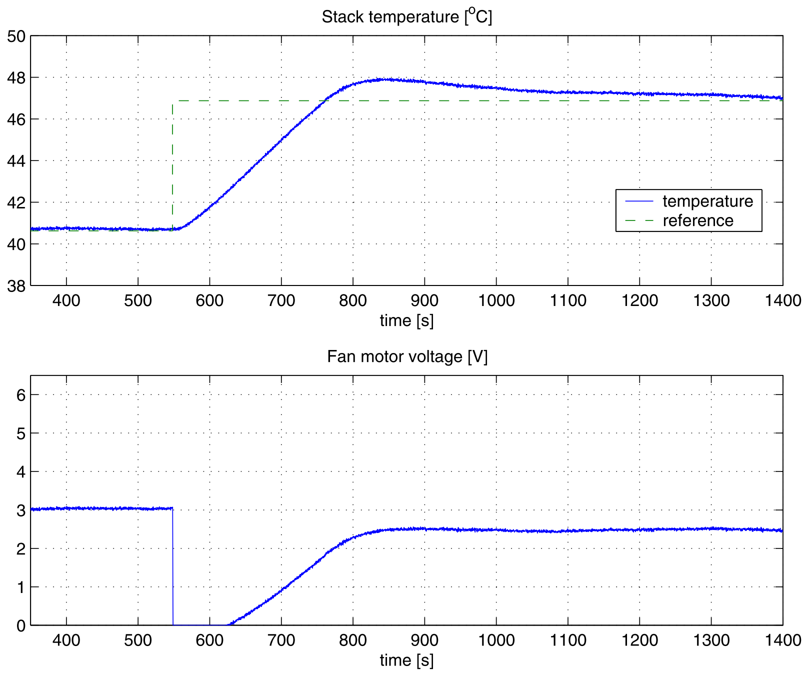


Fig. 10.31 The temperature of the fuel-cell stack under closed-loop control using the calculated PI controller

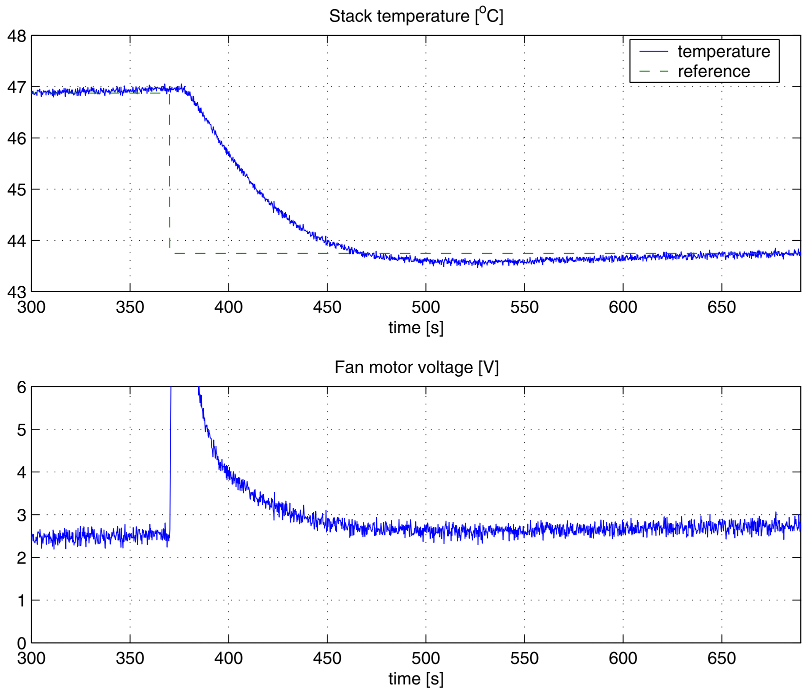


Fig. 10.32 The temperature of the fuel-cell stack under closed-loop control using the calculated PID controller

response with slightly smaller overshoot when compared to the PI controller, all according to the responses predicted in Fig. 10.28. The closed-loop response, when using both controllers, is significantly improved when compared to the response of the factory built-in controller.

The disturbance rejection properties of the PID controller were tested by changing the load from 20 A to 30 A at approximately $t = 700$ s. The closed-loop responses are shown in Fig. 10.33. It can be seen that the controller response is relatively fast and the highest error is about 0.6 °C. However, the approach to the reference is relatively slow. Namely, the PID controller parameters were tuned for tracking, since tracking the response of the internal stack temperature reference is more important than disturbance rejection properties.

10.4.2 Level Control in a Three-Water-Tank System

The second experiment was performed on the three-water-tank system shown in Fig. 10.34 [14]. The system was chosen in order to test more complex features of the environment, i.e., control of TITO multivariable systems.

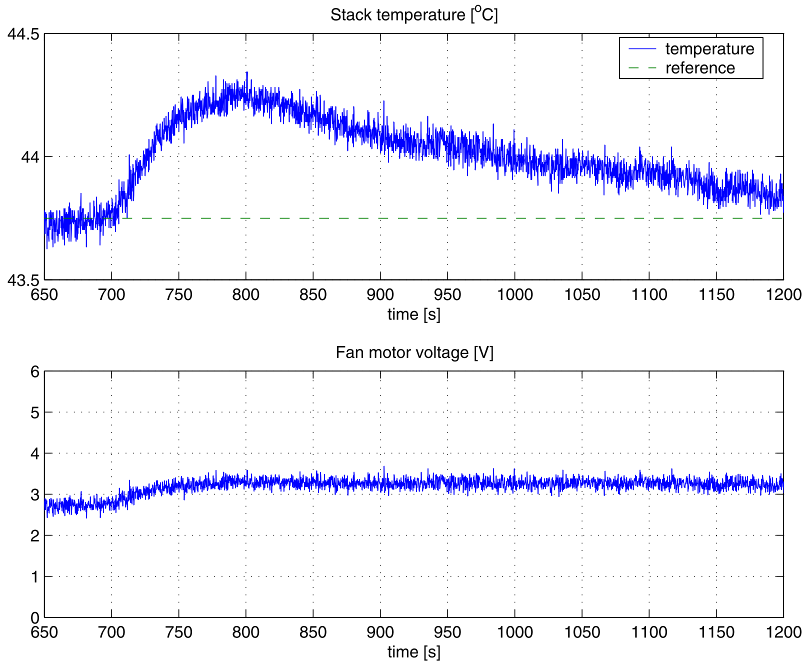


Fig. 10.33 The temperature of the fuel-cell stack under closed-loop control using the calculated PID controller with changed load

The three-water-tank setup consists of two water pumps, a reservoir and three water tanks. The water tanks can be connected by means of electric valves. In our setup, two water tanks were used (R_1 and R_2), as depicted in the block diagram shown in Fig. 10.35.

The selected multivariable system consists of reservoir R_0, pumps P_1 and P_2, electric (on/off) valves V_1 (open) and V_2 (open), and water tanks R_1, R_2, and R$_3$. Valve V_3 is closed. The process inputs are the voltage on pumps P_1 and P_2 and the process outputs are the water levels in the first (h_1) and the third tank (h_3), measured by the pressure to the voltage transducer. Similarly as in the previous case, the voltage signals are measured and provided by a Mitsubishi Melsec Q62P PLC controller with A/D and D/A modules. The D/A module has an output span from 0 to 4000 digits, which corresponds to voltages on the pumps from 0 to 10 V. The A/D module, which measures voltages from the pressure to the voltage transducers, has input span values from 0 to 4000 digits, which corresponds to 10 V. A Beijer Electronics OPC Server was running on a personal computer.

After connecting the OPC server and defining the process input and output OPC signals and sampling time $T_s = 0.5$ s, the manual experiment was initiated, where voltages on both pumps were modified by introducing step-changes between 800 digits (2 V) and 1100 digits (2.75 V) at a particular process input. The experi-

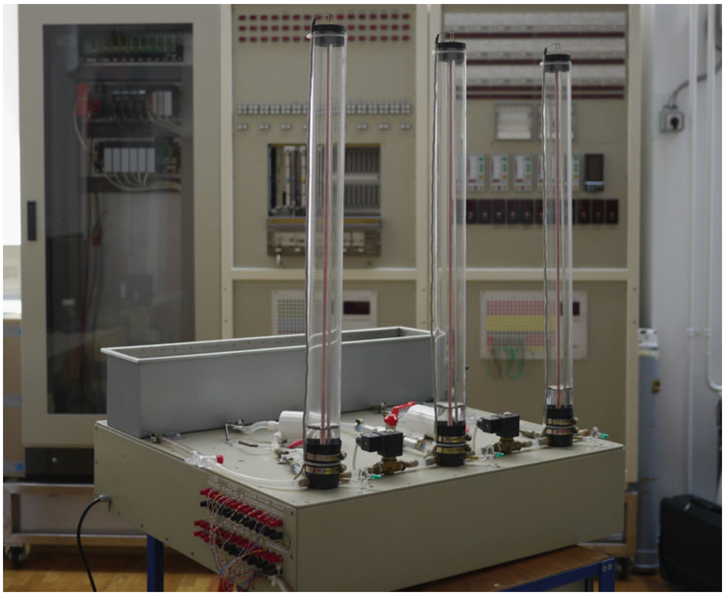


Fig. 10.34 Picture of the laboratory hydraulic setup

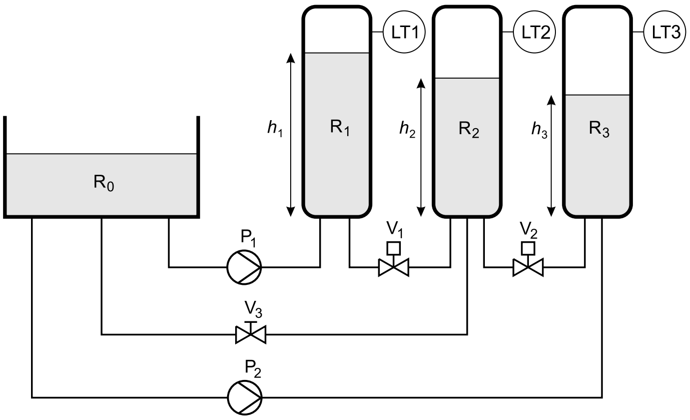


Fig. 10.35 Block diagram of the laboratory hydraulic setup

ment window and voltages on the pumps and water levels h_1 and h_3 are shown in Figs. 10.36 and 10.37.

From Fig. 10.37 it can be seen that the process output has significant fluctuations in the steady-state. The process response from $t = 900$ s to $t = 1400$ s has been chosen for the process identification, since it shows the least fluctuation in the steady-state.

A detailed process response is shown in Fig. 10.38.

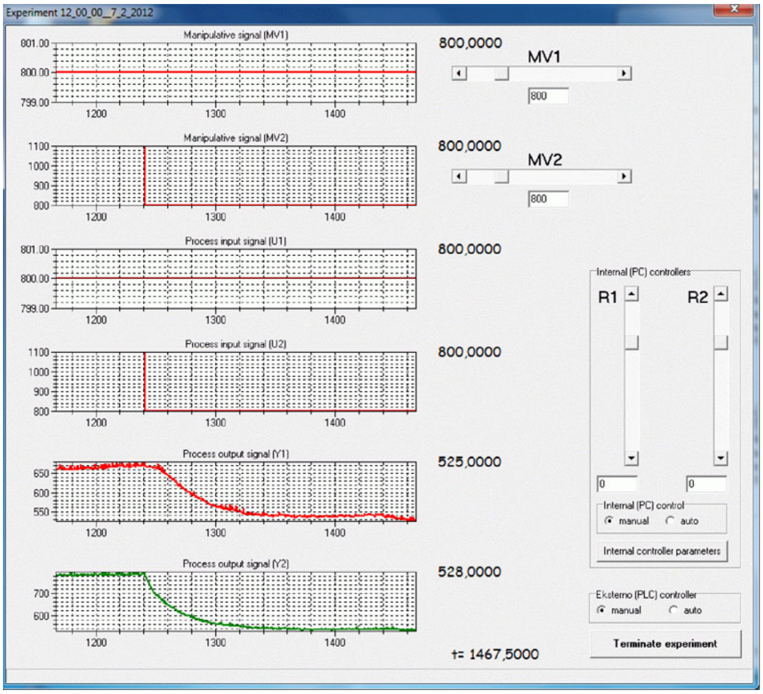

Experiment 12_00_00_7_2_2012
Manipulative signal (MV1)
Manipulative signal (MV2)
Process input signal (U1)
Process input signal (U2)
Process output signal (Y1)
Process output signal (Y2)
800,0000
MV1
800
800,0000
MV2
800
800,0000
800,0000
525,0000
528,0000
t= 1467,5000
Internal (PC) controllers
R1
R2
0
0
Internal (PC) control
manual
auto
Internal controller parameters
Eksterno (PLC) controller
manual
auto
Terminate experiment

Fig. 10.36 The process signals in the experiment window during the open-loop experiment

The start of the experiment considered for the calculation of the model is set just before the first changes in the process input signals and the end of the experiment is set when the liquid level almost settles. The calculated transfer functions of the multivariable model are the following:

$$\begin{aligned} g_{P11}(s) &= \frac{0.722}{1+19.46s} \\ g_{P12}(s) &= \frac{0.448e^{-1.5s}}{1+39.2s+382s^2} \\ g_{P21}(s) &= \frac{0.69e^{-2s}}{1+77.79s+29s^2} \\ g_{P22}(s) &= \frac{0.79}{1+19.9s} \end{aligned} \tag{10.17}$$

Note that the process model has been obtained from the actual A/D and D/A signals (digits) instead of the physical values (voltages).

A comparison of the model response and the process measurement for the first process input (the voltage on pump 1) and the second process output (the liquid level h_3) is depicted in Fig. 10.39.

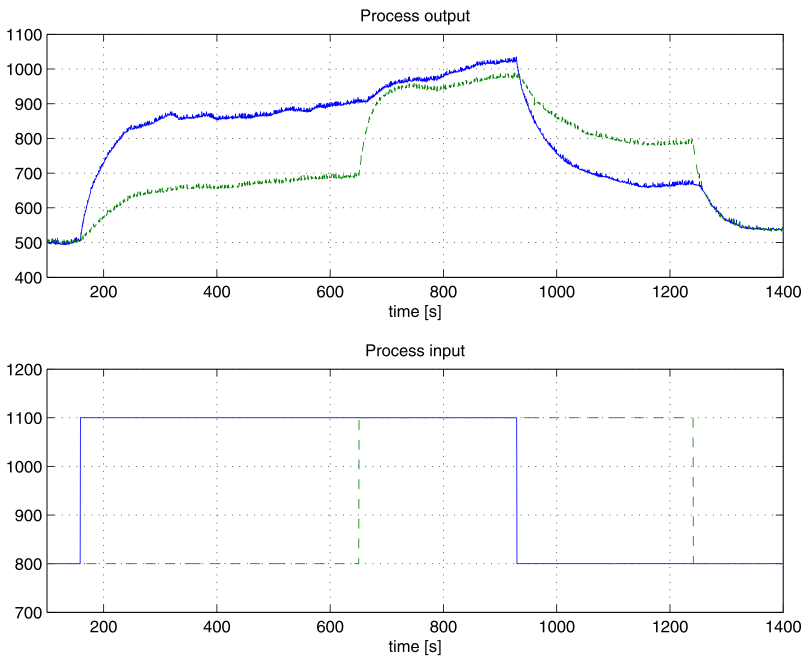


Fig. 10.37 The process input and output signals (in digits) during the open-loop experiment on the three-water-tank setup. The *solid lines* represent the first and the *broken lines* the second input and output

The calculated models approximately fit the actual measurements (note that the model quality in the window is assessed as "fair" and the relative model error is about 6.7 %), so we started the calculation of the controller parameters. The parameters are calculated in accordance with the MOMI tuning method for multivariable processes [23].

The calculated decouplers and the PI controllers parameters were

$$
\begin{aligned}
d_1(s) &= \frac{0.873(1+20.1s)}{1+80s+199.4s^2} \\
d_2(s) &= \frac{0.62(1-1.39s)}{1+19.85s} \\
c_1(s) &= \frac{6.92s+0.39}{s} \\
c_2(s) &= \frac{6.33s+0.35}{s}
\end{aligned}
\tag{10.18}
$$

Since the embedded TITO controller is not yet incorporated into the RaPro environment, the experiment on the process, with the calculated decouplers and con-

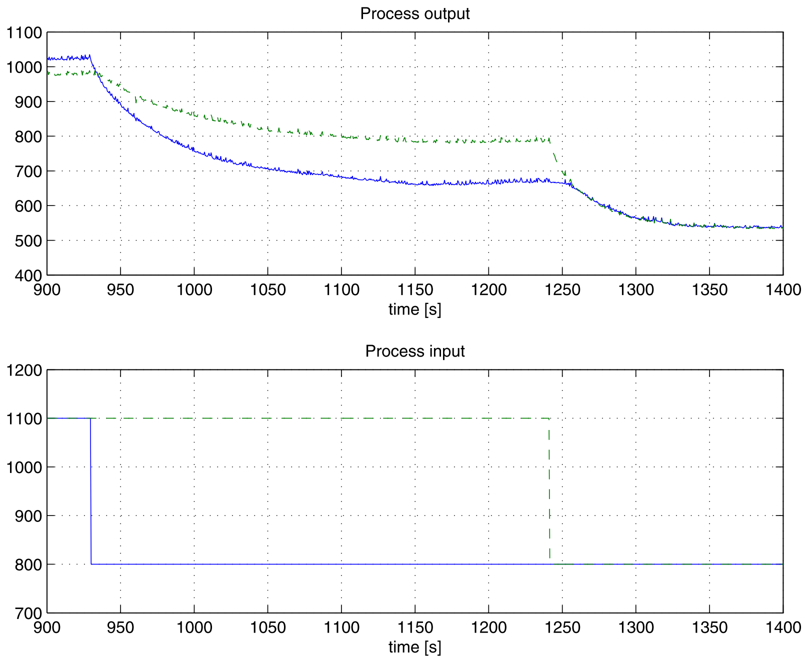
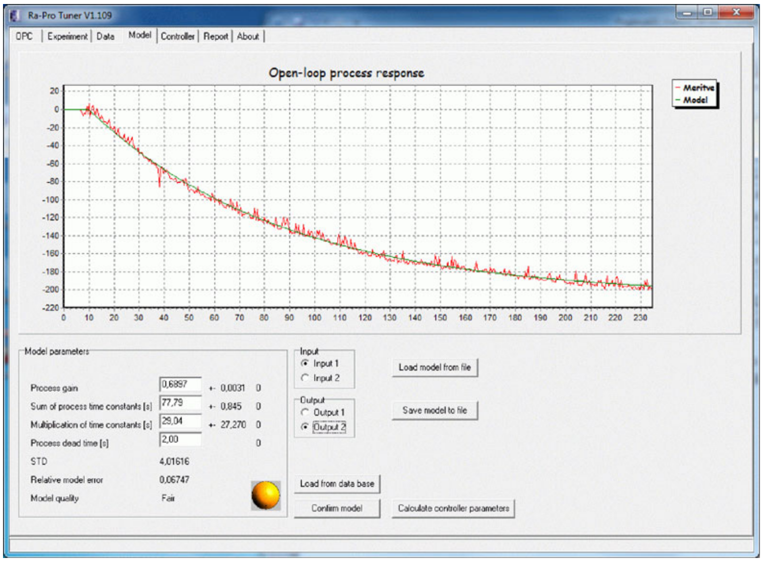


Fig. 10.38 A detailed view of the chosen process input and output signals (in digits)



Model parameters

Process gain	0,6897	+- 0,0031	0
Sum of process time constants [s]	77,79	+- 0,845	0
Multiplication of time constants [s]	29,04	+- 27,270	0
Process dead time [s]	2,00		0
STD	4,01616		
Relative model error	0,06747		
Model quality	Fair		

Input: [✓ Input 1] Input 2

Output: Output 1 [✓ Output 2]

Fig. 10.39 The process model and the actual measurement for the first process input and the second process output

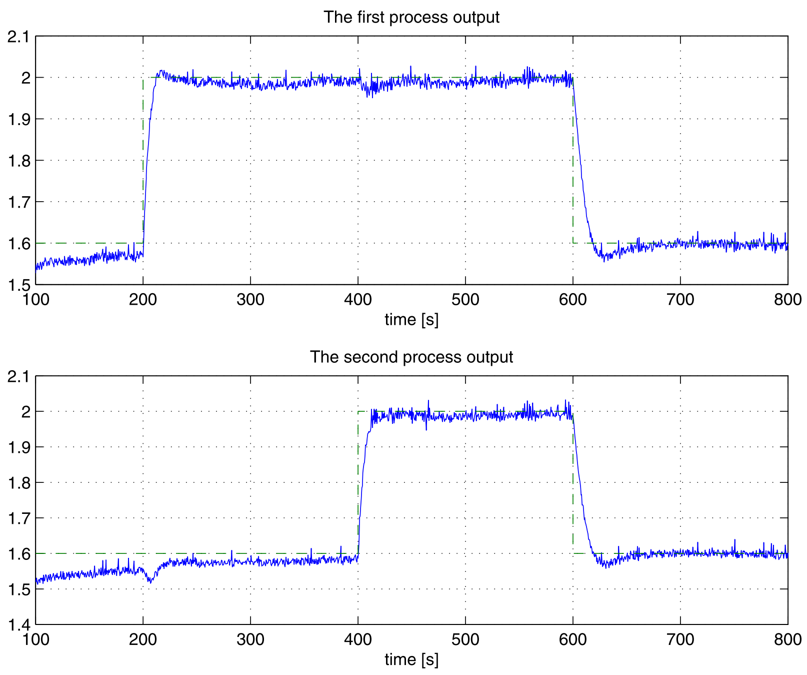


Fig. 10.40 The process outputs under closed-loop control using the calculated decouplers and PI controllers

trollers, was performed in the MATLAB/Simulink program environment with a National Instruments A/D and a D/A converter (NI USB 6215).

The liquid level set-points were changed between 1.6 V and 2 V. The closed-loop responses of the process outputs (liquid levels h_1 and h_3) are shown in Fig. 10.40, while the process inputs (the voltages on pumps V_1 and V_2) are shown in Fig. 10.41. We can see that the closed-loop responses are relatively fast (compared to the open-loop responses) and stable. It can also be observed that the cross-coupling between both channels is relatively small.

10.5 Discussion in the Context of Theory/Practice Issues

The aim of the project was to develop a tool which would simplify the work of control engineers in solving industrial control problems and to bring some more advanced methods nearer to system integrators and maintenance engineers. Although the tool is still under development, the experience gained so far has shown that it really helps in making things quicker and better. However, during development also many problems were encountered, and we had to surmount several obstacles to make the environment really useful.

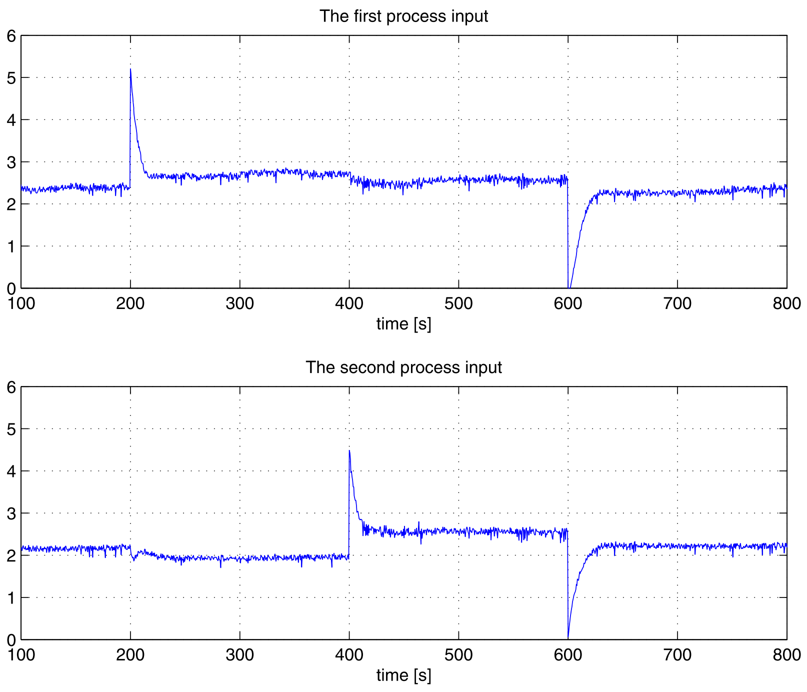


Fig. 10.41 The process inputs under closed-loop control using the calculated decouplers and PI controllers

Reliable process modelling is one of the most important stages, since controller tuning and closed-loop performance depend on the process model. One of the first challenges was noisy processes. It is important to stress once again that the use of the ordinary least squares procedure for process identification is not enough. Namely, it often happens that the obtained process model is far from ideal or is biased if the process noise is substantial and/or the number of samples is low. Our experience confirms that the identification method with instrumental variables significantly improves the quality of the process model and should thus be used as a standard approach in noisy environments.

It is frequently not possible to obtain a reliable process model even though the process is relatively linear and noise-free. Such anomalies often happen with processes containing motor-driven valves or flaps. In this case, the actual process input (e.g., liquid/air flow) is different from the controller output for some amount of time. Since process identification in our tool was based on the controller output and process output signals, the calculated process transfer function was inaccurate. Therefore, data fields for the actuator's hard limitations (the maximum and minimum values) and velocity limitations (the actuator's speed) were added for more reliable estimation of the actual process input signal (instead of the controller output signal). The process model quality was therefore substantially improved.

The first versions of the RaPro environment used very small fixed sampling times for calculating the anticipated (predicted) closed-loop responses. However, when applying the calculated controller parameters in practice, the actual closed-loop responses were sometimes considerably different from the predicted ones (usually with much larger overshoots and with an oscillatory response). It was determined that in most cases the main reason for degraded closed-loop response was the much larger sampling time of the external controller. Therefore, we added an option to modify the controller sampling time of the internal (RaPro) controller and instantly observe the difference in the predicted closed-loop response. Additionally, the recommended sampling time for the PID controller is calculated and shown as well.

While testing the RaPro environment some users wanted a slower response than that predicted by the calculated controller. Therefore, we added a slider for the selection of a slower or faster (usually accompanied by a larger overshoot) closed-loop response.

The number of controller input and output signals is usually high in industrial plants. Therefore, it is very important to have OPC signals for control loops and processes hierarchically organised and well documented. Namely, users with well-organised OPC signals were much faster in finding the required control signals and therefore faster in commissioning the loop.

In practice, we often found some control-loops which perform poorly due to inappropriate design of the controller, ill-chosen size of the actuators, high non-linearity of the process, stiction or hysteresis of the valves or due to disturbances influenced by the neighbouring control loops. The RaPro environment can identify some of these problems when selecting an automatic experiment with several input steps in both directions. However, the mentioned problems *cannot be solved* by the environment. The only cure is better control loop design and careful selection of the actuators.

Since the RaPro environment is aimed at tuning more generalised control structures, it lacks a few specific controller structures which are used by some vendors. Moreover, due to its generality, the number of parameters which should be set/selected is somehow larger when compared to some vendor-specific controller tuning packages. However, during the development of the environment, we tried to find a sweet spot between complexity and ease-of-use. The number of feedback structures, controller types, and functions performed by the environment is therefore chosen so as to cover a broader range of control solutions, on one hand, and to remain efficient, on the other.

10.6 Conclusion

The Rapid Prototyping (RaPro) environment for control systems implementation has been presented. The main functions of the environment are setting the parameters of and executing the experiment, data treatment, process modelling, controller tuning, and automatic report generation. The current version of the environment enables tuning and testing of PID types of controllers, including cascade control loops, and Smith predictor controllers for SISO and TITO processes. A predictive functional controller is the next type of controller to be implemented, but such is still under development.

The RaPro environment has been tested on several laboratory and industrial processes. The results of experiments showed that it enables design and experimental evaluation of control loops in an easy and relatively fast way and thus contributes to more efficient implementation of control systems. Note that a simpler version of the developed prototyping environment is available for free on the Internet [22, 24].

A disadvantage of the RaPro environment, especially for TITO systems, is its relatively complex user interface, which is sometimes less intuitive. Moreover, the environment calculates parameters for generic PID controller structures and does not take into account some specific controller structures from different vendors. Therefore, in some cases, users are forced to re-calculate controller parameters for some specific controller structures.

Acknowledgements The author would like to thank to LEK, d.d., the Slovenian Research Agency under grant P2-0001, and the Ministry of Education, Science, Culture and Sport of the Republic of Slovenia for their financial support. The financial support of the European Regional Development Fund in the framework of the Centre of Excellence for Advanced Control Technologies and the Competence Centre for Advanced Control Technologies of the European Union is also gratefully acknowledged.

References

1. Bestune (2011) PID controller auto-tuning software. http://bestune.50megs.com
2. Bucher R, Balemi S (2006) Rapid controller prototyping with MATLAB&Simulink and Linux. Control Eng Pract 14:185–192
3. Butts K, Varga A (2011) Tools and platforms for control systems. In: Samad T, Annaswamy A (eds) The impact of control technology. IEEE Control Systems Society, New York, pp 89–93
4. Control Station, Inc. (2012) Loop-pro product suite. http://www.controlstation.com/page/53-pid-controller-tuning
5. dSPACE (2012) Test and experiment software. http://www.dspace.com/en/ltd/home/products/sw/expsoft.cfm
6. ExperTune (2011) PIDLOOP optimizer. http://www.expertune.com/standard.html
7. Garnier H, Wang L (2008) Identification of continuous-time models from sampled data. Springer, London
8. Gerkšič S, Strmčnik S, Van den Boom TJJ (2008) Feedback action in predictive control: an experimental case study. Control Eng Pract 16:321–332
9. Gonzalez-Martin R, Lopez I, Morilla F, Pastor R (2003) Sintolab: the REPSOL-YPF PID tuning tool. Control Eng Pract 11:1469–1480
10. Hercog D, Jezernik K (2005) Rapid control prototyping using MATLAB/Simulink and a DSP-based motor controller. Int J Eng 21(4):596–605
11. Intune (2011) Loop tuning, performance monitoring & diagnostic software. http://www.controlsoftinc.com/intune5.shtml
12. IPCOS NV (2012) INCA PID tuner. http://www.ipcos.com/en/inca_pid_tuner
13. Isermann R (2008) Mechatronic systems—innovative products with embedded control. Control Eng Pract 16:14–29

14. Juričić Đ, Dolanc G, Rakar A (1997) Semi-physical modelling of a FDI benchmark process. In: Troch I, Breitenecker F (eds) Proceedings ARGESIM 1997, Vienna, Austria, pp 815–820
15. Klán P, Gorez R (2000) Balanced tuning of PI controllers. Eur J Control 6(6):541–550
16. Mathworks (2011) MATLAB OPC toolbox. http://www.mathworks.com/products/opc
17. MATLAB Control Systems Toolbox (2012) Function "pidtool". http://www.mathworks.com/help/toolbox/control/ref/pidtool.html
18. Pannocchia G, Laachi N, Rawlings JB (2006) A candidate to replace PID control: siso-constrained LQ control. AIChE J 51:1178–1189
19. Qin SJ, Badgwell TA (2003) A survey of industrial model predictive control technology. Control Eng Pract 11(7):733–764
20. Stewart G, Samad T (2011) Cross-application perspectives: application and market requirements. In: Samad T, Annaswamy A (eds) The impact of control technology. IEEE Control Systems Society, New York, pp 95–100
21. U-Tune PID Tuning Package. http://www.contek-systems.co.uk/UTune.htm
22. Vrančić D (2012) LEK tuner. Jožef Stefan Institute, Ljubljana. http://dsc.ijs.si/lektuner/
23. Vrančić D, Ganchev I, Lieslehto J (2002) Tuning multivariable controllers by using magnitude optimum approach. In: Proceedings of 2002 Asian control conference (ASCC'02), Singapore, pp 2163–2167
24. Vrančić D, Huba M (2007) LEK tuner—program package for tuning PID controllers. In: Proceedings of the 2007 process control conference, Štrbské Pleso, Slovakia, pp 225-1–225-5
25. Vrančić D, Kocijan J, Strmčnik S (2004) Simplified disturbance rejection tuning method for PID controllers. In: Proceedings of the 2004 Asian control conference (ASCC'04), Melbourne, Australia, pp 491–496
26. Vrančić D, Lieslehto J, Strmčnik S (2001) Designing a MIMO PI controller using the multiple integration approach. Process Control Qual 11(6):455–468
27. Vrančić D, Strmčnik S, Juričić Đ (2001) A magnitude optimum multiple integration tuning method for filtered PID controller. Automatica 37:1473–1479
28. Vrančić D, Strmčnik S, Kocijan J (2004) Improving disturbance rejection of PI controllers by means of the magnitude optimum method. ISA Trans 43:73–84
29. Vrečko D, Vrančić D, Juričić Đ, Strmčnik S (2001) A new modified Smith predictor: the concept, design and tuning. ISA Trans 40:111–121